micro:bit

primjeri integracije microbita u nastavu s različitim međupredmetnim sadržajima

vježbe i zadatci sa Pythonom na micro:bitu

priručnik za učitelje

python™

PROFIL Klett

Nenad Hajdinjak

PRIMJERI INTEGRACIJE MICROBITA U NASTAVU S RAZLIČITIM MEĐUPREDMETNIM SADRŽAJIMA

▪ vježbe i zadatci sa Pythonom na Micro bitu ▪

priručnik za učitelje

IZDAVAČ
Profil Klett d.o.o.
Zagreb, Petra Hektorovića 2

ZA IZDAVAČA
Dalibor Greganić

DIREKTORICA UREDNIŠTVA
Petra Stipaničev Glamuzina

UREDNICI
Dragan Vlajinić
Damir Tadić

RECENZENT
Jozo Pivac

LEKTORICA
Mirjana Gašperov

FOTOGRAFIJE
arhiva Profil Kletta
Nenad Hajdinjak

PRIJELOM
Profil Klett, Zagreb

NASLOVNICA
Profil Klett, Zagreb

TISAK
Mediaprint - tiskara Hrastić d.o.o., Zagreb

Zagreb, Hrvatska, 2017.

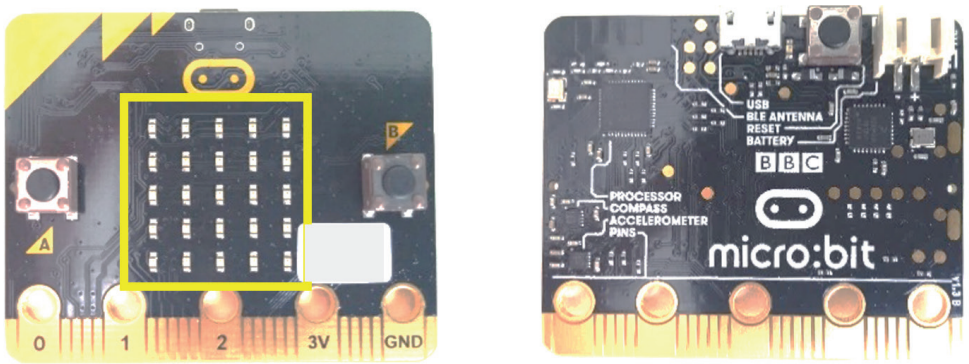
EUROPEAN
EDUCATIONAL
PUBLISHERS
GROUP

Profil je član Europskog udruženja izdavača udžbenika.

SADRŽAJ

1. KAKO „RAZGOVARATI“ S MICRO:BIT-OM?

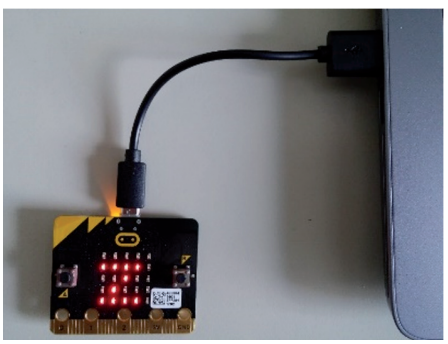
Micro:bit je minijaturno računalo (slika 1.) namijenjeno djeci da bi na zabavan i jednostavan način shvatila načela programiranja. Ono „razumije” cijelu lepezu programskih jezika, a nas trenutačno zanima Python. Inačica koja „se vrti” na micro:bitu naziva se MicroPython.



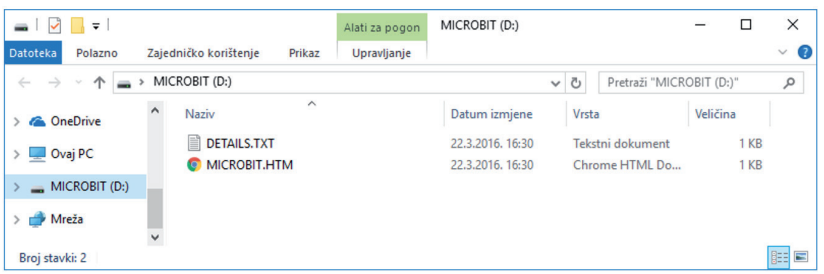
Slika 1. – micro:bit s prednje i sa zadnje strane

Način funkcioniranja micro:bita je jako jednostavan. U nekom editoru (softveru prilagođenom za pisanje naredbi u nekome programskom jeziku) napiše se kôd koji se USB kablom prebaci („flasha”) na micro:bit. Nakon toga se na zaslonu micro:bita počinje prikazivati program (slika 1. – žuti pravokutnik omeđuje zaslon koji sadržava pet redova s po pet LED-ica u svakom od njih).

Prvo uz pomoć USB kabla spojimo micro:bit na računalo (slika 2.). Računalo će ga prepoznati kao novi diskovni pogon s imenom MICROBIT (slika 3.).



Slika 2. – micro:bit je spojen na računalo i na zaslonu svijetle LED-ice koje tvore slovo z

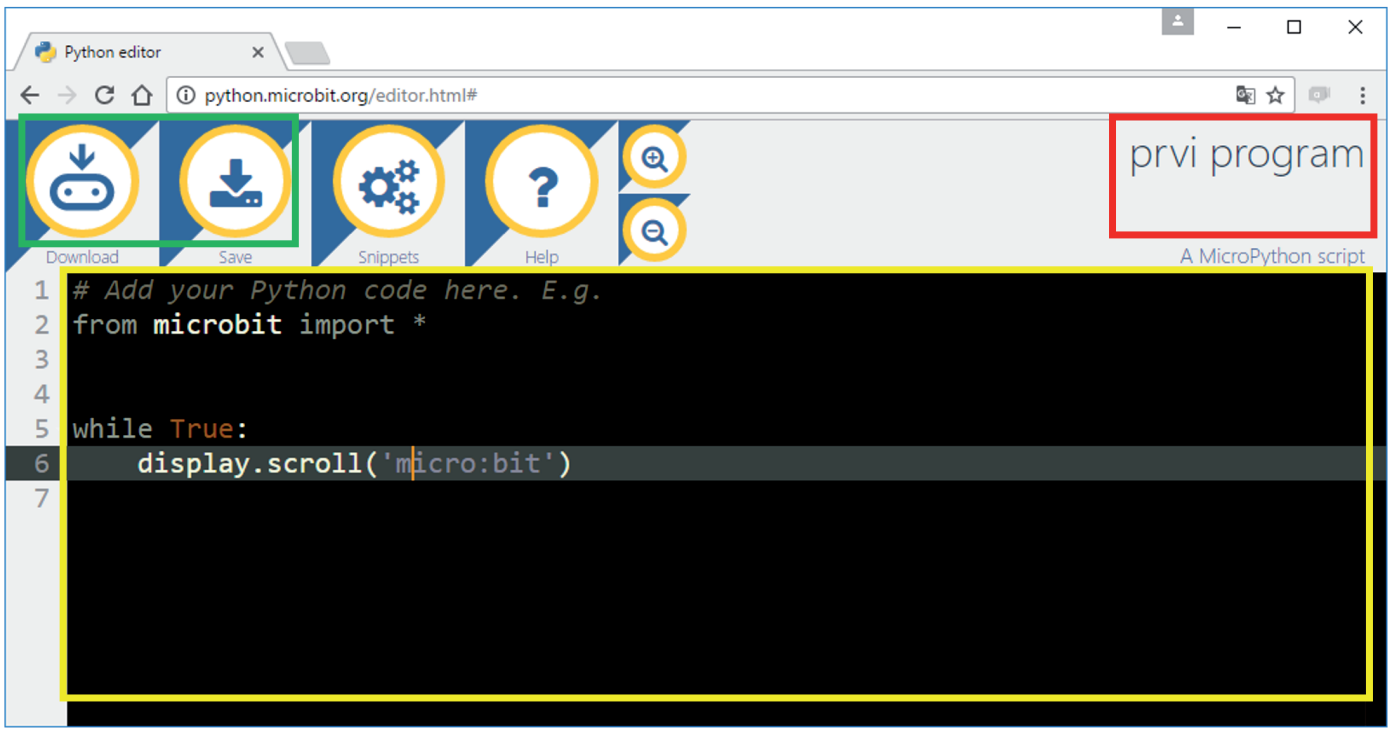


Slika 3. – ovako računalo vidi micro:bit

Ako je micro:bit spojen kao na slici 2. (mikro USB priključak spojen u micro:bit, a standardni USB priključak u računalo) i računalo ga prepoznaje (slika 3.) kao novi diskovni pogon, spremni smo za posao. Jedino moramo odabrati editor u kojem ćemo pisati. Dostupne su dvije varijante – ili ćemo kodirati „online" ili ćemo preuzeti i instalirati editor na računalo.

„ONLINE" EDITOR

Može ga se pronaći na više adresa. Jedna od popularnijih je http://python.microbit.org/editor.html (slika 4.).

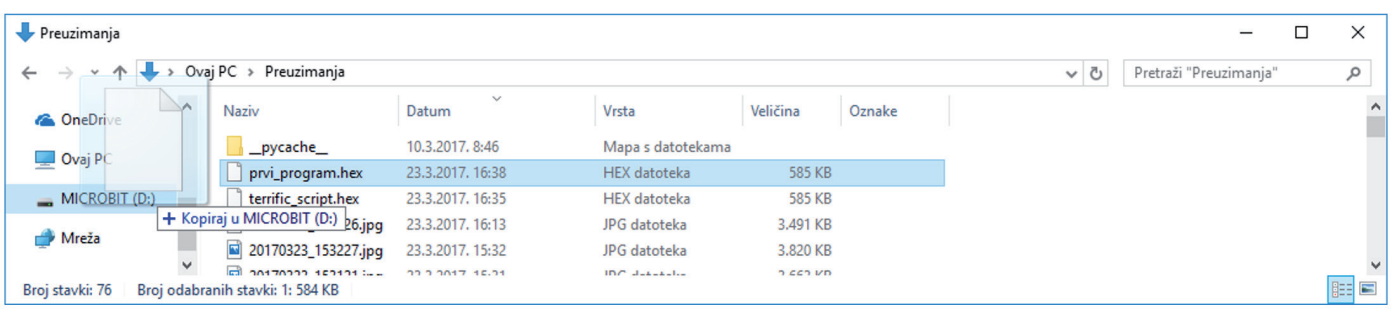


Slika 4. – izgled online editora za MicroPython

Crveni dio „online" editora (slika 4.) služi za upis naziva programa, a žuti pravokutnik za upis naredbe, kôda.

Da bismo radili u „online" editoru, moramo se koristiti gumbićima Download i Save (na slici 4. u zelenom pravokutniku).

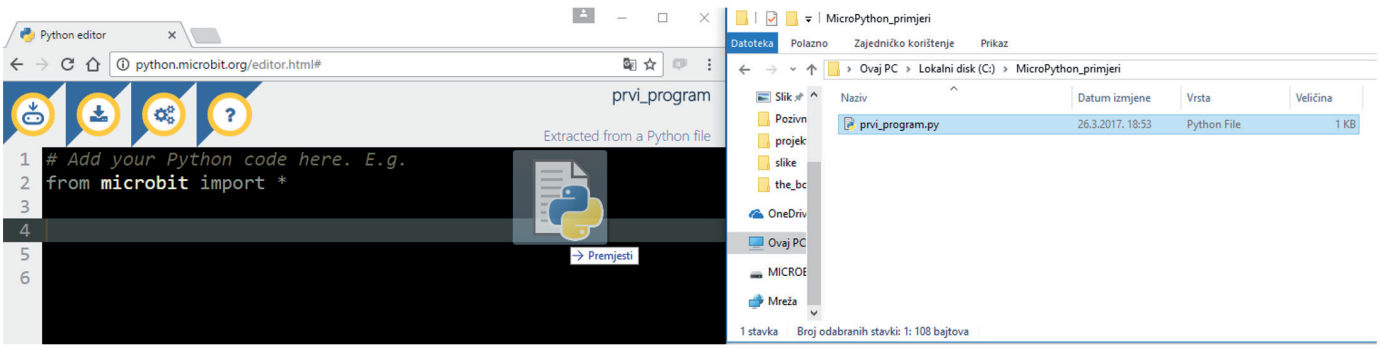
Pritiskom gumbića Download preuzimamo trenutačnu inačicu napisanog programa u .hex formatu (informacije u ASCII tekst formatu – često se upotrebljava za programiranje raznih logičkih uređaja). Datoteka najčešće završi u mapi Preuzimanja (ako nismo drukčije postavili putanju) i onda ju treba kopirati (na koji god način znamo) na micro:bit (slika 5.).



Slika 5. – metodom povuci i ispusti kopiramo .hex datoteku u micro:bit

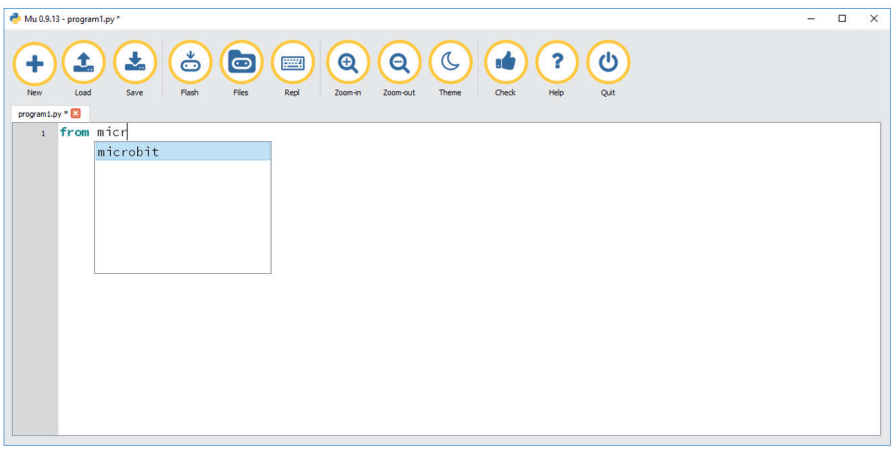
Tijekom kopiranja datoteke na micro:bit na uređaju bljeska žuta LED-ica (na zadnjoj strani mikroračunala) koja označava da je transfer u tijeku. U trenutku kada počne konstantno svijetliti, program je prebačen i počinje se izvršavati, a njegove efekte možemo vidjeti/pročitati na zaslonu.

Nakon što je program spremljen na naše računalo, možemo vrlo lako i u bilo kojem trenutku prebaciti njegov kôd natrag u „online" editor (slika 6.). Na .py program koji je spremljen u mapi pritisnemo lijevom tipkom miša (držimo ju pritisnutu) i vučemo ju u prozor „online" editora te tamo pustimo lijevu tipku miša. Naredbe iz programa će se odmah prikazati u prozoru „online" editora.



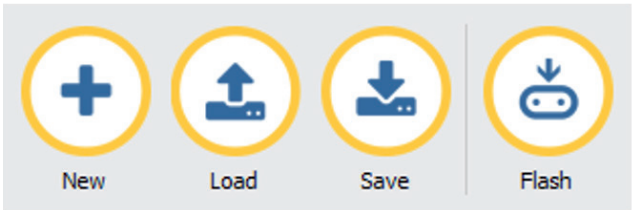
Slika 6. – prebacivanje .py programa u online editor

„OFFLINE" (INSTALACIJA NA RAČUNALU, PROGRAMIRANJE I BEZ PRISTUPA INTERNETU) EDITOR



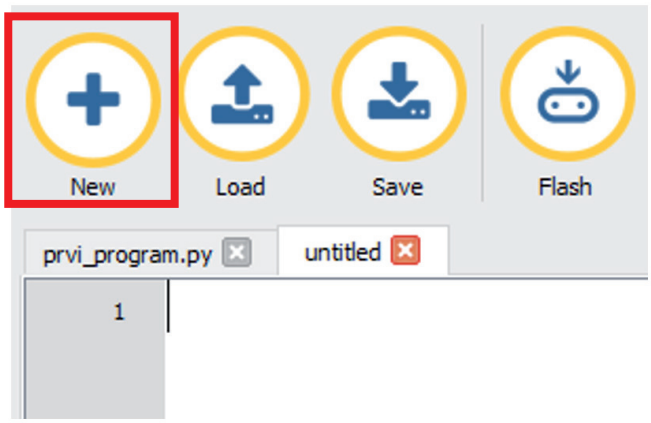
Slika 7. – MU editor

MU editor možemo preuzeti na više mjesta (https://codewith.mu/ ili http://ardublockly-builds.s3-website-us-west-2.amazonaws.com/?prefix=microbit/windows/). Važno je odabrati odgovarajuću inačicu operacijskog sustava. MU editor se ne instalira, već moramo pokrenuti preuzetu datoteku (.exe) i početi kodirati. Mi ćemo se koristiti ovim načinom programiranja jer MU editor vrlo jednostavno prebacuje program u micro:bit (samo pritisnemo gumbić Flash) i kada počnemo pisati naredbe, nudi moguće inačice naredbi te nije potrebno spajanje na internet. Navedene prednosti nisu jedine.



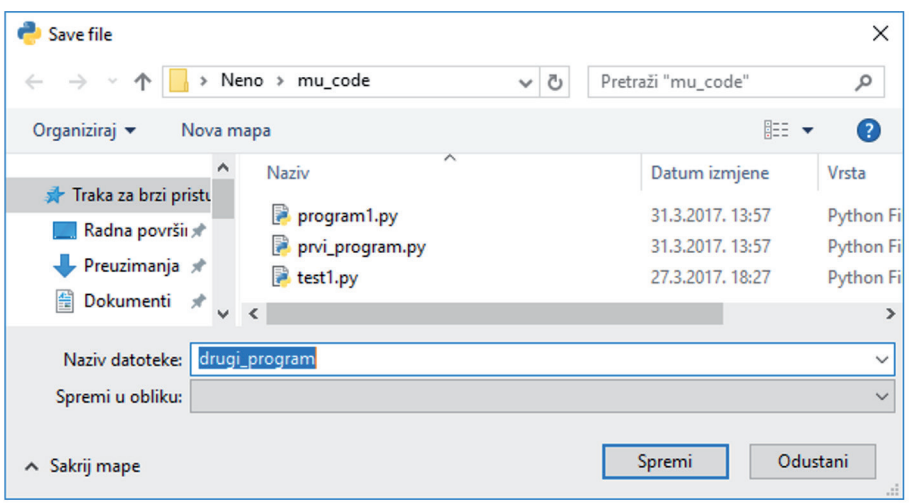
Slika 8. – najvažniji gumbići u MU editoru

Na slici 8. prikazani su najvažniji i najčešće korišteni gumbići za rad u MU editoru. Vrlo je lako shvatiti čemu služe i kako se upotrebljavaju. Pritiskom gumbića New otvara se novi tab u MU editoru (slika 9.) i ispisuje se novi program bez da to utječe na sve što smo napravili dosad.



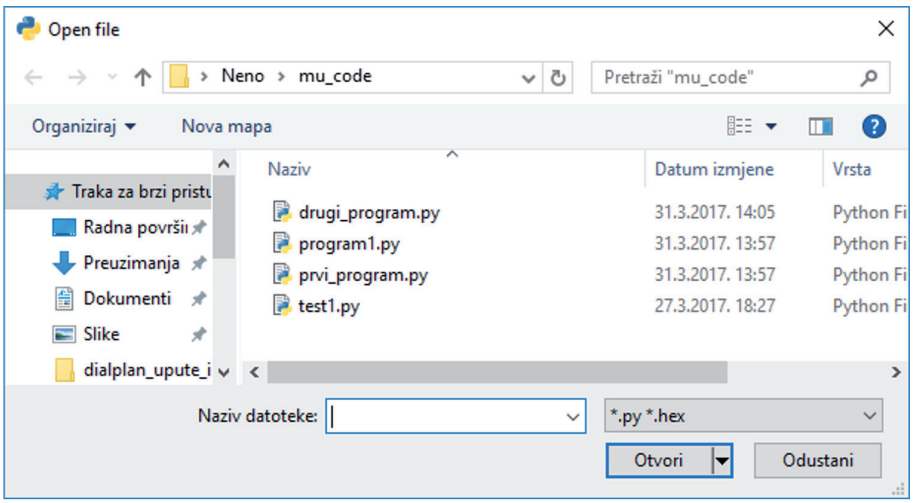
Slika 9. – pritiskom gumbića New otvara se novi tab u MU editoru (označen crvenim pravokutnikom)

Ako želimo promijeniti naziv novog taba odnosno novog programa u kojemu želimo raditi, moramo pritisnuti gumbić Save i upisati željeni naziv te ga spremiti na odabranu lokaciju (slika 10.). Nakon toga će se naziv taba/programa automatski promijeniti u naziv koji je upisan.



Slika 10. – prozor koji se otvara pritiskom gumbića Save

Pritiskom gumbića Load (prevedeno s engleskog jezika na hrvatski jezik – Učitaj) otvara se isti prozor kao i pritiskom gumbića Save, ali s nazivom Open file (slika 11.). Tu trebamo potražiti već prije napisani program te ga pritiskom gumbića Otvori učitati u prozor MU editora.



Slika 11. – Load program – odabir programa koji želimo pokrenuti/uređivati u MU editoru

Vrlo je lako shvatiti funkciju gumbića Flash, ali je prilično teško Flash precizno prevesti na hrvatski jezik. To bi značilo „zapisati". Program koji uređujemo pritiskom gumbića Flash zapisujemo u memoriju micro:bita koji ga odmah nakon toga pokušava izvršiti. Funkcije ostalih gumbića naučit ćemo kada nam to bude potrebno.

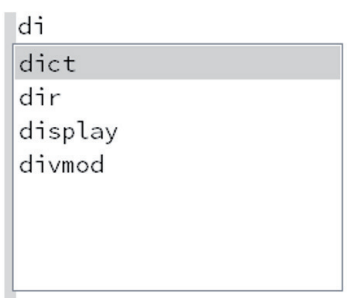
Vježba 1. – Prikaži ime svojega kućnog ljubimca na zaslonu micro:bita.

U editor trebamo napisati:

```
from microbit import *
```

Ako se želimo koristiti mogućnostima ugrađenim u micro:bit, u editor trebamo napisati naredbu (`from` – iz, `microbit` – naziv modula (biblioteka/„library") u kojem postoji unaprijed napisan kôd/naredbe, `import` – uvezi/pozovi, `*` – sve što postoji unutra) s kojom ćemo uvijek započinjati naš niz naredbi odnosno program.

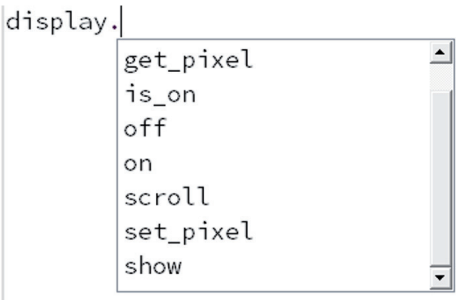
Poslužit ćemo se znanjem engleskog jezika i mogućnostima koje nudi MU editor. Prvo moramo odrediti gdje želimo prikazati ime svojeg ljubimca. S obzirom da micro:bit ima zaslon, logično je da ćemo odabrati taj način ispisa. Riječ „zaslon" prevedena na engleski jezik glasi „display" (slika 12.).



Slika 12. – već nakon upisa prvih dvaju slova riječi display prikažu se sve naredbe s tim početnim slovima

„Display" je objekt iz modula micro:bit koji, ako ga upišemo, govori da želimo nešto prikazati na zaslonu našeg mikroračunala. Taj prvi dio naredbe naziva se objekt.

