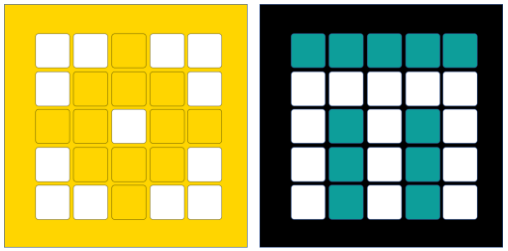
PRIME LESSONS

By the Makers of EV3Lessons



LOOPS

BY SANJAY AND ARVIND SESHAN

This lesson uses SPIKE 3 software

Obiectivul lecției

- Învățăm cum să repetăm o acțiune utilizând LOOPS

COD DE REPETARE

- Să spunem că dorești ca robotul tău să repete o acțiune din nou și din nou.
  - Vei copia codul din nou și din nou?
  - Cum ar fi dacă ai dori ca acțiunea să se repete la infinit?
- Poți folosi LOOPS pentru a repeta o acțiune pentru un anumit număr de ori sau până când se îndeplinește o anumită condiție
- Python are 2 tipuri de LOOPS: Loops și While loops

WHILE LOOPS

- Să spunem că dorim ca un task să fie executat până când o condiție se îndeplinește (Adevărat)
  - E.g. Cât sunt în librărie, stai liniștit
- În Python, vom utiliza declarația while: să rulăm codul cât timp declarația este adevărată
- În exemplul din dreapta, x==8 este întotdeauna Adevărată, așa că "Yay!" va fi afișat mereu
  - Dacă introduci x=10 în interiorul LOOP, "Yay!" va fi afișat doar o singură dată, de exemplu, LOOP-ul nu va continua.

```
x = 8
while (x == 8):
    print("Yay!")
```

Output:

```
Yay!
Yay!
Yay!
... [repeats forever]
```

Notă:
Adu-ți aminte să introduci codul pe care vrei să-l rulezi într-un LOOP

WHILE LOOPS ÎN SPIKE PRIME

- LOOP While sunt utile pentru repetarea unei comenzi până la citirile unui senzor:

```
# Mergi înainte până când senzorul de distanță returnează o valoare
<=10cm
while (getDistance() > 10):
    moveForward()
# Ia în considerare că getDistance() primește valoarea distanței în
# centimetri și moveForward() mișcă robotul înainte
```

Note:
Remember to indent the code you want to run in the loop

WHILE LOOPS NEDEFINITE

- Poți utiliza while loops pentru a repeta la infinit

```
while True:
        Code
```

- Prin setarea unei condiții care să fie **Adevărată** tot timpul, loop-ul va repeta codul tot timpul

PROVOCARE

- Creează o variabilă x și atașează-i o valoare
- Creează un While loop care afișează toate pătratele (e.g., 4, 9, 16, …) care sunt mai mici ca x pe HUB

PROVOCAREA SOLUȚIE

```
from hub import light_matrix
import runloop, math

async def main():
    # aceasta crează variabila x și setează valoareea lui x la 51
    x = 51

    # aceasta creează variabila y care va fi folosită ca a
    # loop counter. Începem cu y = 1
    y = 1

    # aceste loops repetă acțiunea până când pătratul lui y este >= x
    while(math.pow(y,2) < x):
        # pentru a afișa, convertește un număr real la un întreg  și apoi la un șir
        # Altfel, va afișa decimale e.g. 1.0, 4.0
        await light_matrix.write(str(int(math.pow(y,2))))
        # Trebuie să creștem y pentru a considera valoarea netă
        y += 1

runloop.run(main())
```

FOR LOOPS

- Similar cu while loops, dar rulează pentru un calcul fix
  - E.g. sari de 10 ori
- Un exemplu de for loop de bază este ca în exemplul din dreapta
- În "for i în intervalul (start, sfîrșit, rată de creștere):"
  - intervalul() crează un set de numere între un număr de start și un număr mai mic decât numărul de final (sau doar un număr de sfârșit cînd este prezent unul din parametri) careau ca regulă o rată de creștere. Startul și valorile de creștere sunt opționale.
  - Variabila i ia valoarea următoare din set de fiecare dată (poți să denumești această variabilă oricum dorești; convenția standard este i, j, k)
  - În exemplu, i va fi doar între 0 și 9, din moment ce îndeplinește condiția de verificare < n (nu <=)

```
for i in range(0,10):
    print("Jump!")
    print(i)
```

