Závěr

Nyní jste již prošli celou učebnici a měli byste micro:bit a MicroPython ovládat. Učebnice i kvůli některým úmyslným zjednodušením nemůže nahradit manuál, a proto pokud vám něco chybělo anebo se chcete dozvědět více, navštivte stránky MicroPythonu pro micro:bit.

Na úplný závěr si nyní můžete ještě vyzkoušet závěrečný příklad. Pro jeho zvládnutí potřebujete nejméně dva micro:bity a doporučena je práce v nejméně dvoučlenné skupině. Zadání příkladu naleznete v Průvodci hodinou anebo v Pracovním listě.

Co dál

Možná vás nyní napadne, co s micro:bitem dále. Zde bychom poradili zkusit žákům obstarat nějaký „microbití“ hardware. Dají se sehnat například vozítka připravená pro micro:bit nebo simulace herní konzole. Můžete si třeba také zkusit ovládat pomocí micro:bitu například stále více populárního Otto bota. Nebudeme zde uvádět odkazy na konkrétní produkt nebo e-shop, věříme, že si svůj kus hardware spolu se studenty sami vyberete.

Odkazy pro další studium

https://microbit.org/

https://www.microbiti.cz/

http://robotika.sandofky.cz/otto-bot/

PŘÍLOHA

Syntaxe některých použitých programových konstrukcí Pythonu.

PODMÍNĚNÝ PŘÍKAZ

Provede se pouze je-li splněna podmínka

Základní tvar

```
if (podmínka):
    příkaz1
    příkaz2
příkaz3
```

Příkazy 1 a 2 se provedou pouze je-li splněna podmínka, příkaz 3 vždy. Pro odsazení se používají **čtyři mezery**. Obvykle je možné nahradit čtyři mezery tabelátorem, ale nemusí to být vždy pravidlem, naopak použití tabelátoru může někdy způsobit chybu. Většina editorů pozná podmíněný příkaz díky dvojtečce na konci podmínky a odsazení zařídí sama. Konec podmínky je dán ukončením odsazování.

Příklad:

```
if (a<>0):
    b = 1 /a
    print(b)
```

Předchozí podmínku lze psát pouze

```
if (a):
```

Pokud výraz v závorce (hodnota a) má hodnotu různou od nuly je podmínka brána jako splněná. Pokud naopak je roven nule pak jako nesplněná.

Používají se relační (porovnávací) znaménka:

- < - menší
- > - větší
- == - rovno (neplést s = přiřazovací příkaz)
- != - nerovno
- <= - menší nebo rovno
- >= - větší nebo rovno

Logické spojky:

- **and** (a zároveň)
- **or** (nebo)
- **not** (negace , ne)

Příklad:

```
if ((a>1) and (a<2)):
```

totéž je:

```
if not((a<=1) or (a>=2)):
```

Poznámka – závorky nejsou povinné, ale pomáhají v přehlednosti

Rozšířený podmíněný příkaz

```
if (podminka1):
    příkazy
else:
    příkazy
```

Není-li splněna podmínka provedou se příkazy za `else`. Příklad

```
if (a):
    b = 1/a
    print("b je ",b)
else:
    print("Nelze dělit nulou")
```

Příkaz lze dále rozvětvit pomocí části `elif`, kterou lze opakovat libovolněkrát:

```
if (podminka1):
    příkazy
elif (podminka2):
    příkazy
elif(podminka3):
    příkazy
else:
   příkazy
```

Poznámky

- část s `else` není povinná
- nezapomínat na dvojtečku za podmínkou nebo `else` – častá chyba
- namísto části s `elif` lze použít vícenásobné opakování s podmínkou `if`

CYKLUS WHILE

Jedná se o cyklus s proměnným počtem opakování – před začátkem nevíme kolikrát proběhne. Nemusí proběhnout ani jednou.
Syntaxe:

```
while (podmínka):
     příkazy
```

Př:

```
i = 0
while (i < 5):
    i = i+1
    print(i)
```

Je možné použít příkazy

- `continue` – skok na začátek cyklu s opětovným vyhodnocením podmínky.
- `break` – vyskočení z cyklu

Příklad:

```
i = 0
while (i < 5):
    i = i+1
    if (i == 2):
        continue
    if (i == 6):
        break
    print(i)
```

Nekonečný cyklus – jako podmínku použijeme True. Např:

```
while True:
    vstup = input("Zadej matematicky vyraz: ")
    print(eval(vstup))
```

Jediná možnost, jak takový cyklus ukončit je pomocí příkazu `break`.

CYKLUS FOR

Jedná se o cyklus s pevným počtem opakování – víme kolikrát proběhne
Syntaxe:

```
for promenna in (iterovatelný objekt):
     příkazy
else:
     příkazy
```

Část za else se provede, pokud není podmínka splněna. Tato část je nepovinná a obvykle se nepoužívá.
iterovatelný objekt – lze jej rozložit na části, MicroPython podporuje:
`range(A)` – od 0 do A-1, po jedné
`range(A,B)` – od A do B-1, po jedné
`range(A,B,C)` – od A do B-1 po C

Příklady:

```
for I in range (4):
    print(I)
```

(provede se pro 0,1,2,3)

```
for I in range (2,4):
    print(I)
```

provede se pro 2,3

```
for I in range (2,8,2):
    print(I)
```

provede se pro 2,4,6

```
for I in range (6,3,-1):
    print(I)
```

provede se pro 6,5,4