Nakon objekta slijedi točka koja se upotrebljava za odvajanje objekta od ostatka naredbe odnosno željenog načina prikaza nečega (u našem slučaju) na zaslonu (slika 13.).



Slika 13. – načini prikaza na zaslonu

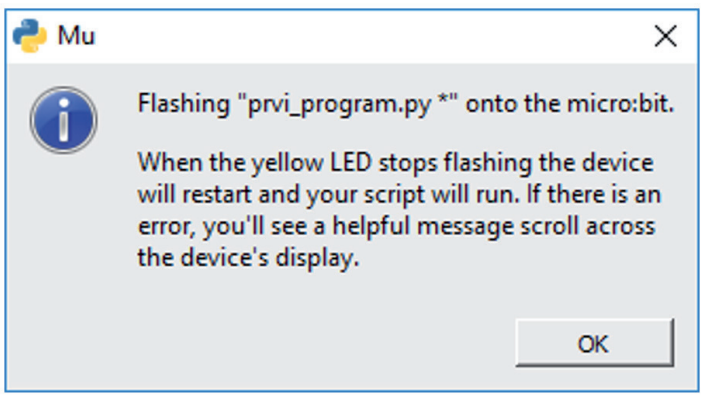
Čim upišemo točku, pojavi se popis načina rada sa željenim objektom. To nazivamo metodama (engl. „method"). Odabrat ćemo metodu „show" jer znamo da to znači „prikaži".

Dosad smo micro:bitu rekli gdje i kako, a još mu moramo napisati što želimo vidjeti na zaslonu (u našem je slučaju to ime omiljenoga kućnog ljubimca).

Taj dio obvezatno upisujemo unutar okruglih zagrada te omeđujemo dvostrukim navodnicima. To nazivamo argumentom (u našem je slučaju argument riječ „Kira"). Pokušat ćemo napisati cijeli program te ga prebaciti („flashati") na micro:bit.

```
from microbit import *
display.show ("Kira")
```

Nakon što pritisnemo gumbić Flash, na zaslonu računala pojavit će se obavijest kao na slici 14.



Slika 14. – *obavijest koja se prikazuje pri prebacivanju programa u micro:bit*

Ta obavijest kaže da je prebacivanje u tijeku (tako dugo dok žuta LED-ica svjetluca) i da će se sve eventualne obavijesti o greškama prikazati na zaslonu micro:bita, a kasnije će samo smetati.

Namjerno ćemo pogriješiti u kôdu (napisat ćemo argument bez dvostrukih navodnika) da vidimo kako se prikazuju obavijesti o greškama.

```
from microbit import *
display.show (Kira)
```

Na zaslonu micro:bita prikazat će se sljedeća obavijest:

```
Line 2 Name Error Name Kira is not defined
```

`Line 2` – pogreška u liniji 2

`NameError` – pogreška pri upisu naziva/imena funkcije/varijable

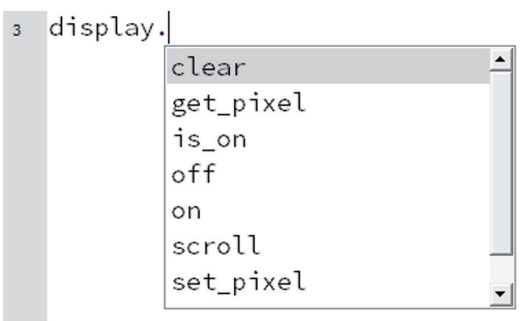
`Name Kira is not defined` – nismo prethodno definirali ništa što se naziva „Kira".

Ako unutar okruglih zagrada upisujemo nešto bez dvostrukih navodnika, micro:bit pretpostavlja da je to naziv nekakve funkcije/varijable koja je prethodno definirana. Ako nije definirana, javlja grešku. Tekst koji želimo vidjeti na zaslonu micro:bita se uvijek mora omeđiti dvostrukim navodnicima.

Vježba 2. – Napiši naziv svojega omiljenog predmeta u školi. Napiši ga tako da slova prolaze („scrollaju") po zaslonu micro:bita odnosno tako kao da se prikazuje greška na zaslonu.

Vidjeli smo u prvoj vježbi da se na zaslonu ispisuje jedno po jedno slovo i da se slova zadrže određeno vrijeme na zaslonu te ih kasnije zamijeni novo slovo. Pokušat ćemo naziv svojega omiljenog predmeta prikazati kao, npr. tekst u tramvaju, reklamu koja stalno prikazuje svoj sadržaj na način da se „vrti" u krug. To će nam biti lakše napraviti nego objasniti.

Koristit ćemo se opet „display" objektom, ali odabrat ćemo jednu od metoda rada (slika 15.).



Slika 15. – različite metode rada s display objektom

Slova se, kao što je u vježbi navedeno, moraju „scrollati" pa ćemo se zbog toga pokušati koristiti tom metodom.

```
from microbit import *
display.scroll ("Informatika")
```

„Scrollanjem" se slova „kreću" po zaslonu i da bismo mogli pročitati cijeli izraz, moramo pozornije pratiti. Vrlo je važno da na zaslonu nakon što se ispiše cijela riječ ne ostane ništa.

Zadatci za ponavljanje

1. Napravi novi tab/program pod nazivom omiljeno_jelo.py.
2. U program omiljeno_jelo.py upiši naredbe koje će ispisati omiljeno jelo slovo po slovo.
3. Pokušaj prilagoditi program tako da na kraju ne ostane vidljivo posljednje slovo.
4. Pokušaj upisati neki hrvatski znak s kvačicama (možeš isprobati sve). Što zaključuješ?
5. Napravi novi tab/program pod nazivom omiljena_tekucina.py.
6. U program omiljena_tekucina.py upiši naredbe koje će ispisati omiljeno piće tako da se „kreće" po zaslonu.
7. Spremi ispravne inačice programa.

2. KONTROLA ISPISA NA ZASLONU MICRO:BITA

U prošloj smo cjelini imali vježbu u kojoj smo na zaslonu micro:bita trebali napisati svoje omiljeno jelo, ali tako da na zaslonu ne ostane vidljivo slovo. S obzirom da još nismo naučili puno naredbi, morali smo nekako riješiti problem pa smo na posljednje mjesto, nakon naziva našega omiljenog jela, stavili razmak (prazno mjesto).

Kôd izgleda ovako:

```
from microbit import *
display.show ("Pizza ")
```

Svako se slovo prikaže pola sekunde, a na kraju ostane razmak odnosno prikaže se prazan zaslon. Riješit ćemo ispravno istu vježbu uz pomoć naredbi ugrađenih u MicroPython.

Vježba 3. – Ispiši riječ koja opisuje tvoju omiljenu zabavu na otvorenom. To ćeš učiniti tako da se svako slovo prikaže na zaslonu pola sekunde te se nakon ispisa cijele riječi zaslon obriše.

Rješenje:

```
from microbit import *
display.show ("Bicikliranje")
display.clear()
```

Nakon što ove naredbe prebacimo („flashamo") na micro:bit, na njegovu će se zaslonu prikazati riječ „bicikliranje" i to slovo po slovo, a na kraju će se izbrisati sadržaj zaslona. Moguće je primijetiti da na kraju riječi „bicikliranje" nema razmaka, već se prazan zaslon dobije metodom `Clear`. Potpuna naredba (objekt + metoda) glasi `Display.Clear()`. U okruglim zagradama nema ničega, što znači da je metoda `Clear` korištena bez argumenata.

Vježba 4. – Napiši program koji će na zaslonu micro:bita ispisati dvije stvari koje voliš raditi u zatvorenom prostoru. Neka se ispišu tako da se po zaslonu prvo „kreće" jedna stvar koju voliš, a pet sekundi kasnije druga stvar.

Da bismo riješili ovu vježbu, možemo se poslužiti trikom. Ako znamo da se svako slovo na zaslonu zadrži pola sekunde, stavimo između riječi 10 razmaka („spaceova"). S obzirom da micro:bit umjesto hrvatskih znakova s kvačicom prikazuje znak upitnika, prikazat ćemo slovo „č" slovima „ch".

Rješenje:

```
from microbit import *
display.scroll ("chitanje          programiranje")
```

Možemo je riješiti i uz pomoć funkcije koja govori micro:bitu koliko dugo zaslon mora mirovati prije nego što prijeđe na izvršavanje nove naredbe – `sleep`. Funkcija `sleep` zaustavlja izvođenje programa na određeno vrijeme, a to se vrijeme upisuje u milisekundama (unutar okruglih zagrada).

Funkcija je vrlo slična metodi, ali je samostalna i nije uz pomoć točke priključena nekom objektu.

Rješenje:

```
from microbit import *
display.scroll (“chitanje”)
sleep (5000)
display.scroll (“programiranje”)
```

Vježba 5. – Napiši program koji će na zaslonu micro:bita napisati tvoj omiljeni citat tako da se „kreće” po zaslonu. Nakon što citat „odscrolla” jednom, kreće opet i ne prestaje se „kretati” po zaslonu.

Kako bismo riješili ovu vježbu, iskoristit ćemo životnu istinu koju je izrekao Albert Einstein:

Anyone who has never made a mistake has never tried anything new., što se prevodi kao *Onaj tko nikad nije napravio pogrešku, nikad nije isprobao nešto novo.*

Jedan dio vježbe nije problem riješiti („scrollanje” teksta), ali drugi dio (ponavljanje „scrollanja” kada završi prethodno „scrollanje” i opet tako u krug) je nešto novo. Kako bi se neki dio programa ponavljao unedogled, micro:bit se koristi jednom inačicom `while` petlje. `While` petlja izvršava dio kôda dok je neki uvjet ispunjen odnosno istinit (`True`). S obzirom da uvjeta nema, a želimo da se dio kôda ponavlja, umjesto uvjeta stavit ćemo `True`. Dakle, neće se ništa ispitivati jer je već naznačena riječ `True`.

Rješenje:

```
from microbit import *
while True:
    display.scroll (“Anyone who has never made a mistake has never tried anything new.”)
```

Izuzetno je važno staviti znak dvotočja nakon riječi `True` jer na taj način dajemo programu do znanja da slijedi niz naredbi unutar `while` petlje. MicroPython će to prikazati uz pomoć uvlačenja sljedećih naredbi za četiri znaka udesno.

Ako ne stavimo dvotočje (:), točku (.), navodnike ili nešto slično, micro:bit prikazuje obavijest `SyntaxError`. „Želi nam reći” kako ne može „shvatiti” što smo napisali i da ne postoji takav oblik naredbe.

Zadatci za ponavljanje

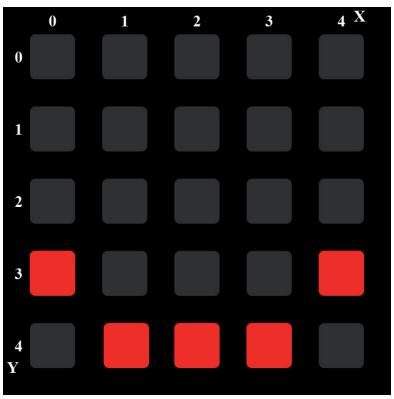
1. Napiši program koji će slovo po slovo ispisati ime omiljenog pjevača/pjevačice na zaslonu micro:bita te će na kraju obrisati zaslon.
2. Napiši program koji će poznatu skraćenicu LOL napisati slovo po slovo u razmaku od dvije sekunde između slova. Nakon što zadnje L bude prikazano dvije sekunde na zaslonu, izbrisat će se sadržaj zaslona.
3. Napiši program koji će na zaslonu micro:bita tako da „scrolla” tekst napisati naziv filma koji sadržava više od jedne riječi.
4. Napiši program koji će na zaslonu micro:bita beskonačno ispisivati naziv omiljene računalne igrice. Neka razmak između završetka i ponovnog ispisivanja naziva igrice bude tri i pol sekunde.

3. SLIKE U MICRO:BITU

S obzirom da sve što napravimo u MicroPythonu vidimo na zaslonu s 5 x 5 LED-ica, te LED-ice možemo upotrijebiti za prikazivanje svakojakih simbola, znakova i slika koje zadamo. MicroPython ima i svoj set slika koje možemo jednostavno prikazati.

```
from microbit import *
display.show (Image.SMILE)
```

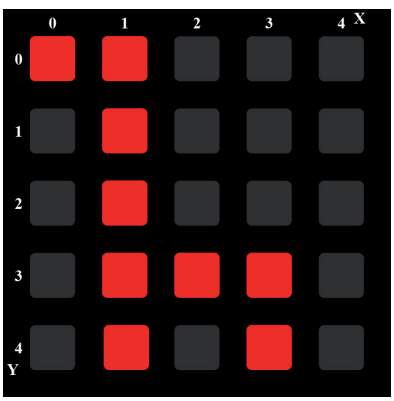
Sve je jasno. Unutar zagrada ne pišemo tekst između navodnika, već pišemo naziv objekta (`Image`) te njegov atribut odnosno njegovu inačicu (`.SMILE`) – na zaslonu micro:bita svijetle LED-ice kao na slici 16.



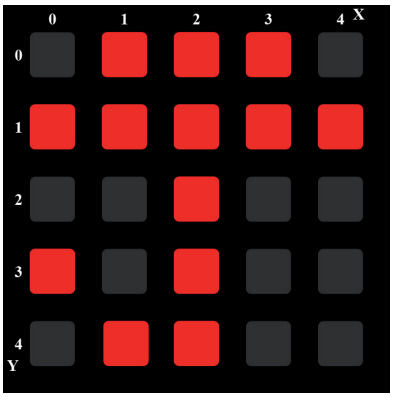
Slika 16. – Image.SMILE

Osmijeh (`SMILE`) je samo jedna od mnogo slika koje su ugrađene u micro:bit. U sljedećem su popisu sve ugrađene slike odnosno njezini atributi koji u kombinaciji s objektom `Image` ispisuju određenu sliku na zaslonu micro:bita: `HEART, HEART_SMALL, HAPPY, SMILE, SAD, CONFUSED, ANGRY, ASLEEP, SURPRISED, SILLY, FABULOUS, MEH, YES, NO, CLOCK1, CLOCK2, CLOCK3, CLOCK4, CLOCK5, CLOCK6, CLOCK7, CLOCK8, CLOCK9, CLOCK10, CLOCK11, CLOCK12, ARROW_N, ARROW_NE, ARROW_E, ARROW_SE, ARROW_S, ARROW_SW, ARROW_W, ARROW_NW, TRIANGLE, TRIANGLE_LEFT, CHESSBOARD, DIAMOND, DIAMOND_SMALL, SQUARE, SQUARE_SMALL, RABBIT, COW, MUSIC_CROTCHET, MUSIC_QUAVER, MUSIC_QUAVERS, PITCHFORK, XMAS, PACMAN, TARGET, TSHIRT, ROLLERSKATE, DUCK, HOUSE, TORTOISE, BUTTERFLY, STICKFIGURE, GHOST, SWORD, GIRAFFE, SKULL, UMBRELLA, SNAKE.`

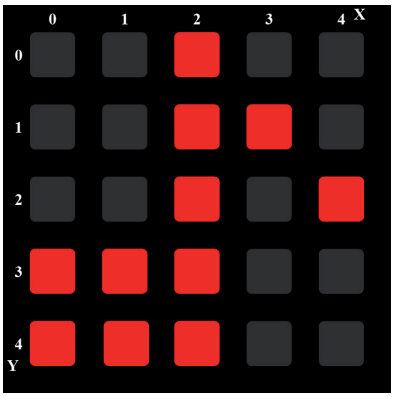
Isprobajmo nekoliko ugrađenih slika.



Slika 17. – display.show (Image.GIRAFFE)



Slika 18. – display.show (Image.UMBRELLA)

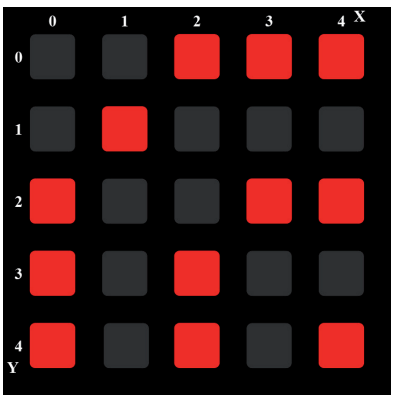


Slika 19. – display.show (Image.MUSIC_QUAVER)

Slike bismo trebali što više isprobavati jer bismo ih mogli iskoristiti u nastavku pri signalizaciji odnosno ispisu rješenja vježbe. Veći će nam izazov biti kreiranje vlastitih slika u MicroPythonu.

Vježba 6. – Na zaslonu micro:bita prikaži pojednostavljeni znak za WiFi vezu (slika 20). Neka se slika zove wifi.

Slika 20. – znak za bežičnu vezu na računalu



Slika 21. – pojednostavljeni znak za bežičnu vezu prikazan na zaslonu micro:bita

Svaka se LED-ica može postaviti na vrijednost 0 (nema osvjetljenja) i na vrijednost 9 (maksimalno osvjetljenje), što omogućuje da proizvede sliku koja nam je potrebna. Vrijednosti za LED-ice postavljaju se redak po redak. U našem bi primjeru prvi redak izgledao ovako: `00999`.

Kada bismo napisali vrijednosti za sve retke, konačno rješenje vježbe izgledalo bi ovako:

```
from microbit import *
wifi = Image ("00999:"
              "09000:"
              "90099:"
              "90900:"
              "90909:")
display.show (wifi)
```

Kreiramo sliku `wifi` te joj pridodajemo vrijednosti intenziteta osvjetljenja svake LED-ice od 0 do 9 i to za svaki redak posebno kako bismo lakše vizualizirali sliku koju moramo prikazati.

Rješenje vježbe moguće je pojednostavniti ako vizualizacija nije problem.

```
from microbit import *
wifi = Image ("00999:09000:90099:90900:90909:")
display.show (wifi)
```

Vježba 7. – Na zaslonu micro:bita prikaži prilagođeni znak za WiFi vezu. Svaka četvrtina kruga mora imati jači intenzitet od prethodne. Neka se slika zove wifi1.

U prethodnoj vježbi intenzitet osvjetljenja svake LED-ice bio je ili 0 (ne svijetli) ili 9 (maksimalno svijetli), a u ovoj nam vježbi trebaju i vrijednosti između, što znači da svaka LED-ica može poprimiti intenzitet osvjetljenja između 1 i 8. To ćemo saznanje upotrijebiti za rješavanje ove vježbe.

Rješenje:

```
from microbit import *
wifi1 = Image ("00999:"
               "09777:"
               "97755:"
               "97533:"
               "97531:")
display.show (wifi1)
```

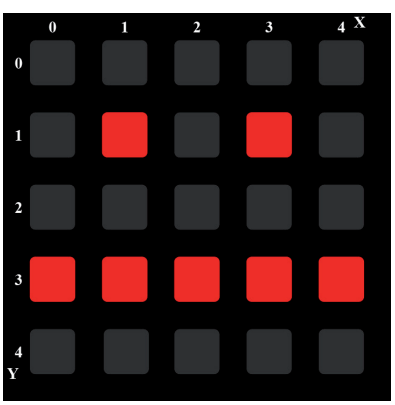
Za točku (zadnju LED-icu u posljednjem redu) upotrijebili smo intenzitet 1, za sljedeću četvrtinu kružnice intenzitet 3, pa intenzitet 5, nakon toga intenzitet 7, a na kraju intenzitet 9 koji je najjači.

Pojednostavljeno rješenje:

```
from microbit import *
wifi1 = Image ("00999:09777:97755:97533:97531:")
display.show (wifi1)
```

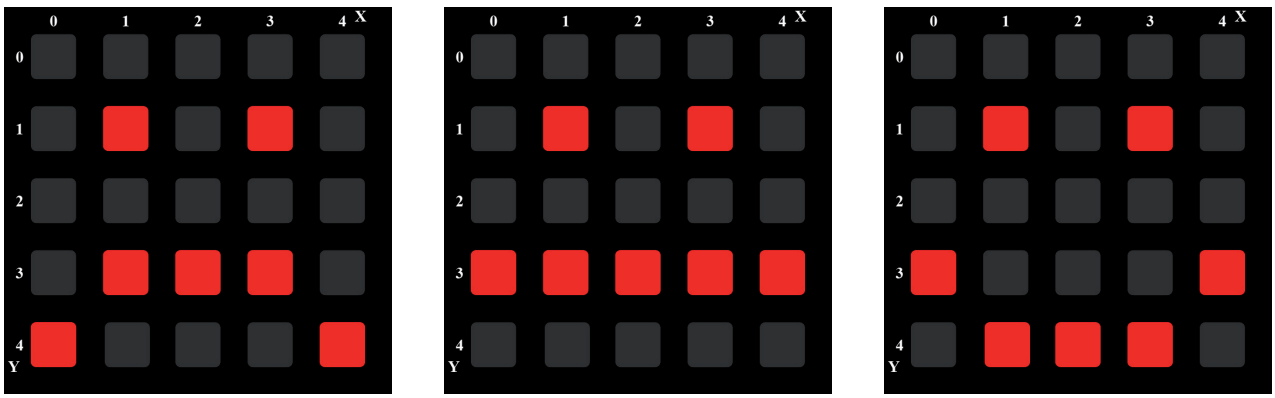
Zadatci za ponavljanje

1. Napiši program koji će na zaslonu micro:bita prvo prikazati sretno lice, a nakon toga žalosno lice. Neka razmak između tih dviju slika bude jedna sekunda. Koristi se slikama ugrađenima u MicroPython.
2. Napiši program koji će kreirati novu sliku `Svejedno` i prikazati ju na zaslonu micro:bita (slika 22.).



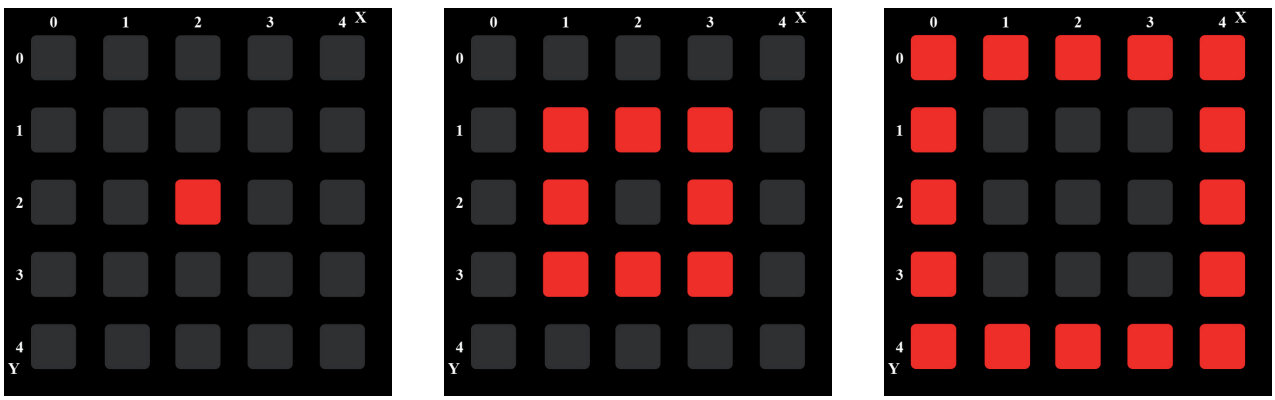
Slika 22. – Svejedno = Image

3. Napiši program koji će raditi animaciju iz žalosnog lica (SAD) u veselo lice (HAPPY), ali tako da se između njih pojavi i lice kojemu je svejedno. Neka između slika bude jedna sekunda vremenskog odmaka i neka se ponavlja unedogled.