Output:

```
Jump!
0
Jump!
1
....
Jump!
9
```

ANALIZĂ: PENTRU LOOPS CU INTERVAL()

- Structură de bază:

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

```
Output:
0
1
2
3
```

- Poți seta o poziție de start:

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

```
Output:
2
3
```

- la în considerare ca 4 nu este inclusă. Funcția interval () exclude maximul pe care îl setezi.
- În final, poți crește cu diferite valori peste 1

```
for i in range(2, 7, 2):
    print(i)
```

Increment

```
Output:
2
4
6
```

PENTRU LOOPS CU O LISTĂ DE NUMERE

- Funcția loops poate fi utilizată pentru a itera peste o paranteză separată de o listă de numere (enclosed by brackets [])

```
for i in [0, 2, 6]:
        print(i)
```

Output:

0

2

6

Notă: Acest exemplu utilizează liste, care nu au fost încă acoperite.

LOOP EXAMPLES

```
# A for loop repeats an action a specific number of
times
# based on the provided range
def sumFromMToN(m, n):
    total = 0
    # note that range(x, y) includes x but excludes y
    for x in range(m, n+1):
        total += x
    return total
```

```
def printStarRectangle(n):
    # print an nxn rectangle of asterisks
    for row in range(n):
        for col in range(n):
            print("*", end="")
        print()
```

```
# use while loops when there is an
indeterminate number of iterations
def leftmostDigit(n):
    n = abs(n)
    while (n >= 10):
        n = n//10
    return n
```

PROVOCARE: NUMERELE PRIME

- Țelul tău este să verifici dacă orice număr întreg pozitiv n este număr prim
- Indicii:
  - Numerele prime sunt numerele divizibile cu 1 și cu el însuși
  - Trebuie să verifici divizibilitatea numerelor dintre 2 si n-1
  - Modulo (%) te va ajuta aici (numărul poate fi testat ca întreg dacăn%a==0)

SOLUȚIA PROVOCĂRII

```
n = 3 # your number here
prime = True # start by assuming it is prime
if (n <= 1): # 1 and lower are not prime
    prime = False
if (n == 2):
        prime = True
for factor in range(3,n): # check all possible factors [3, n]
    if (n % factor == 0): # n%factor == 0 when it is a divisor
        prime = False # set that n is not prime
if prime:
    print("is prime") # the number is prime
else:
    print("not prime") # the number is not prime
```

SOLUȚIA PROVOCĂRII UTILIZÂND SPIKE PRIME

```
from hub import light_matrix
import runloop
import sys # Standard Python system library

# Function to stop the program using a system exception
def stopAndExitProgram():
    sys.exit("Stopping")

async def main():
    n = 3 # your number here
    prime = True # start by assuming it is prime
    if (n <= 1): # 1 and lower are not prime
         prime = False
     if (n == 2): # 2 is prime
         prime = True
    for factor in range(3,n): # check all possible factors [3, n]
        if (n % factor == 0): # n%factor == 0 when it is a divisor
            prime = False # set that n is not prime
    if prime:
        await light_matrix.write("Yes") # the number is prime
    else:
        await light_matrix.write("No") # the number is not prime

    # Stop and exit the program. You should see the Program number on your hub.
    stopAndExitProgram()

runloop.run(main())
```

CREDITS

- Această lecție a fost creată de Sanjay Seshan și Arvind Seshan for SPIKE Prime Lessons
- La această lecție au contribuit membrii comunității FLL Share & Learn.
- Mai multe lecții sunt disponibile pe www.primelessons.org
- Această lecție a fost tradusă în limba romană de echipa de robotică FTC – ROSOPHIA #21455 RO20