Slika 23. – tijek animacije (`SAD`, `Svejedno`, `HAPPY`)

4. Napiši program koji će simulirati valove koje stvara kamenčić bačen u vodu. Prvo će se prikazati samo točka pa prvi val (u našem slučaju kvadrat visine i širine triju LED-ica) pa drugi val (kvadrat visine i širine pet LED-ica) – slika 24. Neka vremenski razmak između valova/slika bude 0,75 sekundi, a neka se slike zovu val1, val2 i val3. Nakon slike val3 neka se postavi prazan zaslon (isto 0,75 sekundi) i neka se to vrti u beskonačnoj petlji.

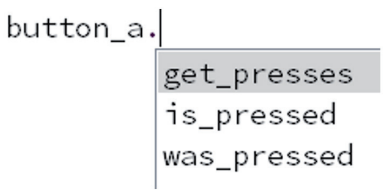


Slika 24. – redoslijed slika kod simulacije valova

5. Napiši program koji će na zaslonu micro:bita prikazati zvijezdu koja sjaji. To ćeš prikazati tako da ćeš centralnu točku (kao prvi dio u slijedu slika 24.) prikazati intenzitetom 9, LED-ice odmah do nje (kao drugi dio u slijedu slika 24). intenzitetom 6, a sve rubne LED-ice intenzitetom 3 (kao treći dio u slijedu slika 24.).

4. „INPUT" – GUMBIĆI A I B

Sve što smo dosad radili jest „OUTPUT" odnosno izlaz (ono što uređaj može prikazati/ispisati). To je lijepo i zabavno, ali za pravi program i ozbiljno programiranje potreban nam je i „INPUT" odnosno ulaz (podatci koje unosimo u uređaj, a on uz pomoć podataka nešto izračuna, usporedi, ispiše itd.).
S obzirom da nemamo klasičnu tipkovnicu, morat ćemo se poslužiti s gumbićima s lijeve i desne strane zaslona micro:bita. Koristit ćemo se zaslonom (5 x 5 LED-ica) za „OUTPUT" podataka, a gumbićima A i B za „INPUT" podataka.



Slika 25. – metode objekta gumbića A

Na slici 25. prikazane su metode odnosno načini upotrebe gumbića A (isto se odnosi i na gumbić B). `Get_presses` će dati brojčanu vrijednost koliko je puta bio pritisnut gumbić. `Is_pressed` će kao odgovor na pitanje je li u trenutku aktiviranja naredbe pritisnut gumbić izbaciti „True" (istina) ili „False" (laž). `Was_pressed` će kao odgovor na pitanje je li u određenom proteklom vremenu bio pritisnut gumbić izbaciti „True" (istina) ili „False" (laž).

Vježba 8. – Napiši program koji će provjeravati je li u pet sekundi (od pokretanja programa) u jednom trenutku bio pritisnut gumbić A.

Rješenje:

```
from microbit import *
sleep (5000)
display.scroll (button_a.was_pressed())
```

Prvi je redak razumljiv. U drugom retku funkcijom `sleep (5000)` određujemo vrijeme u kojemu će se ili neće pritisnuti gumbić A. Treći redak govori micro:bitu da prikaže tekst u pokretu i to „True" ili „False" ovisno o tome je li gumbić u prethodnih pet sekundi bio pritisnut (`button_a.was_pressed()`). Kada radimo u MicroPythonu, ne smijemo zaboraviti argumente i ako ih ima, moramo ih upisati unutar okruglih zagrada. Ako ih nema, moramo ostaviti prazno, ali moraju postojati zagrade.

Nakon što „flashamo" program, radit će pet sekundi odnosno do linije 3 kada će javiti sljedeću pogrešku: `TypeError: can't convert 'bool' object to str implicitly`.

U prijevodu je to pogreška tipa podatka. Ne može se objekt u „bool" obliku („Boolean" tip – „True" ili „False") pretvoriti izravno u „string" odnosno micro:bit na zaslonu može prikazati samo tip podatka „string" – znak. Ako želimo bilo što prikazati na zaslonu, to moramo pretvoriti u tip „string".

Nije bilo problematično ispisati tekst jer kada nešto stavimo unutar dvostrukih navodnika, micro:bit zna da su to znakovi („stringovi") i tako ih prikazuje. Za sve ostalo trebamo novu funkciju koja pretvara podatke u tip „string" – `str()`.

Sve što je unutar okruglih zagrada funkcije `str` bit će pretvoreno u znakove (naravno ako je sve ispravno upisano) te će moći biti ispisano na zaslonu micro:bita.

Ispravno rješenje:

```
from microbit import *
sleep (5000)
display.scroll (str(button_a.was_pressed()))
```

Za razliku od prethodnog rješenja u idućem je rješenju dodana funkcija `str()`. Unutar te funkcije (između okruglih zagrada) su objekti/tipovi podataka koje želimo pretvoriti u „string" (znakove) odnosno u tip podataka koji prepoznaje micro:bit.

Vježba 9. – Napiši program koji će ispisati koliko je puta u osam sekundi bio pritisnut gumbić B.

Rješenje:

```
from microbit import *
sleep (8000)
b = (button_b.get_presses())
display.show (str(b))
```

Uvodimo varijablu. Varijabla je ime (u našem slučaju `b`) kojemu može biti dodana neka brojčana ili znakovna vrijednost, a ta se vrijednost može mijenjati.

Zašto uvodimo varijablu? Kako se naše znanje povećava, povećava se i složenost zadataka odnosno problema koje trebamo riješiti. U varijablu `b` sprema se vrijednost odnosno broj koji označava koliko je puta pritisnuta tipka B i nakon toga se u „string" pretvara samo ta varijabla. Izbjegle su se trostruke zagrade, dugačke naredbe i mogućnost pogreške. U ovakvoj vježbi to nije bilo nužno, ali ta će nam spoznaja itekako pomoći u nastavku.

Operator „=" jest operator dodjeljivanja vrijednosti nekoj varijabli.

Poslužit će nam i operatori za osnovne matematičke operacije u MicroPythonu.

Operacija	Operator
Zbrajanje	+
Oduzimanje	–
Množenje	*
Dijeljenje	/

Uvijek kada se koristimo gumbićima A i B za unos („INPUT") podataka, moramo pretvoriti njihove vrijednosti u „string" (znakove) ako ih želimo ispisati („OUTPUT") na zaslonu micro:bita.

Zadatci za ponavljanje

1. Napiši program koji će provjeriti je li u sedmoj sekundi od početka pokretanja programa bio pritisnut gumbić B.
2. Napiši program koji će provjeriti je li u desetoj sekundi od početka pokretanja programa bio pritisnut gumbić A. Kako bi naznačio početak odvijanja programa, koristi se ugrađenom slikom YES.
3. Napiši program koji će u razmaku od pet sekundi od početka pokretanja programa ispisati koliko je puta bio pritisnut gumbić A. Za početak odvijanja programa koristi se ugrađenom slikom YES, a za završetak slikom NO.
4. Napiši program koji će u razmaku od tri sekunde od početka pokretanja programa ispisati koliko je puta bio pritisnut gumbić A, a zatim koliko je puta bio pritisnut gumbić B.
5. Napiši program koji će u razmaku od 10 sekundi od početka pokretanja programa zbrojiti koliko je puta bio pritisnut gumbić A i koliko je puta bio pritisnut gumbić B te koji će njihov zbroj ispisati na zaslonu micro:bita. U varijablu Z upiši zbroj pritisaka te je pretvori u „string".
6. Napiši program koji će u razmaku od šest sekundi od početka pokretanja programa provjeriti koliko je puta bio pritisnut gumbić A i koliko je puta bio pritisnut gumbić B te koji će njihovu razliku (A – B) ispisati na zaslonu micro:bita. U varijablu *R* upiši razliku pritisaka te je pretvori u „string".

5. DONOŠENJE ODLUKA

S obzirom da imamo gumbiće A i B, bilo bi odlično da svaki od njih može odrediti koji će se dio naredbi izvršiti pritiskom na jednog od njih. To se može vrlo dobro kontrolirati uz pomoć naredbe IF.
Naredba IF izvršava određene naredbe ovisno o tome je li postavljeni uvjet istinit („True") ili lažan („False").

Vježba 10. – Napiši program koji će provjeravati je li pritisnut gumbić A u trenutku završetka stanke od četiri sekunde. Ako je pritisnut, prikazat će ugrađenu sliku TSHIRT na zaslonu micro:bita, a ako nije, neće se dogoditi ništa.

Rješenje:

```
from microbit import *
sleep (4000)
pritisnut = (button_a.is_pressed())
if pritisnut==True:
    display.show (Image.TSHIRT)
```

Prvo određujemo stanku od četiri sekunde (`sleep (4000)`). Nakon toga u varijablu `pritisnut` upisujemo vrijednost „True" ili „False" (`pritisnut = (button_a.is_pressed())`) ovisno o tome je li ili nije pritisnut gumbić A. Nakon toga slijede naredba IF i njezin uvjet (`if pritisnut==True:`), a nakon uvjeta obvezatno je dvotočje (:). Moguće je primijetiti da se u uvjetu upotrebljava novi operator, dvostruki znak jednako (==) koji služi za usporedbu je li nešto jednako nečemu. Nakon dvotočja slijedi niz naredbi koje će se izvršiti ako je uvjet istinit (u našem slučaju `display.show (Image.TSHIRT)`).

Operatori uspoređivanja

Simbol	Naziv	Primjer	Izlaz/rješenje
= =	Jednako	2 = = 2	True
		1 = = 2	False
! =	Različito	2 ! = 2	False
		2 ! = 1	True
>	Veće od	2 > 2	False
		2 > 1	True
<	Manje od	1 < 2	True
		2 < 1	False
>=	Veće ili jednako	2 > = 2	True
		1 > = 2	False
<=	Manje ili jednako	2 < = 2	True
		2 < = 1	False

Moramo razlikovati operator dodjeljivanja vrijednosti (=) od operatora usporedbe jednako

(= =), što se vrlo dobro vidi u vježbi 10.

Postoji i proširena inačica IF naredbe, a naziva se IF-ELSE. Naredba IF-ELSE za razliku od naredbe IF izvršava naredbe i kada je uvjet neistinit.

Vježba 11. – Napiši program koji će provjeravati je li pritisnut gumbić B u trenutku završetka stanke od dvije sekunde. Ako je pritisnut, prikazat će ugrađenu sliku TSHIRT na zaslonu micro:bita, a ako nije, prikazat će ugrađenu sliku ROLLERSKATE.

Rješenje:

```
from microbit import *
sleep (2000)
pritisnut = (button_b.is_pressed())
if pritisnut==True:
    display.show (Image.TSHIRT)
else:
    display.show (Image.ROLLERSKATE)
```

Nastavit ćemo objašnjavati vježbu 10. Nakon retka za ispis slike TSHIRT (`display.show (Image.TSHIRT)`) slijedi naredba `else:`. Naredba `else:` mora biti u istoj okomitoj liniji kao početak naredbe IF jer će u protivnom MicroPython javiti pogrešku u sintaksi. Ne smije se zaboraviti znak dvotočja (:) na kraju retka. Iza tog dvotočja slijedi niz naredbi koje će se izvršiti ako uvjet (`pritisnut==True`) nije istinit. U našem će se slučaju, ako je zadani uvjet neistinit („False"), na zaslonu micro-bita prikazati ugrađena slika ROLLERSKATE (`display.show (Image.ROLLERSKATE)`).

Ova, ali i prethodna vježba mogu imati i kraće rješenje. Zbog lakšeg razumijevanja IF-ELSE naredbe odnosno postavljanja uvjeta pribjegli smo većem broju naredbi.

Skraćeno rješenje vježbe 11.:

```
from microbit import *
sleep (2000)
if button_b.is_pressed():
    display.show (Image.TSHIRT)
else:
    display.show (Image.ROLLERSKATE)
```

Umjesto uvjeta s operatorom uspoređivanja stavili smo funkciju `button_b.is_pressed()` koja sama po sebi može poprimiti vrijednost „False" ili „True". Provjeravamo status te funkcije.

Vježba 12. – Napiši program koji će u razmaku od pet sekundi provjeriti koliko je puta pritisnut gumbić A. Nakon toga provjeri je li taj broj jednoznamenkasti ili dvoznamenkasti i na zaslonu ispiši odgovarajuću poruku.

Rješenje:

```
from microbit import *
sleep (5000)
broj_pritisaka = button_a.get_presses()
```

```
if broj_pritisaka <=9:
    display.scroll ("jednoznamenkasti")
else:
    display.scroll ("dvoznamenkasti")
```

Zbog lakšeg objašnjavanja koristimo se varijablom `broj_pritisaka` u koju spremamo vrijednost koju računa funkcija `button_a.get_presses()`. Na početku IF-ELSE naredbe postavljamo uvjet (`broj_pritisaka <=9`). Ako bude od 0 do 9 pritisaka, uvjet će biti „True" (istinit), a ako bude 10 ili više pritisaka, uvjet će biti „False" (lažan). U skladu s tim postavljene su i naredbe što će se ispisati na zaslonu micro:bita. Prije naredbe `else` je blok naredbi koje se izvršavaju ako je uvjet istinit (`display.scroll ("jednoznamenkasti")`), a nakon naredbe `else` je blok naredbi koje se izvršavaju ako je uvjet neistinit (`display.scroll ("dvoznamenkasti")`).

Ili skraćeno:

```
from microbit import *
sleep (5000)
if button_a.get_presses() <=9:
    display.scroll ("jednoznamenkasti")
else:
    display.scroll ("dvoznamenkasti")
```

Naučimo još nekoliko dodatnih matematičkih operacija koje mogu pomoći pri rješavanju različitih problemskih zadataka.

Operator	Naziv	Primjer	„OUTPUT"/rješenje
%	Modul (ostatak kod cjelobrojnog dijeljenja)	2 % 2	0
		3 % 2	1
		4 % 2	0
		5 % 2	1
		13 % 7	6
**	Potenciranje (drugi broj označava broj koliko se puta prvi broj mora pomnožiti sam sa sobom – 5**2=5x5=25 ili 2**3=2x2x2=8)	2 ** 2	4
		3 ** 2	9
		4 ** 2	16
		5 ** 2	25
		2 ** 3	8
		2 ** 4	16
		2 ** 5	32
//	Cjelobrojno dijeljenje (kao „normalno" dijeljenje samo se sve iza decimalne točke makne)	2 // 2	1
		3 // 2	1
		4 // 2	2
		5 // 2	2
		6 // 2	3
		13 // 3	4

Vježba 13. – Napiši program koji će provjeriti koliko je puta pritisnut gumbić B u rasponu od 9 sekundi i je li taj broj paran ili neparan. Micro:bit će ispisati odgovarajuću poruku na svojem zaslonu.

Rješenje:

```
from microbit import *
sleep (9000)
broj_pritisaka = button_b.get_presses()
if broj_pritisaka % 2 !=0:
    display.scroll (“broj “)
    display.scroll (str (broj_pritisaka))
    display.scroll (“ je neparan”)
else:
    display.scroll (“broj “)
    display.scroll (str (broj_pritisaka))
    display.scroll (“ je paran”)
```

Kako znamo je li neki broj paran ili neparan? Paran je broj koji pri dijeljenju s 2 daje ostatak 0. To ćemo iskoristiti za rješavanje ovog problema. Upotrijebit ćemo operator modul (%) i staviti ga u uvjet (`broj_pritisaka % 2 !=0`). Ako je ostatak pri dijeljenju varijable `broj_pritisaka` i 2 različit od nule, onda je uvjet istinit, što znači da će program ispisati rješenje da je broj neparan. U suprotnom, ako je ostatak nula, broj koji provjeravamo je paran i takvo će se rješenje ispisati.

Zadatci za ponavljanje

1. Napiši program koji će provjeriti je li u nekom trenutku od početka pokretanja programa do isteka šest sekundi bio pritisnut gumbić B. Ako je gumbić bio pritisnut, na zaslonu micro:bita prikazat će se ugrađena slika SURPRISED.
2. Dodaj prethodnoj vježbi dio koji će na zaslonu micro:bita prikazati ugrađenu sliku CONFUSED u slučaju da gumbić B uopće nije bio pritisnut.
3. Napiši program koji će u razmaku od 12 sekundi od početka pokretanja programa prebrojiti koliko je ukupno bilo pritisaka gumbića A i B. Zbroji pritiske i provjeri je li dobiveni broj paran ili neparan. Program će ispisati odgovarajuću poruku na zaslonu micro:bita.
4. Napiši program koji će u razmaku od 10 sekundi od početka pokretanja programa provjeriti je li broj pritisaka gumbića B višekratnik broja pritisaka gumbića A. Višekratnici broja 3 su 3, 6, 9, 12,... Za sve brojeve koje možemo podijeliti s tim brojem, a da daju ostatak pri dijeljenju nula upotrebljavamo operator modul.
5. Napiši program koji će u razmaku od četiri sekunde provjeravati broj pritisaka gumbića A i B. Program će nakon toga izvršiti potenciranje tih brojeva na način da je baza uvijek veća od eksponenta, npr. 52 = 5x5=25. Broj 5 je baza, a broj 2 je eksponent. Provjeri koji je broj veći između pritisaka gumbića A i B te na temelju tih podataka postavi rješenja zadatka.

6. JOŠ NAČINA „INPUTA”

Osim gumbića A i B micro:bit može dobivati ulazne („INPUT”) informacije na različite načine. Jedan je od načina dodirivanje donjeg ruba pinova. Pinovi osjetljivi na dodir su 0, 1 i 2 (računalo uvijek počinje brojiti od nule) i to u kombinaciji s pritisnutim pinom GND. Ako jednim prstom držimo pritisnut pin GND i drugim dodirnemo, npr. pin 1, to je kao da je jednom dodirnut/aktiviran pin 1. Pokažimo to u primjeru.

Vježba 14. – Napiši program koji će prikazati ugrađenu sliku HEART kada se dodirne pin 1. Ako pin 1 nije dodirnut, pojavit će se slika HEART_SMALL.

Kompletna funkcija koja ispituje je li ili nije dodirnut neki od pinova jest `pin1.is_touched()` (u našem je slučaju 1 oznaka zadanog pina u vježbi 14. – tu može biti vrijednost 0, 1 ili 2). `Pin1` je objekt, a `is_touched` je metoda kako se taj objekt upotrebljava/daje neku vrijednost. U ovom slučaju vrijednost može biti „True” ili „False”. To možemo iskoristiti kod uvjeta u IF-ELSE naredbi.

Rješenje:

```
from microbit import *
if pin1.is_touched():
    display.show (Image.HEART)
else:
    display.show (Image.HEART_SMALL)
```

Rješenje je u potpunosti ispravno, ali kad „flashamo” program na micro:bit, uvijek ćemo dobiti sliku HEART_SMALL. To je zato što kada se pokrene program, a to je jedan mali dio sekunde, pin 1 neće biti dodirnut i prikazat će se rješenje kada pin 1 nije pritisnut. Trebamo si dati određeno vrijeme kako bismo mogli dodirnuti željeni pin. Koristit ćemo se beskonačnom `while` petljom.

Prilagođeno rješenje:

```
from microbit import *

while True:

    if pin1.is_touched():

        display.show (Image.HEART)

    else:

        display.show (Image.HEART_SMALL)
```

Sada imamo dovoljno vremena za dodirnuti željeni pin. S beskonačnom `while` petljom dobili smo program koji stalno provjerava je li pin 1 pritisnut. Ta se petlja izvrši jako brzo i opet počinje iznova i zbog toga izgleda kao da se na zaslonu micro:bita stalno vidi HEART_SMALL, a tek kada dodirnemo pin 1, slika se promijeni u HEART.

Vježba 15. – Napiši program koji će provjeriti koliko je puta dodirnut pin 0 u razmaku od 10 sekundi i koji će ispisati broj dodira. Početak odbrojavanja vremena označit ćemo ugrađenom slikom YES, a završetak slikom NO.

Koristit ćemo se funkcijom `running_time()` koja kao izlaznu vrijednost daje broj milisekundi od trenutka kada je program pokrenut.

Rješenje:

```
from microbit import *
brojac = 0
display.show(Image.YES)
while True:
    if pin0.is_touched():
        brojac = brojac + 1
    if running_time() > 10000:
        break
display.show(Image.NO)
sleep (1000)
display.scroll (str(brojac))
```

Prvo postavljamo varijablu `brojac` na nulu. U tu ćemo varijablu spremati broj dodira zadanog pina. Zašto tu varijablu treba postaviti na nulu? Treba ju postaviti na nulu jer bi pri drugom, trećem ili bilo kojem sljedećem pokretanju programa u toj varijabli postojala neka vrijednost pa rezultati ne bi bili točni.

Prije nego što uđemo u `while` petlju, koja će provjeravati dodir pina 0, trebamo postaviti sliku YES koja će biti prikazana tijekom trajanja zadanih 10 sekundi. To će biti signal/poruka da je potrebno dodirnuti ili ne dodirnuti pin 0.

Prva IF naredba će, u slučaju da je dodirnut pin 0, povećati vrijednost varijable `brojac` za 1 (`brojac = brojac + 1`). To će provjeravati svaki put kada se ponovno pokrene petlja. Još će se provjeravati koliko je vremena prošlo od pokretanja programa (`if running_time() > 10000`) i ako je to više od 10 sekundi, aktivirat će se naredba `break` odnosno izlaz iz petlje.

Naredba `break` služi za trenutačni izlazak iz petlje/programa.

Nakon izlaska iz petlje prikazat će se ugrađena slika NO koja nam govori da je vrijeme za dodir/pritisak isteklo i to sa zadrškom od jedne sekunde kako bismo ju lakše uočili. Na kraju se ispisuje rješenje (`display.scroll (str(brojac))`).

Ako gore navedeno rješenje „flashamo" na micro:bit, rezultat neće biti točan što god dodirnuli i koliko god puta dodirnuli pin 0. Rezultat će biti točan jedino ako uopće ne dodirnemo pin 0. To je tako jer `while` petlja vrti/izvršava munjevitom brzinom, a ruka odnosno prsti nisu toliko brzi i precizni pa će program za jedan dodir zabilježiti više njih i taj dodir nikad neće jednako trajati. Zato ćemo ovom programu dodati jednu zadršku nakon što se dodirne pin 0 (`sleep (500)`) kako bismo mogli maknuti prst s pina te ga ponovno dodirnuti. Prilagođavamo program ljudskoj brzini i refleksima.

Rješenje:

```
from microbit import *
brojac = 0
display.show (Image.YES)
```

```
while True:
    if pin0.is_touched():
        brojac += 1
        sleep (500)
    if running_time() > 10000:
        break
display.show(Image.NO)
sleep (500)
display.scroll (str(brojac))
```

Moguće je zamijetiti još jednu zanimljivu skraćenicu naredbe (`brojac += 1`). To dodjeljivanje vrijednosti objašnjeno je u sljedećoj tablici.

Operator	Naziv operatora	Primjer	Izlaz/Rješenje
=	Dodjeljivanje vrijednosti	X=3	3
+=	Zbrajanje	X=3 X+=2	5
-=	Oduzimanje	X=3 X-=2	1
*=	Množenje	X=3 X*=2	6
/=	Dijeljenje	X=3 X/=2	1.5
**=	Potenciranje	X=3 X**=2	9
//=	Cjelobrojno dijeljenje	X=5 X//=2	2
%=	Modul	X=5 %=2	1

Micro:bit ima ugrađen akcelerometar odnosno prevedeno na hrvatski reagira na vibracije, ali i na naginjanje u određenom smjeru.

Vježba 16. – Napiši program koji će na zaslonu micro:bita prikazivati ugrađenu sliku SQUARE_SMALL, a kada ga se protrese, nakratko će se prikazati slika SQUARE.

Funkcija koja očitava vibracije jest `accelerometer.is_gesture()`. Funkcija provjerava je li se micro:bitu dogodilo nešto od sljedećeg: „up" (okreni gore), „down" (okreni dolje), „left" (okreni lijevo), „right" (okreni desno), „face up" (okreni licem prema gore), „face down" (okreni licem prema dolje), „freefall" (slobodni pad), 3g (sila od 3g), 6g (sila od 4g), 8g (sila od 8g) ili „shake" (protresanje). Te „stringove" razumije ta funkcija i ako je nastupio neki od navedenih pokreta, ona poprima vrijednost „True" (istinito). Važno je znati koji nas pokret zanima, naznačiti ga u zagradi te funkcije i staviti ga unutar dvostrukih navodnika.

Rješenje:

```
from microbit import *
while True:
    if accelerometer.is_gesture("shake"):
        display.show (Image.SQUARE)
    else:
        display.show (Image.SQUARE_SMALL)
```

Imamo beskonačnu `while` petlju (`while True:`) u kojoj se uz pomoć IF-ELSE naredbe provjerava je li postojala gesta/pokret „shake" (protresanje) – `accelerometer.is_gesture("shake")`. Ako je postojalo protresanje, prikazat će se ugrađena slika SQUARE, a ako nije, prikazat će se ugrađena slika SQUARE_SMALL.

Vježba 17. – Napiši program koji će provjeravati je li micro:bit zakrenut u odnosu na os x. Ako nije zakrenut, prikazat će crticu (u vodoravnom je položaju), ako je okrenut ulijevo, prikazat će ARROW_W, a ako je okrenut udesno, prikazat će ARROW_E.

Rješenje:

```
from microbit import *
while True:
    if accelerometer.get_x()>50:
        display.show (Image.ARROW_E)
    elif accelerometer.get_x()<-50:
        display.show (Image.ARROW_W)
    else:
        display.show ("-")
```

U ovom se slučaju koristimo naredbom ELIF koja se dodaje IF-ELSE naredbi. IF-ELSE naredba pokriva maksimalno dva raspona vrijednosti, a ELIF pomaže ako imamo tri potrebna raspona. Naredbom ELIF se koristimo kada uvjet može imati tri različite vrijednosti, npr. veće od 100 (IF), manje od 100 (ELIF) i između tih vrijednosti (ELSE).

```
if vrijednost > 100:
elif vrijednost < -100:
else:
```

`Accelerometer.get_x()` je funkcija koja pokazuje kolika je vrijednost zakretanja osi *x*. Ta funkcija mjeri vrijednost zakretanja osi u mili-g, a može izmjeriti vrijednosti za osi *x*, *y* i *z*. Može biti negativna (u slučaju okretanja osi *x* u lijevu stranu) i pozitivna (u slučaju okretanja osi *x* u desnu stranu).

U našem smo slučaju uzeli vrijednosti od 50 i -50 jer je akcelerometar jako osjetljiv i ako bismo uzeli manje vrijednosti, možda nikad ne bismo uspjeli dobiti „crticu" odnosno vodoravno položen micro:bit. Jasno je da se moramo koristiti beskonačnom `while` petljom kako bismo u svakom trenutku mogli očitati smjer zakretanja našeg ljubimca.

Zadatci za ponavljanje

1. Napiši program koji će provjeravati je li dodirnut pin 2. Ako je dodirnut, prikazat će se ugrađena slika ARROW_S, a ako nije dodirnut, prikazat će se ugrađena slika SQUARE_SMALL.
2. Napiši program koji će za sva tri pina koji omogućuju dodir provjeravati je li neki od njih dodirnut. Ako je neki od njih dodirnut, prikazat će se njegova oznaka (0, 1 ili 2). Program cijelo vrijeme provjerava je li došlo do dodirivanja.
3. Napiši program koji će provjeravati dodir pinova i koji će nakon 20 sekundi ispisati koliko su ukupno puta pinovi bili dodirnuti.
4. Napiši program koji će provjeravati je li micro:bit nagnut prema naprijed ili prema natrag. Ispitivanje provodimo s obzirom na os y. Ako je vodoravan, na zaslonu će se prikazati okomita linija, a ako je nagnut naprijed ili natrag, strelice će prikazati smjer naginjanja.
5. Napiši program koji će provjeriti micro:bit okrenut prema gore (vidljiv zaslon) ili prema dolje. Ako je okrenut prema gore, prikazat će se ugrađena slika CHESSBOARD, a ako nije, prikazat će se ugrađena slika NO.

7. FOR PETLJA

FOR petlja je jedna od najvažnijih petlji u programiranju i za nju znamo točan broj ponavljanja. Različite varijacije FOR petlje se lako mogu prilagoditi zadanom problemu pa se uz pomoć nje može puno toga riješiti.

FOR petlja je niz ponavljanja od početnog do završnog broja gdje se broj ponavljanja regulira korakom porasta varijable.

```
FOR varijabla in range(početna_vrijednost,završna_vrijednost):
     naredbe koje se izvršavaju
```

To je lakše objasniti uz pomoć vježbe 18.

Vježba 18. – Napiši program koji će na zaslonu micro:bita ispisati brojeve od 1 do 5. Koristi se FOR petljom. Svaki broj mora ostati vidljiv jednu sekundu.

Rješenje:

```
from microbit import *
for x in range (1, 6):
    display.show (str(x))
    sleep (1000)
```

Varijabla `x` će se kretati u rasponu (`in range`) od 1 do 5 (`(1, 6)`), a završna vrijednost ne ulazi u raspon. U prvom prolazu kroz FOR petlju varijabla `x` ima vrijednost 1. Izvrše se sve naredbe unutar FOR petlje (pretvaranje vrijednosti varijable u „string" i prikazivanje iste na zaslonu – `display.show (str(x))` i stanka od jedne sekunde – `sleep (1000)`) te se petlja ponovno izvršava ispočetka, ali tako da se varijabla `x` poveća za 1. Povećanje za 1 je automatsko jer drukčije nije naznačeno. U drugom prolazu varijabla `x` ima vrijednost 2 pa se izvrše sve naredbe u FOR petlji. Nakon toga se varijabla povećava za 1 i tako sve do završne vrijednosti varijable. Kada varijabla poprimi završnu vrijednost, FOR petlja stane. Zbog toga je u našem slučaju završna vrijednost 6, a potreban nam je niz brojeva od 1 do 5. Potrebna je stanka od jedne sekunde kako bi se vidio svaki broj. Da nema stanke, zbog sporosti ljudskog oka odnosno brzine izvršavanja naredbi, bio bi vidljiv samo posljednji broj u nizu.

Vježba 19. – Ispiši sve parne brojeve od 2 do 8 na zaslonu micro:bita. Koristi se FOR petljom.

Rješenje:

```
from microbit import *
for x in range (1, 5):
    c=2*x
    display.show (str(c))
    sleep (1000)
```

Imamo FOR petlju koja mijenja vrijednosti varijable `x` od 1 do 4 i množi ih s brojem 2 kako bismo dobili parne brojeve. Međutim, to je kompliciraniji način izrade rješenja za ovaj problem. Modificiranjem FOR petlje rješavamo problem

bez dodatnih matematičkih operacija.

Rješenje:

```
from microbit import *
for x in range (2, 9, 2):
    display.show (str(x))
    sleep (1000)
```

U okrugle je zagrade dodan treći element koji označava korak varijable 2 zbog toga što je svaki drugi broj paran. Uz to je početna vrijednost postavljena na 2 jer se tako u vježbi traži. Završna vrijednost FOR petlje je 9 odnosno za 1 veći broj od posljednjeg broja koji trebamo ispisati.

```
FOR varijabla in range(početna_vrijednost, završna_vrijednost, korak):
      naredbe koje se izvršavaju
```

U FOR petlju je dodan novi element (uz početnu i završnu vrijednost petlje dodan je i broj koji označava promjenu varijable – `korak`). Ako je taj treći element upisan unutar uglatih zagrada, vrijednost se varijable u FOR petlji ne povećava za jedan nego za vrijednost `korak`. `Korak` može biti negativna ili pozitivna vrijednost.

Vježba 20. – Napiši program koji će ispisati sve višekratnike broja 3. Započni brojem 9, a završi brojem 3. Neka stanka između prikazivanja brojeva bude sekunda i pol.

Rješenje:

```
from microbit import *
for y in range (9, 2, -3):
    display.show (str(y))
    sleep (1500)
```

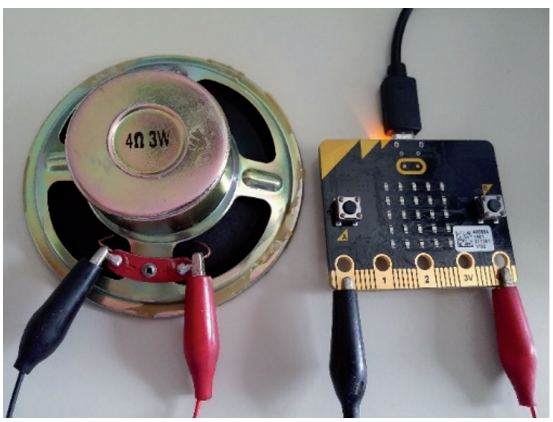
U ovom ćemo slučaju iskoristiti mogućnost FOR petlje da se „kreće" u negativnom smjeru odnosno da početna vrijednost bude veća od završne. Vrijednost varijable `y` u prvom prolazu kroz FOR petlju jest 9, a korak -3. Vrijednost varijable `y` u drugom prolazu kroz FOR petlju jest 6 (9 – 3 = 6), a u trećem prolazu 3 (6 – 3 = 3).

Zadatci za ponavljanje

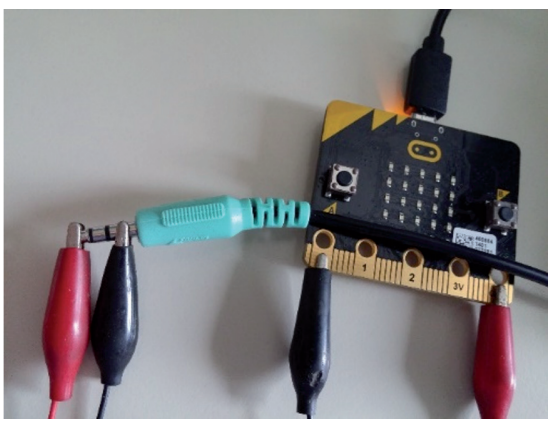
1. Napiši program koji će pet puta na zaslonu micro:bita prikazati ugrađenu sliku STICKFIGURE i to tako da se slika prikaže na pola sekunde pa nestane na pola sekunde i tako pet puta. Koristi se FOR petljom.
2. Napiši program koji će ispisati sve brojeve između 10 i 20. Koristi se metodom „scroll" i FOR petljom. Razmak između prikazanih brojeva neka bude pola sekunde.
3. Napiši program koji će ispisati sve brojeve između 100 i 90. Koristi se metodom „scroll" i FOR petljom. Razmak između prikazanih brojeva neka bude pola sekunde.
4. Napiši program koji će ispisati sve neparne brojeve od 1 do 9. Razmak između prikazanih brojeva neka bude 0,8 sekundi.
5. Napiši program koji će ispisati sve višekratnike broja 7 između 20 i 50. Razmak između prikazanih brojeva neka bude 1 sekunda.

8. MICRO:BIT I ZVUK

Najzabavnije računalo na svijetu ima i mogućnost reproduciranja zvuka. Kako bismo reproducirali zvuk, nije dovoljan samo micro:bit. Potrebna je i zvučnička jedinica ili bilo koji drugi izvor zvuka (zvučnici za računalo, autozvučnici, slušalice itd.) – slike 26. i 27.



Slika 26. – spajanje micro:bita na zvučničku jedinicu



Slika 27. – spajanje micro:bita na jack od 3,5 mm (standardni priključak računalnih zvučnika, slušalice itd.)

Iz prethodnih je slika vidljivo da pinovi 0 i GND na micro:bitu služe za spajanje na zvučnike. Druga strana tih žica spaja se na konektore zvučničkih jedinica ili na „jackove" od 3,5 mm kao što je prikazano na slikama. Treba paziti da GND pin na micro:bitu spojnim kablom povežemo na uzemljeni (GND) ili ( ) dio na priključku zvučnika ili slušalica. Isto tako, treba paziti da spojnim kablom povežemo neki od naponskih pinova s micro:bita (P0, P1 ili P2) s pozitivnim naponskim dijelom (+) na priključku zvučnika ili slušalica.

Vježba 21. – Napiši program koji će odsvirati jednu od ugrađenih melodija u micro:bit. Isprobajmo nekoliko melodija zanimljivih naziva.

Micro:bit se za reprodukciju glazbe koristi objektom `music` u kombinaciji s metodom `play` (`music.play`). U sljedećem su popisu sve ugrađene melodije odnosno njihovi atributi koje micro:bit može reproducirati u kombinaciji s objektom `music`: `DADADADUM`, `ENTERTAINER`, `PRELUDE`, `ODE`, `NYAN`, `RINGTONE`, `FUNK`, `BLUES`, `BIRTHDAY`, `WEDDING`, `FUNERAL`, `PUNCHLINE`, `PYTHON`, `RINGTONE`, `BADDY`, `CHASE`, `BA_DING`, `WAWAWAWAA`, `JUMP_UP`, `JUMP_DOWN`, `POWER_UP`, `POWER_DOWN`.

Rješenje:

```
import music
music.play (music.PRELUDE)
```

Vrlo jednostavno možemo pretvoriti micro:bit u glazbenu škrinju. Samo trebamo pozvati muzički modul (`import music`) i nastaviti s naredbom `music.play` te pozvati željenu melodiju. Melodija je jako puno. Isprobajmo ih.

Vježba 22. – Gotove melodije ne dopuštaju da iskažemo svoju kreativnost. Napiši program koji će odsvirati dio Beethowenove skladbe Für Elise.

Rješenje:

```
import music

za_elizu=["b5:3", "a#5:3", "b5:3", "a#5:3", "b5:3", "f#5:3", "a5:3", "g5:3",
"e5:3", "r:3", "g4:3", "b4:3", "e5:3", "f#5:3", "r:3", "b4:3", "d#5:3", "f#5:3",
"g5:3"]

music.play (za_elizu)
```

Za svaku notu moramo odrediti visinu tona (dio prije dvotočja) i trajanje te iste note (dio nakon dvotočja). Primjerice, "b5:3" - b označava notu, 5 označava oktavu (taj broj može biti od 0 do 8 – veći broj znači viši ton), a 3 označava trajanje tona (2 je jako kratko, a veći broj predstavlja dulje trajanje). R označava „sviranje tišine". Svaka nota mora biti unutar dvostrukih navodnika i odvojena zarezom od sljedeće note. Melodiju moramo spremiti pod nekim nazivom kako bi mogla biti reproducirana uz pomoć naredbe `music.play`.

I to je moguće pojednostaviti. Ako se oktava i trajanje ponavljaju, ne treba ih posebno naglašavati jer ih MicroPython zapamti. Tek kada se nešto od toga mora mijenjati, tada to treba i napisati.

Skraćeno rješenje:

```
import music

za_elizu=["b5:3", "a#", "b", "a#", "b", "f#", "a", "g", "e", "r:3", "g4:3", "b",
"e5:3", "f#5", "r:3", "b4:3", "d#5:3", "f#", "g"]

music.play (za_elizu)
```

Vježba 23. – Napravi zvuk hitne pomoći. Neka se reproducira 10 sekundi.

Rješenje:

```
import music
from microbit import *
while True:
    for i in range (1200, 2000, 26):
        music.pitch (i, 8)
    for i in range (2000, 1200, -26):
        music.pitch (i, 8)
```

```
if running_time () > 10000:
    break
```

U ovome smo se slučaju poslužili metodom `music.pitch`. Ta metoda kao ulaz očekuje frekvenciju koju će odsvirati (u našem slučaju varijabla `i`) te ukazuje koliko će dugo u milisekundama ta frekvencija biti reproducirana (u našem slučaju broj 8 odnosno 8 milisekundi).

Ostalo je jednostavno objasniti. Koristimo se beskonačnom `while` petljom kako bismo dobili više puta isti zavijajući zvuk. Prekinut ćemo ga nakon deset sekundi jer je tako zadano u vježbi. Zadajemo da se varijabla mora kretati u rasponu od 1200 do 2000 i mora se povećavati (nakon toga i smanjivati u istom rasponu) 26. Nakon svakog prolaska FOR petljom metoda `music.pitch` odsvira frekvenciju koja je označena varijablom i koja traje 8 milisekundi.

Zadatci za ponavljanje

1. Napiši program koji će odsvirati ugrađenu melodiju ENTARTAINER točno pet puta.
2. Napiši program koji će reproducirati ugrađenu melodiju CHASE 15 sekundi.
3. Napiši program koji će odsvirati melodiju policijske sirene u trajanju od 12 sekundi.
4. Napiši sljedeće note u micro:bit (“f4:2”, “f4:2”, “f4:2”, “a#4:4”, “d5:4”, “f4:2”, “f4:2”, “f4:2”, “a#4:4”, “d5:6”, “a#4:2”, “a#4:2”, “a4:2”, “a4:2”, “g4:2”, “g4:2”, “f4:6”) te pokušaj shvatiti koja je to melodija.
5. Okušaj se kao skladatelj i napiši vlastitu skladbu.

9. SLUČAJNI BROJEVI

Često nam za raznorazne primjene treba nekakav slučajni broj, npr. kada želimo pronaći „dragovoljca" u razredu ili kada treba odlučiti o prioritetu nečega gdje je sve jednako važno itd. S obzirom da većinom znamo engleski jezik, odmah ćemo se prisjetiti riječi „random" koja se upotrebljava i u programiranju. Dosta priče, idemo raditi.

Vježba 24. – Napiši program koji će pri svakom pokretanju ispisivati slučajan broj u rasponu između 1 i 9.

Rješenje:

```
import random
from microbit import *
slucajni = random.randint (1,9)
display.show (str (slucajni))
```

Prvo moramo pozvati modul `random` jer su u njemu sadržani objekti i metode koje su vezane uz slučajne brojeve. Nakon toga u varijablu `slucajni` trebamo spremiti broj između 1 i 9, a odabire ga metoda `random.randint`. Ta se metoda upotrebljava svaki put kada trebamo cijeli broj u nekom rasponu (u našem je slučaju raspon između 1 i 9). I početni i zadnji broj raspona uključeni su u slučajne brojeve koji se mogu odabrati.

To se može i skraćeno napisati:

```
import random
from microbit import *
display.show (str (random.randint (1, 9)))
```

Ako želimo ponovno pokrenuti program bez još jednog „flashanja" s računala na micro:bit, moramo na njegovoj zadnjoj strani potražiti gumbić „reset" (gumbić između mikro USB priključka i konektora za baterije – jedini gumbić na poleđini). Tim gumbićem ponovno započinjemo program koji je već na micro:bitu.

Zanimljiva je i funkcija `random.random ()`. Ona će ispisati slučajni broj u formatu „float" za raspon od 0.0 do 1.0, npr. `random.random()` bi mogla izbaciti broj 0.3197264 – broj između nula i jedan (iza decimalne točke je sedam decimalnih mjesta – „float" format).

Vježba 25. – Napiši program koji će slučajnim odabirom ispisivati samoglasnike.

Rješenje:

```
import random
from microbit import *
display.show (random.choice ("aeiou"))
```

U ovome se slučaju koristimo funkcijom `random.choice` koja omogućuje odabir jednog elementa u znakovnom nizu omeđenom dvostrukim navodnicima. Nije potrebna funkcija `str` jer se podrazumijeva da je, ako nešto upisujemo unutar dvostrukih navodnika, to „string" i da ne zahtijeva dodatne pretvorbe.

Vježba 26. – Napiši program koji će ispisivati slučajni neparni broj u rasponu od 1 do 9.

Rješenje:

```
import random
from microbit import *
display.show (str(random.randrange (1, 10, 2)))
```

Funkcija `random.randrange (pocetak, zavrsetak, korak)` izbacuje slučajne brojeve u rasponu od `pocetak` do `zavrsetak`, ali s jednom bitnom razlikom. U rasponu se nalaze samo brojevi koji su u razmaku od vrijednosti `korak` i to ako se ta vrijednost dodaje vrijednosti `pocetak` (npr. ako je `pocetak` vrijednost 2, `zavrsetak` je vrijednost 10, a `korak` 3, brojevi koji ulaze u izbor slučajnih su 2, 5, 8). Vrijednost `zavrsetak` ne ulazi u raspon brojeva koji mogu biti odabrani kao slučajni. Koristimo se funkcijom `str` jer se brojevi ne mogu prikazati na zaslonu micro:bita.

`Random.getrandbits (broj_bitova)` je još jedna zanimljiva funkcija iz „slučajne" skupine. Najvažnija je varijabla `broj_bitova` s kojima prikazujemo neki broj, npr. `broj_bitova = 1` – funkcija odabire slučajni broj između 0 i 1; `broj_bitova = 2` – funkcija odabire slučajni broj iz niza 0, 1, 2 i 3; `broj_bitova = 3` – funkcija odabire broj iz niza 0, 1, 2, 3, 4, 5, 6 i 7. itd.

Vježba 27. – Napiši program koji će odabrati slučajni broj u rasponu gdje se najveći broj može prikazati uz pomoć pet bitova.

Rješenje:

```
import random
from microbit import *
display.scroll (str(random.getrandbits(5)))
```

Primjenom matematike ćemo vrlo brzo shvatiti rješenje ovog zadatka – 2^5 odnosno 2 na 5-u potenciju je 32. To znači da se uz pomoć pet bitova mogu prikazati 32 broja. S obzirom da se i nula ubraja među ta 32 broja, najveći broj koji se može prikazati uz pomoć pet bitova je 31.

Mogli smo ovu vježbu riješiti i uz pomoć prethodno naučene funkcije (`random.randint(0,31)`), ali ponekad će nam biti lakše predstaviti neki broj uz pomoć broja bitova s kojima se on može prikazati, a možda će i zadatak biti upravo tako postavljen da traži određenu funkciju.

Zadatci za ponavljanje

1. Napiši program koji će ispisati slučajni broj u rasponu od 10 do 20.
2. Napiši program koji će ispisati slučajni parni broj u rasponu od 10 do 20.
3. Napiši program koji će slučajnim odabirom ispisati jedno slovo engleske abecede koje nije u hrvatskoj abecedi.
4. Napiši program koji će slučajnim odabirom ispisati četiri suglasnika hrvatske abecede.
5. Napiši program koji će svaki put kada se dogodi gesta „shake" ispisati neki od brojeva između 1 i 6 (kao da svaki put bacamo kocku).

10. NIZOVI

S nizovima smo se već susreli, a da toga nismo bili ni svjesni. Svaki niz znakova koji je omeđen dvostrukim navodnicima MicroPython prepoznaje kao „string" odnosno niz. „Stringovi" mogu biti prikazani putem zaslona micro:bita bez ikakvih dodatnih prilagođavanja.

Vježba 28. – Napiši program koji ima kreirani niz rijec te upiši najdulju riječ koje se sjetiš. Zatim iz tog niza ispiši slučajno slovo.

Rješenje:

```
from microbit import *
import random
rijec = ("prijestolonasljednica")
display.scroll (random.choice (rijec))
```

Ova riječ nije najdulja, ali je dovoljno duga. Spremili smo ju u varijablu `rijec (rijec = ("prijestolonasljednica"))`. Važno je naglasiti da je ta riječ omeđena dvostrukim navodnicima i okruglim zagradama. Te okrugle zagrade govore MicroPythonu da se radi o znakovnom nizu odnosno „stringu".

Nakon toga je potrebno samo upotrijebiti jednu od metoda za odabir slučajnih brojeva (`display.scroll (random.choice (rijec))`) i vježba je riješena.

Vježba 29. – Uz pomoć programa prebroji koliko znakova ima u „stringu" rijec.

Rješenje:

```
from microbit import *
import random
rijec = ("prijestolonasljednica")
display.scroll (str (len(rijec)))
```

Funkcija `len (naziv_niza)` izbacuje brojčanu vrijednost koja označava broj znakova u traženom nizu. U našem je slučaju broj znakova koje će funkcija izračunati 21, a riječ ima 20 slova. To je tako jer engleski jezik ne prepoznaje slovo lj kao jedan znak nego kao dva znaka.

Rezultat funkcije `len()` je brojka. Da bismo ju mogli prikazati na zaslonu micro:bita, moramo ju pretvoriti u „string" (`str (len(rijec))`).

Vježba 30. – Iz znakovnog niza rijec ispiši peti znak po redu.

Rješenje:

```
from microbit import *
rijec = ("prijestolonasljednica")
display.scroll (rijec [5-1])
```

Većina nizova ima indeksirane odnosno brojem označene članove pa ih možemo pojedinačno izdvojiti i ispisati. Funkciju koja izdvaja članove dobivamo tako da napišemo naziv niza te indeks odnosno oznaku člana koji želimo prikazati. U našem slučaju je to `rijec [5-1]`.

Zašto 5-1 ako trebamo peti znak po redu? To je zato što indeksiranje nizova počinje s nulom pa tek onda s 1, 2, 3... Zbog toga uvijek stavljamo minus 1 od željene oznake znaka.

Vježba 31. – Napiši program koji će iz „stringa" `rijec` ispisati znak koji je točno na sredini niza. Ako niz ima paran broj članova, ispisat će dva središnja člana.

Rješenje:

```
from microbit import *
rijec = ("prijestolonasljednica")
duljina = len (rijec)
parnost = duljina % 2
sredina = duljina//2
if parnost ==1:
    display.scroll (rijec [sredina])
else:
    display.scroll (rijec  [sredina-1: sredina+1])
```

Kada je u pitanju neparan broj članova niza, nemamo nikakav problem. Već iz prethodne vježbe znamo da se može ispisati određeni član niza. Izračunamo duljinu niza (`duljina = len (rijec)`), parnost (`parnost = duljina % 2`) odnosno je li ostatak dijeljenja s 2 0 ili 1 te koji će biti srednji član niza (`sredina = duljina//2`). Srednji član niza računamo cjelobrojnim dijeljenjem tako da se ne moramo zamarati s minus 1 s obzirom da indeksiranje članova niza počinjemo s nulom (npr. 3/1=1,5; 3//1=1; ako su tri člana u nizu, prvi ima indeks 0, a drugi indeks 1 odnosno upravo onaj koji smo izračunali cjelobrojnim dijeljenjem). Tu završava priča s ispisom srednjeg člana u nizu s neparnim brojem članova.

Ostaje nam još samo problem parnog niza gdje moramo ispisati dva centralna člana. Možemo ih ispisati jednog po jednog, no MicroPython nudi i funkciju koja ispisuje nekoliko članova u rasponu između naznačenih indeksa.

`naziv_liste [prvi_clan_niza : posljednji_clan_niza]` – posljednji član niza ne ulazi u odabir, npr. `niz = ("terasa"); niz [2 : 5]; ispis će biti "ras"`

U skladu s tim u naš smo kôd uvrstili naredbu koja ispisuje zadane članove (`rijec [sredina-1: sredina+1]`) – od člana `sredina-1` (moramo oduzeti 1 zbog toga jer indeksiranje počinje s nulom) pa do člana `sredina+1` (s tim da taj član ne ulazi u članove koji će biti ispisani).

Zadatci za ponavljanje

1. Napiši program koji će sadržavati znakovni niz `ner` (Najdulja Engleska Riječ) u koji ćeš upisati najdulju riječ na engleskome jeziku koje se sjetiš. Nakon toga će program provjeriti njezinu duljinu i ispisati ju.
2. Napiši program koji će sadržavati znakovni niz `no` (Naziv Otoka) u koji ćeš upisati otok kojeg se sjetiš (ne mora biti hrvatski otok) s najviše znakova u njegovu imenu. Nakon toga će program ispisati prvo i posljednje slovo u razmaku od dvije sekunde.
3. Napiši program koji će sadržavati znakovni niz `ng` (Naziv Grada) u koji ćeš upisati grad kojeg se sjetiš s najviše znakova u njegovu nazivu. Program će ispisati sve znakove između druge i šeste pozicije (uključujući i te dvije pozicije). Koristi se FOR petljom.
4. Napiši program koji će sadržavati znakovni niz `nd` (Naziv Države) u koji ćeš upisati državu koje se sjetiš s najviše slova u njezinu nazivu. Nakon toga će program 10 sekundi pratiti pritiske gumbića A i B te broj pritisaka spremiti u varijable `pA` i `pB`. Usporedit će vrijednosti u varijablama te ispisati znakove u njihovu rasponu, npr. `Pa=5`, `Pb=10`, `nd=("Bjelorusija")`, ispis će biti `"orusij"`. Ispis će biti isti i ako se zamijene vrijednosti varijabli `pA` i `pB`. Program će uspoređivati vrijednosti i uvijek će kao početnu vrijednost uzeti nižu vrijednost, a kao završnu višu vrijednost.
5. Napiši program koji će sadržavati znakovni niz `nr` (Naziv Rijeke) u koji ćeš upisati rijeku koje se sjetiš s najviše slova u njezinu nazivu. Nakon toga će program ispisati drugu polovinu niza. Ako se niz sastoji od neparnog broja članova, ispisat će se središnji član i svi nakon njega.

11. LISTE

U prethodnoj cjelini pričali smo o nizovima odnosno „stringovima". Oni se relativno jednostavno mogu prikazati na zaslonu micro:bita. Sve ostale inačice nekakvih sljedova trebaju određene preinake da bi ih se moglo prikazati. Jedan je od često korištenih tipova podataka lista.
Lista je svaki niz koji je napisan unutar uglatih zagrada, npr. treba napisati listu `radni_tjedan`; `radni_tjedan = ["ponedjeljak", "utorak", "srijeda", "cetvrtak", "petak"]`.

Vježba 32. – Napiši program koji će u listu `djevojcice` upisati imena svih djevojčica u tvojem razredu. Nakon toga će se slučajnim odabirom, svaki put kad pritisneš gumbić B, ispisati jedno ime.

Rješenje:

```
import random
from microbit import *
djevojcice = ["Lara", "Tanja", "Marija", "Alica", "Ana", "Mia", "Lucija"]
while True:
    if button_b.is_pressed():
        display.scroll (random.choice(djevojcice))
```

Prvo smo kreirali listu s popisom imena djevojčica u razredu pod imenom `djevojcice`. Nakon toga smo u beskonačnoj petlji postavili uvjet o pritisnutosti gumbića B. Ako je gumbić B pritisnut, ispisuje se slučajni član liste `djevojcice` i to svaki put kada se gumbić B pritisne.

Vježba 33. – Napiši program koji će provjeriti koliko je puta pritisnut gumbić A u 10 sekundi i koji će spremiti tu vrijednost u varijablu n. Nakon toga će se ispisati n-ti član liste `djevojcice`. Ako je broj n veći od broja članova liste, ispisat će se poruka da lista nema toliko članova.

Svaka je lista, a u prethodnoj smo cjelini naučili da se to odnosi i na niz, indeksirana. Prisjetimo se. To znači da svaki član, s obzirom na poziciju u listi, dobiva svoj redni broj počevši od nule. Prema tome, ime Lara ima poziciju 0, Tanja poziciju 1 itd. Indeksiranje također može biti i negativno – tada je zadnji član indeks 1, predzadnji 2 itd.

Rješenje:

```
from microbit import *
djevojcice = ["Lara", "Tanja", "Marija", "Alica", "Lana", "Mia", "Lucija"]
while True:
    if running_time ()>10000:
        break
n = button_a.get_presses()

display.scroll (djevojcice[n-1])
```

Nakon definiranja članova liste korisniku dajemo vrijeme da pritišće gumbić A – to dobivamo beskonačnom petljom (`while True:`) iz koje program izlazi nakon 10 sekundi (`if running_time ()>10000: break`). Nakon toga u varijablu `n` upisujemo broj pritisaka gumbića A (`n = button_a.get_presses()`) te ispisujemo n-1 po redu član liste `djevojcice` (`display.scroll (djevojcice[n-1])`) i pazimo na uglate zagrade. Zašto n-1? To je zato što indeksiranje počinje s nulom pa je prvi član nulti, drugi je prvi itd.

Na taj je način problem samo dijelom riješen. Programu trebamo dodati i dio kada je broj pritisaka gumbića A veći od brojeva članova liste. U tom slučaju treba ispisati poruku o greški.

Funkcija koja ispisuje broj članova liste je `len (naziv_liste)`. To ćemo ukomponirati u naš program.

Konačno rješenje:

```
from microbit import *
djevojcice = [“Lara”, “Tanja”, “Marija”, “Alica”, “Lana”, “Mia”, “Lucija”]
br_cl=len(djevojcice)
while True:
    if running_time ()>10000:
        break
n = button_a.get_presses()
if n > br_cl:
    display.scroll (“Greska. Nema toliko clanova u listi.”)
else:
    display.scroll (djevojcice[n-1])
```

Broj članova liste upisujemo u varijablu `br_cl` (`br_cl=len(djevojcice)`) i taj broj uspoređujemo s n (broj pritisaka gumbića A) te na temelju istinitosti uvjeta pokrećemo jednu od dviju mogućnosti: ispis n-tog člana liste ili ispis o greški.

Vježba 34. – Napiši program koji će imati listu `ksjd` (skraćeno od Komu Se Ja Divim) u koju ćeš upisati svoje omiljene sportaše, muzičare, glumce itd. Neka lista ima barem šest članova. Program će slučajnim odabirom ispisati tri različita člana liste.

Rješenje:

```
from microbit import *
import random
ksjd = [“Jobs”, “Musk”, “Einstein”, “Mercury”, “Lennon”, “Pele”, “Eastwood”]
for i in range (1,4):
    display.scroll (random.choice(ksjd))
```

Ovo je naizgled lagan zadatak. Kreiramo listu – ništa lakše (`ksjd = [“Jobs”, “Musk”, “Einstein”, “Mercury”, “Lennon”, “Pele”, “Eastwood”]`). Nadam se da smo ih sve prepoznali. Nakon toga uz pomoć triju prolaza kroz FOR petlju ispišemo tri člana liste. Pokušajmo više puta resetirati micro:bit. Može se dogoditi da naiđemo na problem i da se ponekad isti član ispiše dva puta.

Taj ćemo problem riješiti naredbom `remove` tako da kada odaberemo neki član, odmah ga nakon toga izbacimo iz liste.

Konačno rješenje:

```
from microbit import *
import random
ksjd = ["Jobs", "Musk", "Einstein", "Mercury", "Lennon", "Pele", "Eastwood"]
for i in range (1,4):
    odabrani = random.choice(ksjd)
    display.scroll (odabrani)
    ksjd.remove (odabrani)
```

Pri svakom prolasku kroz FOR petlju spremamo slučajni odabir u varijablu `odabrani` (`odabrani = random.choice(ksjd)`). Nakon toga se ispiše sadržaj varijable (`display.scroll (odabrani)`). Problem višestrukog slučajnog odabira istog člana liste riješit ćemo izbacivanjem odabranog člana iz liste (`ksjd.remove (odabrani)`) prije nego što ponovno odaberemo novoga slučajnog člana.

Vježba 35. – Dodaj još jednom `ksjd` listi člana prema želji (mora biti jedan od već navedenih – mi ćemo još jednom dodati član "`Jobs`"). Koristi se naredbom za dodavanje listi i na kraju provjeri postoje li u novoj `ksjd` listi dva člana. Ispis će biti broj koji pokazuje koliko se puta u listi pojavljuje član "`Jobs`".

Rješenje:

```
from microbit import *
ksjd = ["Jobs", "Musk", "Einstein", "Mercury", "Lennon", "Pele", "Eastwood"]
ksjd.append ("Jobs")
display.scroll (str(ksjd.count ("Jobs")))
```

Funkcija `append` (`naziv_liste.append ("član")`) dodaje željeni član na kraj liste. Funkcija `count` (`naziv_liste.count ("član")`) ispisuje koliko se put traženi član pojavljuje u listi.

Ako se vratimo na vježbu, vidimo praktične primjene prethodno navedenih funkcija. Dodajemo isti član listi (`ksjd.append ("Jobs")`) te nakon toga provjeravamo koliko se puta taj član pojavljuje (`ksjd.count ("Jobs")`) u `ksjd` listi.

Možemo upotrebljavati sljedeće funkcije s listama:

`naziv_liste.index ("član")`
- Ispisuje na kojem se mjestu u listi nalazi neki član te liste.
- Primjerice, `ksjd.index ("Jobs")` bi izbacio rješenje 0. Indeksiranje počinje s nulom pa pravu poziciju dobivamo tako da dobivenom rješenju dodamo broj 1.

`naziv_liste.insert (indeks, "član")`
- Dodaje se željeni član na točno određeno mjesto (`index`).

`naziv_liste [indeks]`
- Ispisuje člana liste koji je indeksiran kao vrijednost varijable `indeks`.
- Primjerice, `ksjd [2]` će ispisati trećeg po redu člana – u našem će slučaju to biti niz `Einstein`.

`naziv_liste [pocetak : zavrsetak]`
- Od stare liste kreira novu listu koja započinje s indeks pozicijom `pocetak`, a završava s indeks pozicijom `zavrsetak-1` (vrijednost `zavrsetak` nije uključena u novu listu).
- Primjerice, `ksjd [2 : 5]` bi kreirao istoimenu listu s članovima `["Einstein", "Mercury", "Lennon"]`.

Zadatci za ponavljanje

1. Napiši program koji će kreirati listu `djecaci` te u nju upiši imena svih dječaka iz razreda. Iz te će se liste slučajnim odabirom, kada se pritisne gumbić A, ispisati jedno ime.
2. Dopuni prethodni program na način da prebroji koliko članova ima lista `djecaci`.
3. Napiši program koji će ispisati drugi i pretposljednji član liste `djecaci`.
4. Napiši program koji će sadržavati listu `razbibriga` i koji će u nju upisati neparan broj članova. Svaki će član označavati nešto što volimo raditi. Program će ispisati srednji član liste te njegovu stvarnu poziciju u listi, što znači da neće ispisati indeks nego poziciju ako počnemo brojiti od 1.
5. Napiši program koji će sadržavati tri liste. Neka se prva zove `deserti` i u nju upiši svojih pet omiljenih deserata. Neka se druga zove `glavno_jelo` i u nju upiši svojih omiljenih pet glavnih jela. I neka se treća zove `osvjezenje` i u nju upiši pet pića koje voliš popiti nakon objeda. Program će pri svakoj gesti „shake" ispisati po jednog slučajnog člana svake liste.
6. Napiši program koji će simulirati izvlačenje brojeva lota. Neka se izvlače brojevi od 1 do 15. Dobitna kombinacija bit će pet različitih brojeva.

12. WHILE PETLJA

Već se neko vrijeme u vježbama koristimo FOR petljom koja zaista može riješiti različite probleme. Čemu onda još jedna petlja? `While` petlja za razliku od FOR petlje nema definirani broj ponavljanja odnosno FOR petlja će se uvijek ponoviti određeno puta, a `while` petlja se ponavlja dok zadani uvjet nije istinit.

Već smo se susretali s jednim oblikom `while` petlje, a to je beskonačna petlja. Njezina je karakteristika što nema postavljen uvjet koji treba ispuniti nego je umjesto njega upisana riječ „True". „True" je istina odnosno ispunjen je uvjet pa se petlja cijelo vrijeme ponavlja dok se na određeni način ne prekine.

Vježba 36. – Napiši program koji će cijelo vrijeme na zaslonu naizmjence prikazivati ugrađenu sliku DUCK i GIRAFFE u razmaku od sekunde i pol.

Rješenje:

```
from microbit import *
while True:
    display.show (Image.DUCK)
    sleep (1500)
    display.show (Image.GIRAFFE)
    sleep (1500)
```

Početak beskonačne petlje izgleda ovako: `while True:`. Moramo paziti gdje je veliko slovo, a gdje malo i nikako ne smijemo zaboraviti znak dvotočja na kraju. Nakon toga slijede naredbe koje će se ponavljati. Moramo obratiti pozornost da sve naredbe budu uvučene, a MicroPython to radi automatski.

S obzirom da se radi o beskonačnoj petlji, izvođenje programa može se prekinuti jedino tako da se u micro:bit učita novi program ili uz pomoć naredbe `break`.

Vježba 37. – Napiši program koji će naizmjence (svakih pola sekunde) prikazivati ugrađene slike HEART i HEART_SMALL. Nakon 10 sekundi se izvođenje programa prekida tako da na zaslonu nema ničega.

Rješenje:

```
from microbit import *
while True:
    display.show (Image.HEART_SMALL)
    sleep (500)
    display.show (Image.HEART)
    sleep (500)
    if running_time () > 10000:
        break
display.clear ()
```

Vježba bi se mogla riješiti uz pomoć beskonačne petlje tako da uz pomoć naredbe `if` provjeravamo kada će proći 10 sekundi od početka izvođenja programa (`if running_time () > 10000`) te u tom slučaju izađemo iz petlje (`break`).

Pokušat ćemo taj problem riješiti bez beskonačne `while` petlje. Koristit ćemo se `while` petljom u kojoj uvjet može biti istinit ili lažan te se na temelju njega `while` petlja izvršava ili zaustavlja.

Rješenje uz pomoć `while` petlje:

```
from microbit import *
while running_time () < 10000:
    display.show (Image.HEART_SMALL)
    sleep (500)
    display.show (Image.HEART)
    sleep (500)
display.clear ()
```

Takav način rješavanja problema zahtijeva pažljivo postavljanje uvjeta. `While` petlja se izvršava toliko dugo dok je uvjet istinit. Tako smo i mi postavili uvjet - `running_time () < 10000` odnosno svaki početak prolaska kroz petlju provjerava uvjet. Ako program traje (`running_time ()`) manje od 10 sekundi (odnosno 10 000 milisekundi), uvjet je istinit (npr. `3000<10000`, što je istinito ili „True") i petlja će se izvršiti. Ako program traje dulje od 10 sekundi (`10500<10000` je neistinito ili „False"), uvjet više nije ispunjen odnosno lažan je i `while` petlja prekida svoje izvođenje.

Vježba 38. – Napiši program koji će na zaslonu micro:bita cijelo vrijeme prikazivati svih 25 LED-ica osvijetljenih intenzitetom 3. Ako dodirnemo pin 1 ili 2, taj će se intenzitet pojačati (na 9). Ako dodirnemo pin 0, zaustavit će se `while` petlja i bit će prikazan samo prazan zaslon.

Rješenje:

```
from microbit import *
slabi_i = Image ("33333:33333:33333:33333:33333")
jaki_i = Image ("99999:99999:99999:99999:99999")
while pin0.is_touched ()== False:
    display.show(slabi_i)
    if pin1.is_touched ():
        display.show (jaki_i)
        sleep(100)
    if pin2.is_touched ():
        display.show (jaki_i)
        sleep(100)
display.clear ()
```

Prvo sami napravimo tražene slike i to jednu s intenzitetom 3 svih LED-ica (`slabi_i`) i jednu s intenzitetom 9 (`jaki_i`). `While` petlja će provjeravati je li dodirnut pin 0. `While` petlja se vrti dok je uvjet istinit pa će u ovoj vježbi istina biti da pin 0 nije dodirnut (`pin0.is_touched ()== False`). Kada se pin 0 dodirne, program će izići iz petlje i izbrisati zaslon.

Ostalo je sve poznato. Ako nije pritisnut nijedan pin, prikazuje se slika `slabi_i`, a u slučaju pritiska pina 1 (`if pin1.is_touched ()`) ili pina 2 (`if pin2.is_touched ()`) prikazat će se slika najjačeg intenziteta (`display.show (jaki_i)`). S obzirom da se `while` petlja vrti vrlo velikom brzinom, ostavili smo malo vremena (jednu desetinku) da se vidi slika (`sleep(100)`).

Vježba 39. – Napiši program koji će dopustiti da unutar pet sekundi pritišćeš gumbić A. Ako je taj broj pritisaka manji od 10 ili jednak 10, `while` petlja će se još jednom pokrenuti i opet dopustiti pet sekundi pritiskanja. U vrijeme kada je pritiskanje dopušteno na zaslonu će biti prikazana ugrađena slika YES, a kada vrijeme istekne na tri desetinke sekunde, prikazat će se ugrađena slika NO.

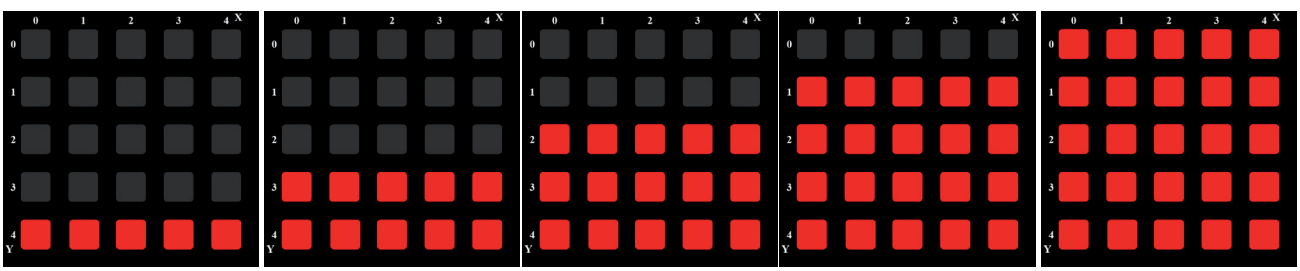
Rješenje:

```
from microbit import *
while button_a.get_presses () < 11:
    display.show (Image.YES)
    sleep (5000)
    display.show (Image.NO)
    sleep (300)
```

U ovome slučaju imamo `while` petlju s uvjetom koji može biti i istinit i neistinit. Ovisi o tome koliko smo puta pritisnuli gumbić A (`button_a.get_presses () < 11`). Dok je broj pritisaka gumbića 10 ili manji od 10, ova je usporedba istinita te se `while` petlja ponovno pokreće. U petlji imamo pet sekundi za pritiskanje gumbića A koje je prikazano ugrađenom slikom YES (`display.show (Image.YES) sleep (5000)`). Kada vrijeme istekne, nakratko se prikaže ugrađena slika NO i provjerava se uvjet te ovisno o njemu `while` petlja ili stane ili se ponovno pokreće.

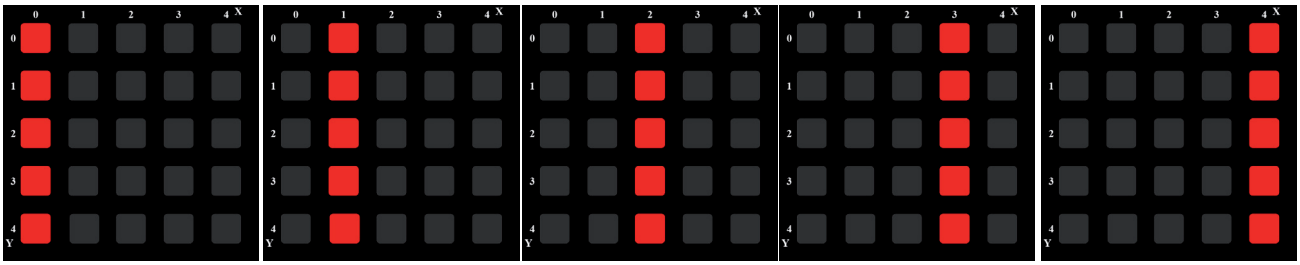
Zadatci za ponavljanje

1. Napiši program koji će u beskonačnoj petlji prikazivati animaciju kao kada se puni baterija. Prvo će se pojaviti zadnja linija na zaslonu micro:bita. Nakon tri desetinke svijetlit će dvije vodoravne linije LED-ica. Nakon toga će u istome vremenskom razmaku svijetliti tri linije pa četiri linije pa pet linija. Nakon toga sve slijedi ispočetka (slika 28.).



Slika 28. – animacija indikatora punjenja baterije

2. Napiši program koji će 12 sekundi prikazivati animaciju okomite linije kako prolazi zaslonom micro:bita. Prvo će se pojaviti prva okomita linija na zaslonu micro:bita pa s 400 milisekundi zakašnjenja druga linija itd. (slika 29.).



Slika 29. – linija prolazi po zaslonu slijeva nadesno

3. Napiši program koji će „vrtiti" `while` petlju dok ne bude dodirnut pin 1. Kada se dodirne pin 1, `while` petlja završava s ispisanom porukom „kraj". Ako ostali pinovi budu dodirnuti, prikazat će se ugrađena slika NO. Aktivnost petlje odnosno njezin početak prikazat će se ugrađenom slikom YES.
4. Napiši program koji će unutar sedam sekundi, koliko će trajati `while` petlja, provjeravati koliko je puta pritisnut gumbić B. Ako je broj pritisaka jednoznamenkasti, prekida se `while` petlja i ispisuje se vrijednost odnosno broj pritisaka gumbića B. Ako je broj dvoznamenkasti, petlja nastavlja s radom.
5. Napiši program koji će ti dopustiti da pritišćeš gumbiće A i B četiri sekunde. Ako je broj pritisaka tipki A i B jednak, petlja će prekinuti s izvršavanjem i ispisat će odgovarajuću poruku. Ako je broj pritisaka različit, petlja nastavlja s radom.

13. LOGIČKE OPERACIJE

Ponekad ne možemo odlučiti što i kako napraviti samo uz pomoć jednog parametra. Katkad treba uzeti u obzir nekoliko čimbenika i nekoliko uvjeta kako bismo što preciznije i točnije mogli odrediti daljnji tijek radnji.
Kada to želimo prikazati računalnim jezikom, pribjegavamo logičkim operacijama ili Boolovoj algebri koja se temelji na logičkim izjavama koje mogu biti istinite („True") ili lažne („False"). Važno je samo da izjava ne bude subjektivna.

Izjava *Ovaj je automobil lijep.* je subjektivna jer ne možemo znati koliko se kojoj osobi sviđa koji automobil.
Izjava *Ovaj je automobil crvene boje.* je logička jer ćemo vrlo lako moći reći je li ona istinita ili lažna.

Boolova algebra nam omogućuje da kombiniramo više takvih izjava ili konkretno u našim problemima više uvjeta kako bismo dobili točnu informaciju.

Logička operacija negacije, NE, NOT	
Izjava	Prikaz logičke operacije negacije
A	$\bar{A}$
0	1
1	0

Vrlo je jednostavno. Izjavimo nešto što je istinito, a logička operacija negacije to pretvori u laž i obrnuto.

Logička operacija konjunkcije, I, AND		
Izjava 1	Izjava 2	Prikaz logičke operacije konjunkcije
A	B	A · B
0	0	0
1	0	0
0	1	0
1	1	1

Kada pričamo o logičkoj operaciji konjunkcije odnosno AND, najvažnije je da, ako imamo dvije logičke izjave, budu obje istinite kako bi i njezin rezultat bio istinit. Npr. Ivan i Miroslav trče štafetu. Ako nijedan ne istrči svoj krug, jasno je da neće završiti utrku. Čak i ako bilo koji od njih otrči jedan krug, a drugi ne, to isto znači da neće završiti utrku. Tek ako obojica odrade svoj dio, onda se može smatrati da su završili štafetnu utrku.

Logička operacija disjunkcije, ILI, OR		
Izjava 1	Izjava 2	Prikaz logičke operacije disjunkcije
A	B	A + B
0	0	0
1	0	1
0	1	1
1	1	1

Kod logičke operacije disjunkcije konačan je rezultat lažan samo ako su obje izjave lažne. Ako je bilo koja izjava istinita (ili obje), istinit je i konačan rezultat.

Npr. Miroslav i Ivan pokušavaju prokrijumčariti kokice u kino. Ako nijedan ne prokrijumčari kokice, neće imati što grickati dok gledaju film. Ako barem jedan od njih uspije u svojem naumu, imat će kokice u kinu. Ako obojica uspiju, definitivno će imati puno toga za grickati.

Vježba 40. – Napiši program koji će ti dopustiti da pritisneš ili ne pritisneš gumbić B u razdoblju od pet sekundi. Ako je gumbić B u bilo kojem trenutku bio pritisnut, `while` petlja će prekinuti svoje izvođenje nakon odrađenog ciklusa od pet sekundi koliko traje ispitivanje uvjeta i ispisat će ugrađenu sliku NO. Ako gumbić B nije bio pritisnut, na zaslonu će biti prikazana ugrađena slika YES.

Rješenje:

```
from microbit import *
while not button_b.was_pressed():
    display.show (Image.YES)
    sleep (5000)
display.show (Image.NO)
```

U petlju stavljamo uvjet koji provjerava je li gumbić B u nekom trenutku bio pritisnut (`button_b.was_pressed()`). Ako je gumbić B bio pritisnut, uvjet je istinit i u tom će se slučaju `while` petlja vrtiti i dalje. S obzirom da nama treba suprotno, upotrijebit ćemo negaciju (not) tog uvjeta (`not button_b.was_pressed()`) kako bi rješenje bilo u skladu s postavljenim zadatkom.

Vježba 41. – Napiši program koji će prekinuti izvođenje `while` petlje ako je dodirnut ili pin 0 ili pin 2. U svim će ostalim slučajevima program prikazivati ugrađenu sliku SILLY. Kada se prekine izvođenje programa, prikazat će se ugrađena slika SURPRISED.

Rješenje:

```
from microbit import *
while not (pin0.is_touched () or pin2.is_touched()):
    display.show (Image.SILLY)
display.show (Image.SURPRISED)
```

Vježbu ćemo riješiti uz pomoć logičke operacije or odnosno disjunkcije (`pin0.is_touched () or pin2.is_touched()`). Dok je bilo koji pin (0 ili 2) dodirnut, petlja se mora prekinuti. S obzirom da taj uvjet stavljamo u `while` petlju, a ona se izvršava kada je uvjet ispunjen, moramo staviti negaciju ispred uvjeta (`not (pin0.is_touched () or pin2.is_touched())`) i rješenje će biti ispravno. Obratimo pozornost na zagrade. OR logička operacija omeđena je zagradama, a ispred njih je negacija. Ako ne bi bilo zagrade, negacija bi se primjenjivala samo na prvi dio disjunkcije (`not pin0.is_touched ()`).

Vježba 42. – Napiši program koji će svake tri sekunde provjeravati jesu li pritisnuti gumbići A i B. Ako nisu pritisnuti, `while` petlja će se i dalje izvršavati i prikazivati ugrađenu sliku PACMAN, a kada se prekine njezino izvođenje, prikazat će se ugrađena slika TARGET.

Rješenje:

```
from microbit import *
while not (button_a.is_pressed () and button_b.is_pressed()):
    display.show (Image.PACMAN)
    sleep (3000)
display.show (Image.TARGET)
```

Slično je kao i s prošlim zadatkom. Prije cijelog uvjeta moramo staviti negaciju jer se `while` petlja izvršava ako je uvjet istinit, a prekida ako je neistinit. Naš je uvjet (`button_a.is_pressed () and button_b.is_pressed()`) istinit prema pravilima konjunkcije kada su oba gumbića u trenutku provjere istodobno pritisnuta. Moramo paziti na zagrade i funkciju kojom se koristimo – trenutačno stanje gumbića, a ne je li nekad prije bio pritisnut.

Zadatci za ponavljanje

1. Napiši program koji će provjeravati svakih pet sekundi je li u bilo kojem trenutku bio dodirnut pin 2. Ako je bio dodirnut, onda se `while` petlja prekida i prikaže se poruka „Pazljivije", a ako ništa nije dodirnuto, na zaslonu se prikaže ugrađena slika YES.
2. Napiši program koji će svakih sedam sekundi provjeravati stanje gumbića A i B. Ako je bilo koji od njih pritisnut u trenutku provjere, prekida se izvršavanje programa s ispisom ugrađene slike SAD. Kada se program izvršava, na zaslonu se pojavljuje slika ROLLERSKATE svakih 200 milisekundi.
3. Napiši program koji će svakih devet sekundi provjeravati je li bio dodirnut neki pin (0 i 1). Ako su oba bila dodirnuta, program se prestaje izvoditi uz ispis ugrađene slike NO. Ako to nije slučaj, program se nastavlja izvršavati uz izmjenjivanje ugrađenih slika DIAMOND i DIAMOND_SMALL svaku desetinku sekunde.
4. Napiši program koji će svake dvije sekunde provjeravati je li na micro:bitu bila primijenjena gesta „shake" i to s pritisnutim gumbićem B. Ako se oboje od navedenog dogodilo u trenutku provjere, program se zaustavlja i ispisuje poruku: „potres". Ako to nije slučaj, na zaslonu se prikaže ugrađena slika UMBRELLA.

14. ANIMACIJE

Kada se nešto kreće i kada je dinamično, uvijek je bolji efekt odnosno izgled nečega. S obzirom da micro:bit na svojemu zaslonu prikazuje slike/tekst, sadržaj će biti zanimljiviji ako ne bude statičan.
Jedan je od načina prikazivanja sadržaja na zaslonu „scroll" metodom. Drugi je način da mi sami kreiramo nekoliko slika pa ih prikazujemo s vremenskim odmakom i tako dočaramo animaciju.

Vježba 43. – Napiši program koji će prikazati animirani sat koji pokazuje sate od 0 ili 12 pa do 11. Samo se pomiče mala kazaljka.

Rješenje:

```
from microbit import *
display.show (Image.ALL_CLOCKS)
```

MicroPython ima puno ugrađenih slika, a neke od njih su i grupirane u liste (ALL_CLOCKS, ALL_ARROWS). Ako je bilo što grupirano kao lista, možemo to lako prikazati na zaslonu, što se vidi iz rješenja vježbe 43.

Ovako riješena vježba prikazat će animaciju odnosno sve slike iz liste samo jednom i to s vremenskim odmakom od nekih pola sekunde. Ponavljanje (jednom ili željeni broj puta) i brzinu izmjenjivanja slika možemo mijenjati.

Rješenje vježbe 43. s dodatnim atributima:

```
from microbit import *
display.show (Image.ALL_CLOCKS, loop = True, delay = 1000)
```

Dodatni atributi mogu kontrolirati količinu ponavljanja (`loop`) i koliki će biti razmak (`delay`) između prikazivanja slika. Ponavljanje možemo kontrolirati uz pomoć različitih petlji (`FOR` ili `while`), a možemo i dodati atribut koji će reći da se na kraju animacije izbriše zaslon (`clear = True`) ili da se sve prekida onoliko dugo koliko traje animacija (`wait = True`).

„Scroll" metoda ima atribute kojima možemo kontrolirati brzinu (`delay`), jedno ponavljanje ili više ponavljanja (`loop`), blokiranje ostalih radnji dok traje animacija (`wait`) i postavljanje širine znakova (`monospace = True` ako je ta mogućnost uključena, svako će slovo imati širinu od pet stupaca i razmak od dva stupca).

Npr.:

```
display.scroll ("hrast kitnjak", loop = True, delay = 100, monospace = True, wait
= True)
```

U ovome smo slučaju uključili beskonačnu petlju (`loop = True`), slova se prikazuju brzinom od 0,1 sekunde (`delay = 100`), širina znakova je svugdje ista (`monospace = True`) i dok traje animacija/„scrollanje", sve ostalo je na čekanju (`wait = True`).

Vježba 44. – Napravi niz od nekoliko slika koje će simulirati brisač stakla i u jednom i u drugom smjeru. Ne prestaje nikad brisati.

Rješenje:

```
from microbit import *
brisac1 = Image ("00000:00000:00000:00000:99900")
```

```
brisac2 = Image ("00000:00000:90000:09000:00900")
brisac3 = Image ("00000:00000:00900:00900:00900")
brisac4 = Image ("00000:00000:00009:00090:00900")
brisac5 = Image ("00000:00000:00000:00000:00999")
brisac = [brisac1, brisac2, brisac3, brisac4, brisac5]
display.show (brisac, delay = 200)
```

Kreirali smo slike koje prikazuju određeni položaj brisača. Te slike spremimo u listu (`brisac = [brisac1, brisac2, brisac3, brisac4, brisac5]`) – to u programu označavamo uz pomoć uglatih zagrada. Nakon toga samo pokrenemo prikaz članova liste. U našem se slučaju članovi izmjenjuju brzinom 200 milisekundi, a ako će „kiša jače padati", povećat ćemo brzinu „brisača". Još samo moramo dodati animaciju u drugom smjeru, a nakon toga slijedi beskonačno ponavljanje.

Konačno rješenje vježbe 44.:

```
from microbit import *
brisac1 = Image ("00000:00000:00000:00000:99900")
brisac2 = Image ("00000:00000:90000:09000:00900")
brisac3 = Image ("00000:00000:00900:00900:00900")
brisac4 = Image ("00000:00000:00009:00090:00900")
brisac5 = Image ("00000:00000:00000:00000:00999")
brisac = [brisac1, brisac2, brisac3, brisac4, brisac5, brisac4, brisac3, brisac2]
display.show (brisac, delay = 200, loop = True)
```

Listi dodajemo one dijelove animacije koji nedostaju (`brisac4, brisac3, brisac2`) i podešavamo prikazivanje slika kao beskonačno (`loop = True`).

Vježba 45. – Napiši program koji će prikazati animaciju utapanja patke. Koristi se ugrađenom slikom DUCK. Neka se utapa brzinom od pola sekunde po redu piksela.

Rješenje:

```
from microbit import *
patka = Image.DUCK
for i in range (1, 7):
    display.show(patka)
    patka = patka.shift_down(1)
    sleep(500)
```

U MicroPythonu postoje zanimljive funkcije koje omogućuju pomicanje slike za određeni broj redaka gore (`shift_up(n)`), dolje (`shift_down(n)`), lijevo (`shift_left(n)`) i desno (`shift_right(n)`). Njihovim korištenjem vrlo lako možemo animirati/pokretati neku sliku u svim smjerovima bez da crtamo svaki njezin položaj.

S obzirom da ugrađenu sliku ne možemo izravno pomicati uz pomoć prethodno opisanih funkcija, spremili smo ju u varijablu `patka` (`patka = Image.DUCK`). Da bismo simulirali utapanje, koristimo se funkcijom `shift_down(1)` i to za jedan redak odnosno piksel. Unutar FOR petlje koja se ponavlja šest puta (prvi put da se prikaže slika patke, a ostalih pet da potpuno nestane sa zaslona micro:bita) stavili smo raspon od 1 do 7 (završetak raspona ne ulazi u raspon). Još moramo odrediti brzinu utapanja od, kako je zadano, pola sekunde (`sleep(500)`).

Vježba 46. – Napiši program koji će prikazati animaciju hokejaškog paka koji se kreće od donjeg desnog do gornjeg lijevog ugla (i nestane u tom uglu). Pak će biti predstavljen kvadratom od 4 piksela. Neka se pak kreće brzinom od 300 milisekundi.

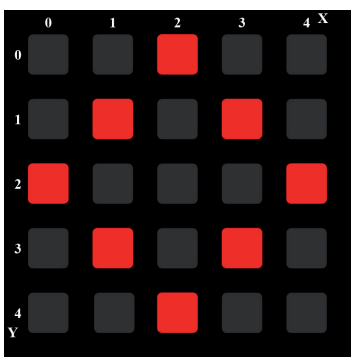
Rješenje:

```
from microbit import *
pak = Image ("00000:00000:00000:00099:00099")
for i in range (1, 7):
    display.show(pak)
    pak = pak.shift_up(1)
    pak = pak.shift_left (1)
    sleep(300)
```

U ovoj smo vježbi kreirali sliku hokejaškog paka (`pak = Image ("00000:00000:00000:00099:00099")`) te je u jednom ponavljanju FOR petlje pomican i prema gore i prema lijevo za 1 piksel/redak.

Zadatci za ponavljanje

1. Napiši program koji će prikazati sve smjerove strelica 12 puta. Neka se slike izmjenjuju dinamikom od 0,3 sekunde.
2. Napiši program koji će stvoriti slike koje simuliraju kretanje brisača, ali brisača koji je učvršćen u gornjemu desnom uglu. Gornji desni ugao bit će točka oko koje se brisač kreće u beskonačnost. Slike će se izmjenjivati brzinom od 300 milisekundi.
3. Napiši program koji će prikazati animaciju polijetanja leptira. Koristi se ugrađenom slikom BUTTERFLY. Neka polijeće brzinom od 0,7 sekundi po redu piksela i neka leti izravno prema gore.
4. Napiši program koji će pomicati jednu točku po zaslonu micro:bita prema putanji koja je nacrtana na slici 30. Svakih 600 milisekundi točka će se pomaknuti za jedno mjesto i to će napraviti točno 10 puta.



Slika 30. – putanja točke (u svakom trenutku bit će vidljiva samo jedna točka)

15. REPRODUKCIJA GOVORA

Znamo da se spajanjem zvučnika micro:bit pretvara u DJ-a koji zna puštati glazbu i raditi zvučne efekte. Je li to sve što može? Naravno da nije. Hoćete razgovarati s njim? Može. Hoćete da pjeva? Može i to.

Vježba 47. – Napiši program koji će omogućiti da micro:bit ispriča u kojem programskom jeziku radiš.

Rješenje:

```
from microbit import *
import speech
speech.say ("MicroPython")
```

Sve što nam treba je modul za govor (`import speech`), malo logike i poznavanje engleskog jezika. Lako je pogoditi naredbu i bez poznavanja engleskog jezika. Govori na način da pričaš (`speech.say`), a znakove koje želimo da izgovoriš pronađi unutar dvostrukih navodnika. Hrvatski jezik mu baš i ne ide, ali svakako ćemo se dobro zabaviti isprobavajući.

Ovo nije sve što možemo očekivati od pričanja micro:bita. Postoji još puno mogućnosti odnosno parametara koje možemo podesiti.

`pitch` – koliko visoko odnosno nisko zvuči govor (0 = visoko, 255 = nisko)
`speed` – koliko brzo želimo da micro:bit priča (0 = brzo, 255 = sporo)
`mouth` – način artikuliranja govora (0 = slično trbuhozborčevoj lutki, 255 = zvuk brodske trube)
`throat` – opuštenost grla (0 = zategnuto, 255 = opušteno)

Najbolje je isprobati sve.

Vježba 48. – Napiši program koji će sporim dubokim glasom putem micro:bita progovoriti poznate riječi iz filma *Live and let die!*.

Rješenje:

```
from microbit import *
import speech
speech.say ("Live and let die", speed = 240, pitch = 240)
```

U ovom smo slučaju dodali atribute koji nam omogućuju usporavanje govora (`speed`) i dublji ton (`pitch`). Kada čujemo kako to zvuči, možda ćemo biti malo razočarani jer to ipak neće biti slično Elvisu Presleyju. Možemo čuti razliku u brzini i tonu. Isto tako, razliku će napraviti i kvaliteta zvučnika na koji ćemo priključiti našeg govorećeg robota.

Vježba 49. – Napiši program koji će vikati *Upomoooooć!*.

U ovom ćemo se slučaju morati malo prisjetiti koja slova u engleskoj abecedi pripadaju kojem izgovoru i malo to prilagoditi. Prilagodba podrazumijeva slova/slogove koji moraju trajati nešto dulje. Naučit ćemo kako natjerati micro:bit da nešto izgovori na način kako mi to želimo.

Slova/slovo koje upisujemo	Slovo koje će se izgovoriti
AE	A
B	B
CH	Č
DH	D
EH	E
F	F
G	G
/H	H
IH	I
J	J
C	K
L	L
M	M
N	N
AO	O
P	P
R	R
S	S
TH	T
SH	Š
UH	U
V	V
ZH	Ž

U engleskom jeziku ima još puno zvukova odnosno kombinacija slova koja ne zvuče kao nešto što možemo reprezentirati hrvatskim slovima. Sve ih možete pronaći na raznim linkovima (npr. http://microbit-micropython.readthedocs.io/en/latest/speech.html).

Rješenje:

```
from microbit import *
import speech
speech.pronounce("UH5P5OH5OH5M5OH5OH5OH5CH5")
```

Koristimo se metodom govora `pronounce` (izgovor) te unutar okruglih zagrada pišemo tekst koji želimo u obliku koji želimo. Prvo napišemo glas (u obliku slova iz prethodne tablice) koji želimo čuti od micro:bita, a nakon toga naglasak na tom glasu (kontroliramo ga brojevima od 1 do 8). Ako želimo da neki glas dulje traje, jednostavno ga upišemo ponovno (koliko puta nam treba) - `OH5OH5` (dva će se puta izgovoriti glas o).

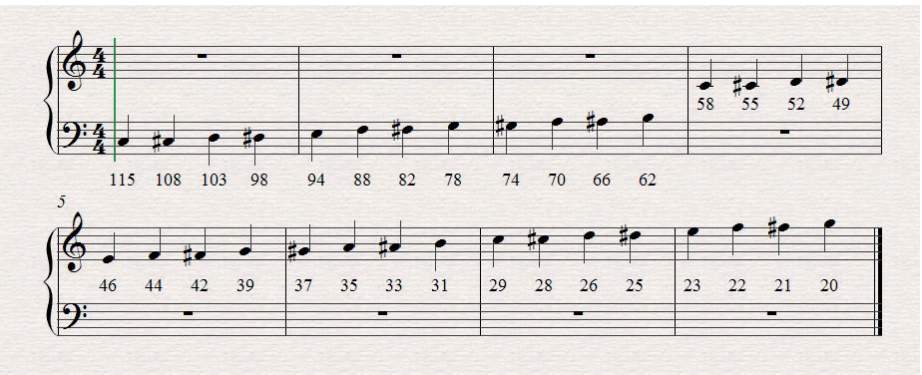
Vježba 50. – Napiši program koji će otpjevati poznati stih *We are the Champions.*

Rješenje:

```
import speech
from microbit import sleep
watc = [
    "#94VEHEH",
    "#88AEREH",
    "#78THEH",
    "#70CHAEM",
    "#70PIHAONS",
]
pjesma = ''.join(watc)
speech.sing(pjesma, speed=100)
```

Brojka na početku označava visinu tona, a nakon visine tona (`pitch`) slijede fonemi/glasovi koje želimo da micro:bit otpjeva.

Tonove zajedno s glasovima spremimo u listu (u našem slučaju `watc`) te ih spojimo u jednu cjelinu (`pjesma = ''.join(watc)`). Nakon toga je micro:bit može otpjevati metodom `sing` (`speech.sing(pjesma, speed=100)`) dodajući joj željene atribute.



Slika 31. – Prikaz visina tonova s pripadajućim pitchom (brojkom) u MicroPythonu

Zadatci za ponavljanje

1. Napiši program koji će „natjerati" micro:bit da kaže naziv filma koji smo posljednji pogledali u kinu i to jedan program koji će naziv izreći na engleskom jeziku, a jedan koji će ga izreći na hrvatskome jeziku.
2. Napiši program koji će ti visokim glasom i brzim tempom izreći riječi: *Everybody get down this is a robbery!*. Pronađi i značenje tih riječi.
3. Napiši program koji će na određeni specifičan način izreći stih: *Are you lonesome tonight.*
4. Napiši program kojim će micro:bit propjevati stari hit: *We don't need no education.*

16. TEMPERATURA I KOMPAS

Micro:bit je jedan od primjera kako i koliko programiranje može pomoći u rješavanju svakodnevnih problema. Micro:bit može izmjeriti i temperaturu. To nije mjerenje vanjske temperature (jer nema senzor), već je to temperatura matične ploče. Nećemo moći reći da je to 100 % precizno jer micro:bit pokazuje temperaturu koja je (otprilike) 3 stupnja Celzijusa viša od temperature mjesta na kojemu se nalazimo. Ako dovoljno puta izmjerimo temperaturu i usporedimo je sa stvarnom temperaturom, oduzimanjem ćemo doći do točne temperature.

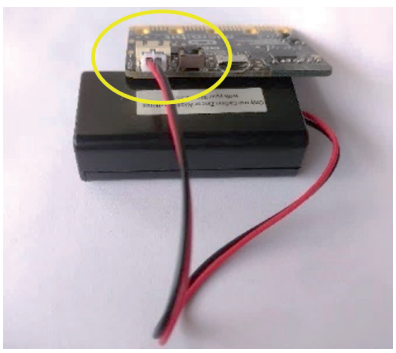
Vježba 51. – Napiši program koji će izmjeriti trenutačnu temperaturu micro:bita.

Rješenje:

```
from microbit import *
display.scroll(str(temperature()))
```

Funkcija `temperature()` izbacuje temperaturu našeg robota, a funkcijom `str()` pretvaramo je u vrijednost koju je moguće ispisati na njegovu zaslonu.

U ovom bi slučaju bilo dobro da micro:bit ima mogućnost pomicanja i da ne bude stalno uključen u računalo nego da ga možemo slobodno odnijeti gdjegod i kakogod nam odgovara. Tomu služi zadnji dio paketa (spremnik za baterije i baterije) o kojem još nismo pričali.

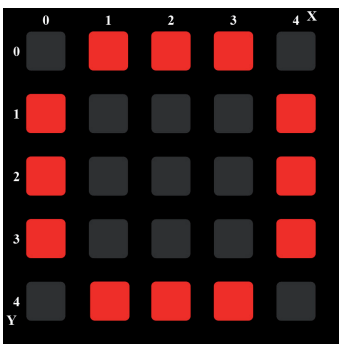


Slika 32. – spajanje spremnika s baterijama s micro:bitom (mjesto spajanja označeno žuto)

Vježba 52. – Napiši program koji će pokazati znak S (Sjever) u trenutku kada micro:bit bude usmjeren (gornji dio – „oči") prema sjeveru.

Kako bismo to mogli napraviti, moramo se upoznati s mogućnostima vezanima uz kompas koje nudi micro:bit.

Prije nego što se uopće krenemo koristiti funkcijama vezanim uz kompas, moramo provesti kalibraciju. Ona se provodi uz pomoć igrice u kojoj okretanjem micro:bita moramo nacrtati krug (slika 33).



Slika 33. – Krug koji je potrebno nacrtati kako bi se micro:bit kalibrirao

Kalibracija se pokreće uz pomoć naredbe `compass.calibrate()`. Nakon toga micro:bit najavljuje što treba napraviti: `DRAW A CIRCLE`. Nakon što se micro:bit ispravno kalibrira, na zaslonu se pojavljuje slika nasmiješenog lica.

Za pokazivanje smjera sjevera MicroPython se koristi objektom `compass` i metodom `heading`. To znači da će naredba `compass.heading ()` ispisati jedan broj između 0 i 360 koji označava za koliko je gornji dio micro:bita (njegove „oči") zakrenut od smjera sjevera u smjeru kazaljke na satu. Npr. ako nam naredba `compas.heading ()` daje vrijednost 90, to znači da je micro:bit usmjeren točno prema istoku.

Rješenje:

```
from microbit import *
compass.calibrate()
while True:
    sleep (100)
    if compass.heading()=0:
         display.show ("S")
         sleep (1000)
         display.clear()
```

Neizostavna komponenta svakog programa koji uključuje kompas, pa tako i našeg programa, jest kalibracija (`compass.calibrate()`). Nakon toga slijedi ono što vježba traži. Trebamo si dati vremena za zakretanje micro:bita (`sleep (100)`) i onda provjeriti usmjerenje (`if compass.heading()=0`). Zadali smo si skoro pa nemoguću misiju. Bit će jako teško točno pogoditi sjever odnosno ispis od 0 stupnjeva. Zbog toga ćemo malo prilagoditi vježbu i ostavit ćemo prostora za odstupanje i to 20 stupnjeva u jednom i drugom smjeru. Ostatak je programa jasan. Ako je uvjet ispunjen, prikazat će se slovo S (`display.show ("S")`) i ostat će prikazano jednu sekundu (`sleep (1000)`) te će se zaslon obrisati (`display.clear()`).

Prilagođeno rješenje:

```
from microbit import *
compass.calibrate()
while True:
    sleep (100)
    a= compass.heading()
    if a>340 or a<20:
        display.show ("S")
        sleep (1000)
        display.clear()
```

Ovo rješenje neće biti 100 % precizno, ali ćemo barem imati program odnosno kompas koji pokazuje vrijednosti, što bismo teško mogli utvrditi za prethodni program. Što smo dodali? Istinitost uvjeta ne temelji se samo na jednoj brojci nego na rasponu brojeva (`a>340 or a<20`). Zašto su vrijednosti naredbe veće od 340 ili manje od 20? Prilagodili smo se tomu da vrijednosti naredbe `compass.heading()` mogu biti od 0 do 360. Ako bi kompas bio usmjeren do 20 stupnjeva prema istoku, vrijednost bi bila manja od 20, a u suprotnom bi vrijednosti bile između 340 i 360. Brojku koja se odnosi na usmjerenje kompasa spremili smo u varijablu kako bismo si olakšali posao odnosno skratili kôd (`a= compass.heading()`).

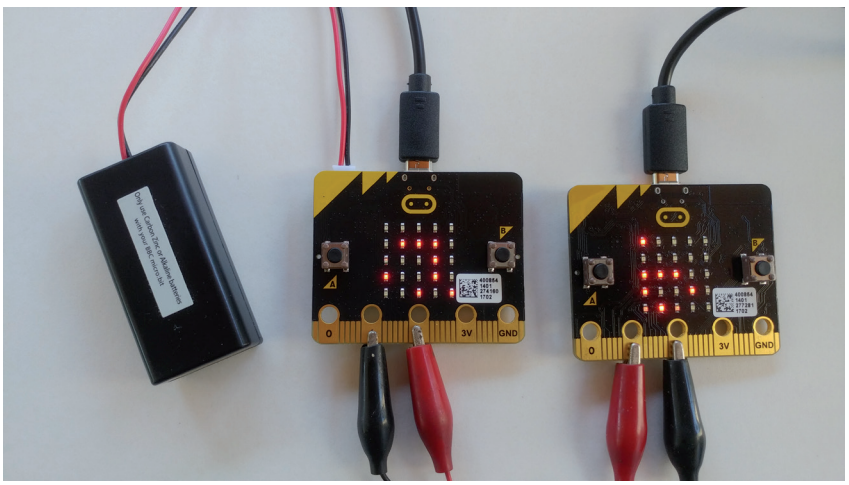
Zadatci za ponavljanje

1. Napiši program koji će svake minute mjeriti temperaturu micro:bita. Nakon svakog mjerenja trebaš držati micro:bit u ruci (povećavajući mu temperaturu) te provjeriti povisuje li se mjerena temperatura.
2. Napiši program koji će ovisno o temperaturi prikazivati različitu sliku na zaslonu micro:bita. Ako je temperatura do 20 stupnjeva Celzijusa, sve će LED-ice svijetliti intenzitetom 1 (najslabijim, jedva vidljivim intenzitetom). Ako je temperatura između 20 i 30 stupnjeva Celzijusa, LED-ice će svijetliti intenzitetom 5, a ako je iznad 30 stupnjeva Celzijusa, svijetlit će najjačim intenzitetom (9).
3. Napiši program koji će pretvoriti micro:bit u potpuni kompas. Vježbom 52. micro:bit pokazuje samo sjever. Dopuni ga tako da prikazuje i ostale strane svijeta i to slovima I, Z te J. Koristi se odstupanjem od točnog usmjerenja za 20 stupnjeva Celzijusa.
4. Napiši program koji će omogućiti da micro:bit na svojem zaslonu ispiše točno odstupanje od usmjerenja sjevera. Stani u sredinu prostorije u kojoj se nalaziš te pronađi za svaki njezin kut koliko je odstupanje od usmjerenja sjevera.

17. DVA MICRO:BITA U ŽIČANOJ MREŽI

Spajanje dvaju micro:bitova u žičanu mrežu može razjasniti na koji način mreže općenito funkcioniraju.
Mreže se sastoje od nekoliko slojeva. Svaki sljedeći sloj ovisi o prethodnom. Nećemo se baviti sa svim slojevima modernih mreža nego samo s onim aspektima koji su nam potrebni da bismo razumjeli funkcioniranje micro:bitova.

Da bi nešto bilo umreženo, treba biti i fizički spojeno. Umrežavanje se može ostvariti i bežično, ali mi ćemo micro:bitove spajati žicama. To će biti naš prvi sloj u umrežavanju – fizički sloj.



Slika 34. – spajanje dvaju micro:bitova žicama

Obratimo pozornost na pinove za koje su zakvačene žice. Pin 1 lijevog micro:bita (crni kabel) spojen je na pin 2 desnog micro:bita i obrnuto (crveni kabel). Ne moraju biti točno tako spojeni, ali važno je da pin jednog micro:bita ne bude spojen na isti pin drugog micro:bita.

Načelo je jednostavno. Razmotrimo malo način komunikacije mobitelom. Mogli bismo reći da je mikrofon jednog mobitela spojen sa zvučnikom drugog mobitela i obrnuto. Ono što jedna strana govori u mikrofon, to druga strana čuje putem zvučnika. Kada bismo se drukčije spojili, bilo bi svega samo ne komunikacije.

Sljedeća razina mreže koja nam je potrebna jest razina signala. Uvijek ovisi o kvaliteti veze – kod nas neće biti problema jer su micro:bitovi blizu i fizički su spojeni žicama.

Signal se šalje putem objekta `pin` i metodom `write_digital`. U praksi bi to izgledalo ovako:

OUTPUT/IZLAZ

`pin0.write_digital(1)` – šalje se signal na pin 0
`pin0.write_digital(0)` – nema signala na pin 0
INPUT/ULAZ
`pin0.read_digital()` – čita signal i može poprimiti vrijednosti 1 i 0.

Nova je potrebna razina komunikacije protokol. Uz pomoć protokola primaju se i šalju informacije. Primjeri protokola su http („hypertext transfer protocol") koji upotrebljava www, ftp („file transfer protocol") koji je korišten za prijenos datoteka na mreži. To je način na koji se informacije pretvaraju/kodiraju kako bi bile prikladne za prijenos i čitanje/dekodiranje. Mi ćemo se ovdje poslužiti mogućnostima i različitim stanjima gumbića A i B.

Vježba 53. – Napiši program koji će pritiskom gumbića A na jednom micro:bitu prikazati kvadrat (ugrađenu sliku SQUARE). Ako nije pritisnut gumbić A, na zaslonu će se vidjeti mali kvadrat (ugrađena slika SQUARE_SMALL).

Rješenje:

```
from microbit import *
while True:
    sleep (100)
    if button_a.is_pressed():
        pin1.write_digital(1)
    else:
        pin1.write_digital(0)
    if pin2.read_digital()==1:
        display.show(Image.SQUARE)
        sleep (200)
    else:
        display.show(Image.SQUARE_SMALL)
        sleep (200)
```

Pin 1 je na obama micro:bitovima „OUTPUT"/IZLAZ. Vrijednost pina 1 mijenja se ovisno o tome je li ili nije pritisnut gumbić A. Ako je gumbić A pritisnut, postoji izlazni signal na pin 1 (`if button_a.is_pressed(): pin1.write_digital(1)`), a ako nije pritisnut, izlazni je signal nula (`else: pin1.write_digital(0)`).

Ulaz/„INPUT" je pin 2 i iz njega se moraju čitati vrijednosti (`pin2.read_digital()==1`). Na temelju tih učitanih vrijednosti izvršavaju se odgovarajuće radnje odnosno ispisuje se veći ili manji kvadrat na zaslonu micro:bita. Da slika ne bi prebrzo nestala, nakon svakog ispisa stavljamo zadršku od dvije desetinke sekunde (`sleep (200)`). To se beskonačno ponavlja (`while True:`).

Napomena: Moguće je da komunikacija neće funkcionirati u obama smjerovima ako oba micro:bita nisu na istoj vrsti napajanja. Oba moraju biti na baterijama ili na USB napajanju.

Zadatci za ponavljanje

1. Napiši program koji će postići komunikaciju između dvaju micro:bitova putem pinova 0 i 2. Ako je pritisnut gumbić B, pojavit će se veliki dijamant (Image.DIAMOND) na zaslonu, a ako nije pritisnut, na zaslonu će biti vidljiva ugrađena slika malog dijamanta (Image.DIAMOND_SMALL). Neka se svaka slika zadrži na zaslonu barem pola sekunde.
2. Napiši program koji će postići komunikaciju između dvaju micro:bitova putem pinova 0 i 1. Ako je pritisnut gumbić A, ispisat će se veliko slovo A, a ako je pritisnut gumbić gumbić B, ispisat će se veliko slovo B. Neka se slova na zaslonu zadrže barem sekundu.
3. Napiši program koji će postići komunikaciju između dvaju micro:bitova putem pinova 1 i 2. Program će čekati 10 sekundi i u tom će vremenu na zaslonu biti prikazana kvačica (ugrađena slika Image.YES). Za to vrijeme potrebno je pritiskati gumbić A. Ako je gumbić A pritisnut više od 10 puta, na zaslonu se pojavljuje slika matematičkog operatora za zbrajanje (+). Ako je gumbić A pritisnut manje od 10 puta, na zaslonu se prikazuje slika matematičkog operatora za oduzimanje (-). Neka od tih slika bit će prikazana dvije sekunde na zaslonu i nakon toga će program krenuti ispočetka.

18. POTPROGRAMI

Gdje god učite programiranje, naići ćete na potprograme. Čemu to uopće? Problem ćete uspješno riješiti ako ga pojednostavnite i razbijete na manje funkcionalne dijelove. To ćemo primijeniti u MicroPythonu.

Vježba 54. – Napiši program koji će imati dva potprograma – zbrajanje i oduzimanje. U potprogramu zbrajanje potrebno je zbrojiti brojeve 321 i 123 te ispisati rezultat, a u potprogramu oduzimanje potrebno je oduzeti te brojeve i ispisati rezultat. U glavnom programu potrebno je pozvati svaki od potprograma te prije svakog od njih ispisati njegov naziv i rješenje metodom „scroll".

Rješenje:

```
from microbit import *
def zbrajanje():
    zbroj=321+123
    display.scroll (str(zbroj))
def oduzimanje():
    razlika=321-123
    display.scroll (str(razlika))
display.scroll("Rezultat zbrajanja je:")
zbrajanje()
display.scroll("Rezultat oduzimanja je:")
oduzimanje()
```

Žutom bojom označen je potprogram zbrajanje. Naredba za početak potprograma jest `def`, a naziv potprograma popraćen je otvorenom i zatvorenom okruglom zagradom. Nakon toga je stavljeno dvotočje i tipkom Enter prelazimo u novi red koji je malo uvučen udesno. Tu pišemo naredbe potprograma, a kada smo gotovi s naredbama, prelazimo u novi red i maknemo uvlaku. Tada završava definiranje potprograma.

Pozivanje potprograma je vrlo jednostavno (označeno crvenom bojom) – na željenom dijelu kôda upišemo njegov naziv te otvorenu i zatvorenu okruglu zagradu (`zbrajanje()`).

Međutim, vidimo da ovakav način rada s potprogramima nema smisla. Umjesto da pomaže i olakšava rad, zapravo ga otežava i povećava broj linija kôda. Potprogram se najbolje primjenjuje kada ima nekakvu ulaznu varijablu i kada na temelju nje izračuna nekakvu izlaznu vrijednost.

Vježba 55. – Napiši program koji će imati potprogram duplo. U potprogramu duplo potrebno je udvostručiti vrijednost ulazne varijable te ispisati rezultat. U glavnom je programu potrebno odabrati ulaznu vrijednost za potprogram. Ulazna će vrijednost biti slučajni broj između 1 i 100.

Rješenje:

```
from microbit import *
import random
```

```
n = random.randint (1,100)
def duplo(n):
    duplo=n+n
    display.scroll (str(duplo))
display.scroll("Dvostruka vrijednost broja ")
display.scroll(str (n))
display.scroll(" je ")
duplo(n)
```

Okrugle zagrade nakon naziva potprograma nisu samo ukras nego imaju svrhu. U njih se upisuje varijabla na temelju koje se vrši izračun unutar potprograma. U potprogramu može biti i više od jedne varijable (pobroji ih se jer su odvojene zarezom).

Kao što je zadano u vježbi, prvo slučajnim odabirom upisujemo vrijednost od 1 do 100 u varijablu `n` (`n = random.randint (1,100)`). Nakon toga se tom varijablom koristimo u potprogramu kako bi se dobila njezina dvostruka vrijednost. Potprogram s ulaznom varijablom poziva se tako da se napiše njegov naziv i odgovarajuće varijable unutar okruglih zagrada. Ostatak programa napisan je kako bi sveukupno izgledalo ljepše i razumljivije.

Vježba 56. – Napiši program koji će od tebe tražiti da pritiskanjem gumbića A upišeš ulaznu vrijednost. Daj si vremena osam sekundi i za to će vrijeme na zaslonu biti prikazana ugrađena slika YES. Ulazna vrijednost bit će broj čije višekratnike (zaključno s brojem 50) treba ispisati na zaslonu micro:bita.

Rješenje:

```
from microbit import *
def gumbicA():
    display.show(Image.YES)
    sleep (8000)
    x = button_a.get_presses()
def visekrat(x):
    for i in range (x, 51, x):
        display.scroll (str (i))
        sleep (200)
gumbicA()
visekrat(x)
```

Imamo prvi potprogram (`gumbicA()`) koji unutar osam sekundi bilježi koliko je puta pritisnut gumbić A i drugi potprogram (`visekrat(x)`) koji ispisuje željene višekratnike. Međutim, ako pokrenemo ovaj program, micro:bit će nam javiti grešku u zadnjoj liniji (kod pozivanja potprograma `visekrat(x)`) – nije definiran x.

To će se dogoditi zato što potprogram funkcionira kao jedna zasebna cjelina i ako to ne zatražimo, varijable kojima se on koristi su njegove. Ako u glavnom programu imamo varijablu, njom se možemo koristiti i u potprogramima, ali ne i obrnuto. Zbog toga varijablu, u našem slučaju varijablu `x`, treba proglasiti globalnom. Tada se i glavni program i svi potprogrami mogu njom koristiti.

Ispravno rješenje:

```
from microbit import *
def gumbicA():
    display.show(Image.YES)
    sleep (8000)
    global x
    x = button_a.get_presses()
def visekrat (x):
    for i in range (x, 51, x):
        display.scroll (str (i))
        sleep (200)
gumbicA()
visekrat (x)
```

Prije korištenja/upisivanja vrijednosti u varijablu (označeno žuto) proglasimo je globalnom odnosno dostupnom svima i problem je riješen.

Zadatci za ponavljanje

1. Napiši program koji će imati dva potprograma – mnozenje i dijeljenje. U potprogramu mnozenje potrebno je pomnožiti brojeve 112 i 16 te ispisati rezultat, a u potprogramu dijeljenje potrebno je podijeliti te brojeve i ispisati rezultat. U glavnom je programu potrebno pozvati svaki od potprograma te prije svakog od njih ispisati njegov naziv i rješenje metodom „scroll".
2. Napiši program koji će imati potprogram kvadrat. Potprogram će na temelju ulazne varijable `k` izračunati kvadrat varijable `k`. Varijabla `k` dobiva se slučajnim odabirom u rasponu brojeva 10 i 20. To će se ispisati na zaslonu micro:bita.
3. Napiši program koji će imati dva potprograma – veci i ostatak. Glavni će program generirati dva broja između 1 i 100. Prvi će potprogram provjeriti koji je od dvaju slučajnih brojeva veći. Nakon toga će drugi potprogram podijeliti veći broj s manjim brojem tako da kao rezultat ispiše ostatak njihova dijeljenja.
4. Napiši program koji će imati tri potprograma: manji, gumbiciAB i ispis. Potprogram gumbiciAB tražit će da pritiskom odgovarajućih gumbića (kada će se brojiti pritisci gumbića A, na zaslonu će biti ispisano slovo A i isto za gumbić B) unesemo ulazne vrijednosti – za svaki od gumbića imat ćemo 10 sekundi. Potprogram manji odredit će koja je vrijednost manja od dviju vrijednosti koje smo unosili. Potprogram ispis ispisat će svaki drugi broj u rasponu od manjeg do većeg (uključujući i veći) koji smo unijeli pritiskanjem gumbića.

19. DOKUMENTI

Vrlo se često nalazimo u situaciji da želimo putem programskog jezika spremati nekakve informacije u dokumente, npr. postignute rezultate igrača u igricama. U ozbiljnim programima stvaramo baze podataka o različitim svojstvima i kriterijima. Micro:bit ima mogućnost otvaranja datoteka te zapisivanja u njih. Mogućnosti su vrlo ograničene, ali ipak postoje.

Vježba 57. – Napiši program koji će kreirati dokument `vazno.txt`. U taj dokument upiši sljedeći tekst: Ovaj dokument sadrzi vazne informacije. Na kraju putem ispisa na zaslonu micro:bita provjeri sadržaj dokumenta.

Rješenje:

```
from microbit import *
import os
with open("vazno.txt", "w") as dok1:
    dok1.write("Ovaj dokument sadrzi vazne informacije.")
with open("vazno.txt") as ispis:
    sadrzaj=ispis.readline()
display.scroll(sadrzaj)
```

Prvo je potrebno pozvati modul za rad s operacijskim sustavom (`import os`). Nakon toga treba otvoriti dokument s funkcijom `open` koja omogućuje zapisivanje i čitanje dokumenta. Svaki dokument (u našem slučaju `vazno.txt`) mora se otvoriti tako da se dodijeli nekom objektu (u našem slučaju dok1). Dokument `vazno.txt` otvoren je tako da se u njega može upisivati – argument w (`with open("vazno.txt", "w")`).

Upisivanje podataka vrši se s novim objektom kojemu je pridružen dokument i metodom write (`dok1.write("Ovaj dokument sadrzi vazne informacije.")`).

Da bi se ispisao sadržaj dokumenta, potrebno ga je otvoriti i dodijeliti određenom objektu (`with open("vazno.txt") as ispis:`), ali nije potreban argument za upisivanje podataka u njega. Nakon toga se sadržaj dokumenta sprema u varijablu (u našem slučaju sadrzaj) i to kombinacijom novonastalog objekta i metode za čitanje linija (`sadrzaj=ispis.readline()`). Na kraju slijedi ispis sadržaja uz pomoć objekta `display`.

Napomena: Čitanje i upisivanje u dokument mora biti unutar naredbe `with open` (malo uvučeno) jer je dokument otvoren samo unutar te naredbe. Izlaskom iz naredbe datoteka se automatski zatvara.

Vježba 58. – Dodaj dokumentu `vazno.txt` još jednu rečenicu: Strogo povjerljivo.

Rješenje:

```
from microbit import *
import os
with open("vazno.txt", "w") as dok1:
    dok1.write("Strogo povjerljivo.")
```

```
with open("vazno.txt") as ispis:
    sadrzaj=ispis.readline()
display.scroll(sadrzaj)
```

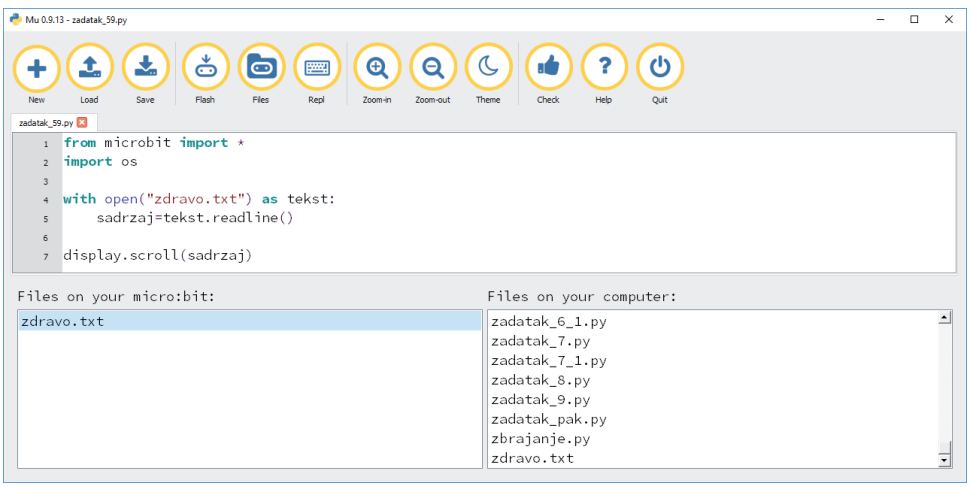
Logično je da će se ako upišemo nekakav tekst, on dodati i zalijepiti ostatku dokumenta. No, to nije slučaj s MicroPythonom. On nema mogućnost dodavanja teksta u datoteku. On jednostavno svakim upisom briše sve što je bilo u dokumentu. Python, veći brat, ima tu mogućnost jer se u ozbiljnim programima ništa ne smije izgubiti.

Nepraktično rješenje (ako želimo dodati tekst u datoteku, moramo napisati i ono što je već bilo unutra i ono što želimo dodati):

```
from microbit import *
import os
with open("vazno.txt", "w") as dok1:
    dok1.write("Ovaj dokument sadrzi vazne informacije.Strogo povjerljivo.")
with open("vazno.txt") as ispis:
    sadrzaj=ispis.readline()
display.scroll(sadrzaj)
```

Vježba 59. – Sam kreiraj tekstualnu datoteku `zdravo.txt` i unutra upiši zdrave stvari te ispiši njezin sadržaj na zaslonu micro:bita.

S obzirom da se svakim „flashanjem" brišu svi podatci s micro:bita, poslužit ćemo se praktičnim rješenjem i dodat ćemo mu datoteku putem editora. Potrebno je datoteku `zdravo.txt` prebaciti u mapu u koju mu editor sprema gotove programe (.py).



Slika 35. – izbornik Files, metodom povuci i ispusti prebacujemo datoteke s računala na micro:bit

Rješenje:

```
from microbit import *
import os
with open("zdravo.txt") as tekst:
    sadrzaj=tekst.readline()
display.scroll(sadrzaj)
```

Tek kada datoteka `zdravo.txt` bude na micro:bitu, program će proraditi. Dotad će javljati pogrešku da ne može pronaći taj dokument. Ostalo smo već u prethodnim zadatcima objasnili.

Za ispis datoteka u datotečnom sustavu koristimo se sljedećim oblikom naredbe: `sustav = os.listdir()`. Na taj se način u listu `sustav` upisuju sve datoteke koje se nalaze na micro:bitu.

Vrlo je jednostavno izbrisati datoteku s micro:bita: `os.remove('naziv_dokumenta.txt')`. Izrazito je važno pozvati modul os na početku programa.

Zadatci za ponavljanje

1. Napiši program koji će uz pomoć MicroPythona kreirati dokument `sreca.txt` i u njega jednom rečenicom upisati ono što za nas znači sreća.
2. Kreiraj datoteku `nezdravo.txt` i u nju upiši ključne nezdrave stvari koje nas okružuju. Nakon toga napiši program koji će tu datoteku učitati u micro:bit te ispisati njezin sadržaj.
3. Ispiši sve datoteke u datotečnom sustavu. Imaj na umu da se sadržaj sprema kao lista.
4. Obriši sve datoteke s micro:bita.

20. RADIOKOMUNIKACIJA

„Bluetooth" nije funkcionalan (zauzima više memorije nego si to MicroPython može dopustiti). MicroPython zbog toga ima radiokomunikaciju. „Bluetooth" modul upotrebljava se kao „hardware" odnosno kao, „bluetooth" radio, ali s puno jednostavnijim protokolima. Micro:bitovi moraju biti u neposrednoj blizini.

Vježba 60. – Napiši program koji će spojiti dva micro:bita radiokomunikacijom. Nakon spajanja oni će moći jedan drugomu slati ugrađenu sliku YES.

Rješenje:

```
from microbit import *
import radio
radio.on()
while True:
    if button_a.was_pressed():
        radio.send('Image.YES')
    paket = radio.receive()
    if paket == 'Image.YES':
        display.show(Image.YES)
```

Prvo moramo pozvati modul radio (`import radio`). Nakon toga moramo uključiti radio (`radio.on()`) i spremni smo za komuniciranje. No, nije baš tako jednostavno.

U ovoj inačici i jedan i drugi micro:bit imaju identične programe tako da i jedan i drugi mogu i slati i primati poruke.

Cijela komunikacijska „priča" smještena je u beskonačnu petlju (`while True`) kako bi stalno mogli osluškivati stiže li im kakva poruka i kako bi, ako to trebaju, mogli i poslati poruku. Slanje se vrši uz pomoć objekta `radio` i metode send (`radio.send('Image.YES')` – u našem slučaju šalje se ugrađena slika YES ako je pritisnut gumbić A). Za primanje je također zadužen objekt `radio`, ali u kombinaciji s metodom receive (`radio.receive()`). Sadržaj primljenog paketa sprema se u varijablu `paket` (`paket = radio.receive()`) i uspoređuje se s ugrađenom slikom YES (`if paket == 'Image.YES'`) te ako je to istinito, ta se slika i ispiše na zaslonu micro:bita.

Kada pokrenemo program na obama micro:bitovima, oni imaju prazne zaslone, a kada na njima pritisnemo gumbić A (prvo na jednome pa na drugome), na zaslonu se stalno prikazuje/vidi ugrađena slika YES. Tu točku pa nadalje više ne možemo zvati komunikacijom.

Vježba 61. – Dodaj prethodnom programu mogućnost slanja ugrađene slike NO pritiskom gumbića B.

Rješenje:

```
from microbit import *
import radio
radio.on()
while True:
```

```
    if button_a.was_pressed():
        radio.send("Image.YES")
    if button_b.was_pressed():
        radio.send("Image.NO")
    paket = radio.receive()
    if paket == "Image.YES":
        display.show(Image.YES)
    if paket == "Image.NO":
        display.show(Image.NO)
```

U ovom slučaju nije potrebno nešto posebno objašnjavati. Dodane su radnje koje prate pritiskanje gumbića B.

Vježba 62. – Napiši program koji će slati poruke (da i ne) između dvaju micro:bitova radiovezom, ali na temelju dodirivanja pinova 0 i 1.

Rješenje:

```
from microbit import *
import radio
radio.on()
while True:
    sleep (1000)
    if pin0.is_touched():
        radio.send("DA")
    if pin1.is_touched():
        radio.send("NE")
    paket = radio.receive()
    if paket == "DA":
        display.scroll("da")
    if paket == "NE":
        display.scroll("NE")
```

Sve bi trebalo biti jasno. Vrlo je važno dodati zadršku (u našem slučaju od jedne sekunde) kako bi micro:bit uopće mogao/stigao reagirati na dodir pina.

Vježba 63. – Napiši program koji će uspostaviti radiovezu između dvaju micro-bitova. Pritiskom gumbića A oni će moći jedan drugomu slati animaciju ugrađenih slika CLOCKS.

Rješenje:

```
from microbit import *
import radio
satovi = [Image.ALL_CLOCKS]
radio.on()
while True:
    if button_a.was_pressed():
```

```
        radio.send("satovi")
    paket = radio.receive()
    if paket == "satovi":
        display.show(Image.ALL_CLOCKS)
```

Prvo definiramo varijablu `satovi` (`satovi = [Image.ALL_CLOCKS]`) u kojoj će se nalaziti lista s ugrađenim slikama svih pozicija satova od 1 do 12. Ostalo je samo igra s obzirom na znanje koje posjedujemo.

Zadatci za ponavljanje

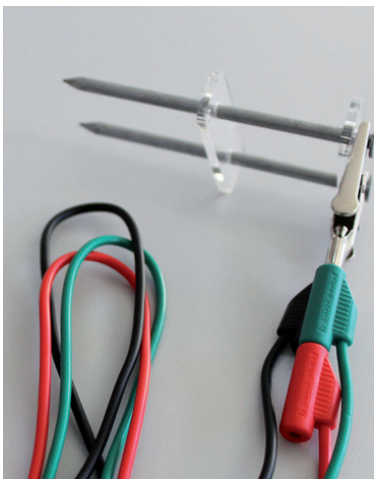
1. Napiši program koji će omogućiti komunikaciju između dvaju micro-bitova u obama smjerovima. Neka se šalju poruke ugrađene slike SAD kada se dodirne pin 0 i ugrađene slike SMILE kada se dodirne pin 2.
2. Napiši program koji će omogućiti komunikaciju između dvaju micro-bitova u obama smjerovima. Međusobno će si micro:bitovi slati poruke OK i NO na temelju pritisnutih gumbića A i B.
3. Napiši program koji će dopuniti vježbu 63. tako da pritiskom gumbića B micro:bitovi međusobno šalju animaciju svih strelica (ALL_ARROWS) te da se nakon animacija obriše zaslon.
4. Napiši program koji će uspostaviti radiovezu između dvaju micro-bitova. Oni će slati poruke jedan drugomu pritiskom gumbića A i B. Slat će si dvije vrste animacija. Jedna je da se točkica kreće sredinom zaslona micro:bita s lijeve na desnu stranu, a druga je da se točkica kreće od dolje prema gore sredinom zaslona micro:bita. Točkica podrazumijeva jednu LED-icu zaslona. Uvijek je osvijetljena samo jedna, a sljedeća će biti osvijetljena nakon 200 milisekundi.

21. ZALIJEVANJE BILJKE – MICROPYTHON

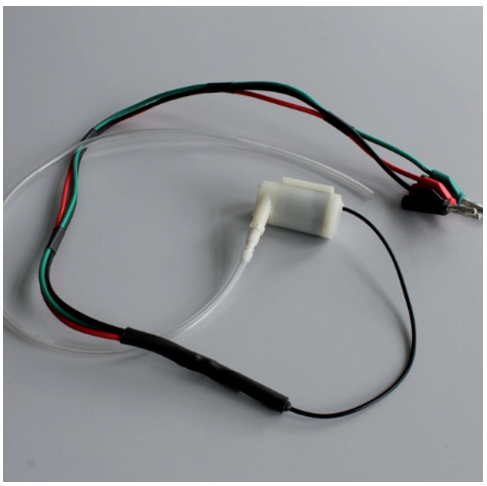
Micro:bit roboti mogu služiti da se kroz igru i zabavu nauči programiranje, a mogu imati i sasvim ozbiljnu te praktičnu primjenu. Sad ćemo naučiti jedan od takvih primjera.
Automatizacija bilo koje proizvodnje će uvijek iznjedriti bolje rezultate bilo da se poveća brzina proizvodnje ili kvaliteta proizvodnje. Ovaj naš primjer zamijenit će osobu zaduženu za zalijevanje biljaka odnosno postavit će mogućnost pogreške na nulu. Svaka će biljka biti zalijevana na vrijeme i s dovoljnom količinom vode.

Što nam sve treba?

- Senzor vlažnosti tla – slika 36.
- Vodena pumpa – slika 37.
- Micro:bit
- Biljka u tegli
- Posuda s vodom



Slika 36 – senzor vlažnosti tla



Slika 37. – vodena pumpa

Algoritam

U programiranju se često preskače algoritam, a možda ćemo već pri njegovu kreiranju otkriti bolji, brži i lakši način da nešto napravimo.

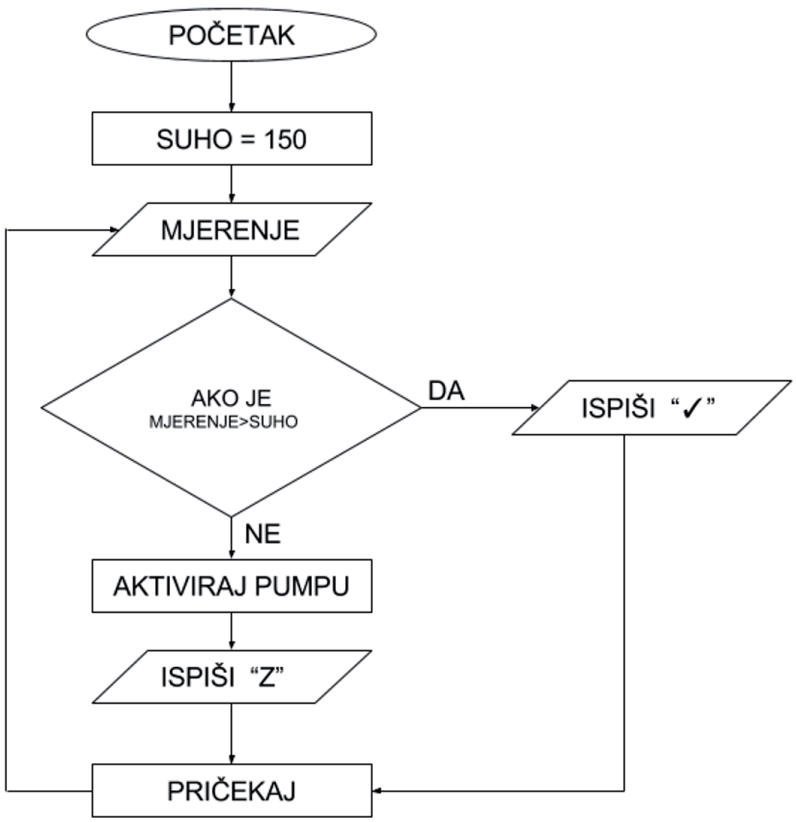
POČETAK
Provjera vlažnosti tla
Ako je vlažnost „U redu", pričekaj određeno vrijeme pa provjeri ponovno.
Ako je vlažnost „Kritična" (premala), aktiviraj vodenu pumpu na određeno vrijeme.
ZAVRŠETAK

Prvo treba provjeriti kolika je vlažnost tla. Aktiviranjem senzora za provjeru vlage dobivamo ulaznu vrijednost (MJERENJE) na temelju koje se postupak dalje grana. Prije cijelog postupka moramo odrediti kolika je granica za

preslabu vlažnost tla (vrijednost SUHO). Usporedimo li MJERENJE s vrijednosti SUHO, dobivamo razlog grananja programa. Ako je vrijednost koju smo izmjerili veća od minimalne vlažnosti (SUHO), isključujemo senzor na određeno vrijeme. Ako mjerenje vlage prikazuje premalu vlažnost tla, uključujemo pumpu za navodnjavanje.

Možda bi bilo bolje to prikazati malo jasnije. Poslužit ćemo se dijagramom tijeka.

Dijagram tijeka



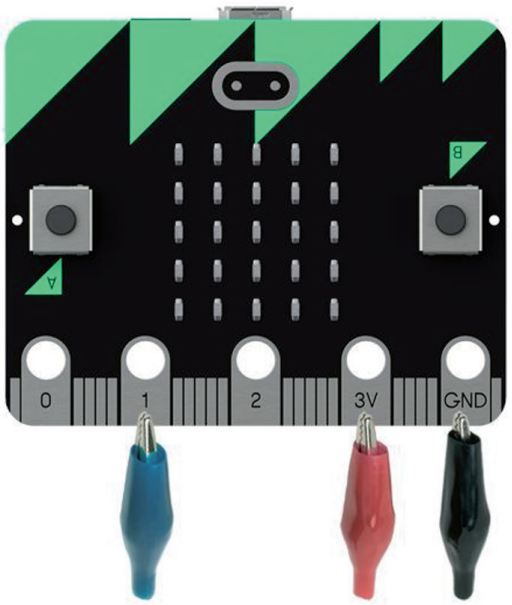
Slika 38. – Dijagram tijeka za sustav navodnjavanja

Iz dijagrama se vidi da ćemo sustav posložiti kao beskonačnu petlju koja svako određeno vrijeme provjerava vlažnost tla te na temelju tih MJERENJA „odlučuje" što će raditi (IF-ELSE petlja). Moguća su dva scenarija. Ako je sve u redu, na zaslonu micro:bita prikazat će se ugrađena slika YES („✓"), a ako nije u redu, prikazat će se slovo „Z" (zalijevanje) te će se aktivirati pumpa. Nakon svega toga slijedi stanka i postupak se ponavlja.

Programiranje

Prvo ćemo se malo poigrati sa senzorom vlažnosti tla. Izmjerit ćemo vrijednosti nekoliko vrsta zemlje da bismo dobili uvid u to kako radi senzor te kolika je nekakva minimalna vrijednost kada je potrebno zalijevati biljku.

Spojit ćemo senzor za ispitivanje vlažnosti tla kao na slici 4. Pin 1 je signalni vodič, a 3V i GND su same po sebi jasne (plus = crvena i uzemljenje/minus = crna).



Slika 39 – konektori senzora za ispitivanje vlažnosti tla

Otvorit ćemo MU editor za MicroPython i aktivirati REPL opciju (upisat ćemo naredbu po naredbu izravno u micro:bit). Upisat ćemo sljedeće:

```
>>>pin1.read_analog()
90
```

Pin 1 upisujemo zbog toga što je signalni vodič spojen na njega, a ostalo su naredbe za analogno očitanje vrijednosti. Očitana vrijednost mjerenja je 90.

Nakon nekoliko mjerenja zaključili smo da je nekakva granica (ispod koje se računa da je tlo previše suho) 150 i da će ta vrijednost odlučivati o akcijama našeg programa.

Prvo ćemo napisati dio kôda bez vodene pumpe. Samo ćemo odrediti grananje i ispise na zaslonu micro:bita.

```
from microbit import *

SUHO = 150

while True:
    pin1.write_digital(1)
    sleep(2000)
    if pin1.read_analog() > SUHO:

        display.show(Image.YES)
    else:
        display.show("Z")
    pin1.write_digital(0)
    sleep(300000)
```

Objasnit ćemo korak po korak što smo napravili.

```
SUHO = 150.
```

- Postavit ćemo vrijednost varijable SUHO na 150.

```
while True:
```

- Započet ćemo beskonačnu `while` petlju.

```
pin1.write_digital(1)
sleep(2000)
```

- Uključit ćemo senzor (postavit ćemo mu vrijednost na 1).
- Ostavit ćemo ga da se smiri dvije sekunde – preporuka proizvođača (da bi mjerenja bila što vjerodostojnija).

```
if pin1.read_analog() > SUHO:
    display.show(Image.YES)
else:
    display.show("Z")
```

- Provjerit ćemo očitanje vlažnosti tla (`pin1.read_Analog()`) i usporedit ga s vrijednosti SUHO.
- Ako je očitanje veće od minimalne vlažnosti tla, na zaslonu će se prikazati kvačica (`display.show(Image.YES)`), a u suprotnom slučaju ispisat će se slovo „Z" (zalijevanje potrebno).

```
pin1.write_digital(0)

    sleep(300000)
```

- Nakon svega se senzor isključuje jer nema potrebe da stalno radi (`pin1.write_digital(0)`).
- Isključuje se na pet minuta (`sleep(300000)`) prema ovom programu.
- Kako pet minuta? Vrijednosti koje upisujemo su u milisekundama (300000/1000=300 sekundi; 300/60=5 minuta).
- Ovo se vrijeme podesi proizvoljno.

Konačno rješenje

```
from microbit import *
SUHO = 150
while True:
    pin1.write_digital(1)
    sleep(2000)
    if pin1.read_analog() > SUHO:
        display.show(Image.YES)
    else:
        display.show(Z)
        pin0.write_digital (1)
        sleep (5000)
        pin0.write_digital (0)
```

```
    pin1.write_digital(0)
    sleep(300000)
```

Prije svega, potrebno je spojiti i vodenu pumpu na micro:bit. Treba spojiti signalni vodič na pin 0, a preostale dvije na plus i minus.

U prethodni smo program dodali dio kôda koji je istaknut žutom bojom.

```
pin0.write_digital (1)
sleep (5000)
pin0.write_digital (0)
```

- Vrlo je jednostavno objasniti što se događa kada se aktivira taj dio kôda.
- Pokretanje vodene pumpe (`pin0.write_digital (1)`)
- Čekanje pet sekundi (`sleep (5000)`) dok vodena pumpa radi – pumpa vodu u teglu (vrijeme je proizvoljno)
- Isključivanje vodene pumpe (`pin0.write_digital (0)`)

Slika 40. – spojeni sustav

Zadatci

1. Promijeni vrijeme provjere vlažnosti tla u jedan sat.
2. Promijeni vrijeme rada pumpe na 10 sekundi.
3. Postavi drugu „sliku" koja će signalizirati da je tlo dovoljno vlažno (koristi se ugrađenim slikama te kreiraj još jednu svoju).
4. Zamijeni signalizacijske pinove za senzor i pumpu.