Zajęcia 4: pętle

Python dla kognitywistów

Marcin Jukiewicz

Pętle

Wykonywanie czegoś w pętli oznacza wielokrotne wykonywanie pewnego zestawu czynności.

Przykład: rozbijanie jajek
1. Weź jajko
2. Chwyć jajko dwiema rękoma
3. Uderz jajkiem o krawędź naczynia,
4. Wylej zawartość jajka do naczynia.

Jeśli masz jeszcze jakieś jajka, powtórz czynności od punktu 1.

Pętla for

Pętlę for stosujemy wtedy, kiedy wiemy dokładnie ile razy ma zostać wykonana. Poniżej przykład wypisujący kolejne cyfry od 0 do 9.

```
for i in range(10):
    print(i)
```

Litera "i" to tzw. licznik. Jest to zmienna, pod którą zapisywany jest numer aktualnie wykonywanej iteracji. Powyższy program służy jedynie do wypisywania kolejnych numerów iteracji. Funkcja "range" służy od określenia od jakiej liczby ma rozpocząć się iteracja (w tym wypadku od 0) i na jakiej ma się skończyć (w tym wypadku od 9). Należy pamiętać o tym, że wartość końcowa wpisywana jako argument funkcji "range" musi być o 1 większa od założonej wartości końcowej.
Podobnie jak w przypadku instrukcji warunkowych, w przypadku stosowania pętli istotne jest stosowanie wcięć!

Pętla for

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

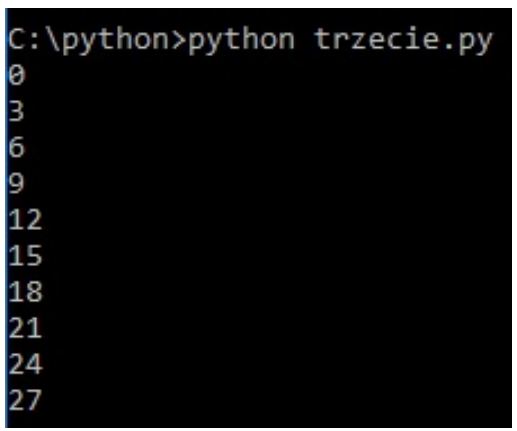
Powyższy przykład daje taki sam efekt jak ten z poprzedniego slajdu. Różnica polega na tym, że range ma tym razem dwa argumenty. Argument pierwszy, którego wcześniej nie było, służy do ustalenia początku zakresu (gdyby miał być inny niż domyślny). Przykład wyświetlenia cyfr od 5 do 9:

```
for i in range(5,10):
    print(i)
```

Pętla for

```
for i in range(0,30,3):
    print(i)
```

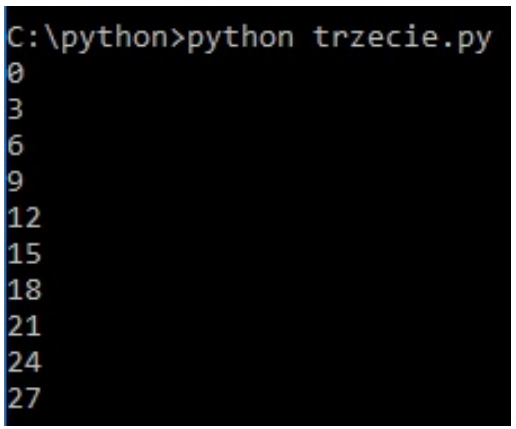
Nie każda iteracja musi zwiększać się o 1. Podając trzeci argument funkcji "range", ustalamy o ile ma się zwiększać licznik w kolejnych krokach.
efekt wykonania powyższego kodu:



Pętla for

```
for i in range(0,30,3):
    print(i)
```

Nie każda iteracja musi zwiększać się o 1. Podając trzeci argument funkcji "range", ustalamy o ile ma się zwiększać licznik w kolejnych krokach.
efekt wykonania powyższego kodu:



Pętla while

Innym typem pętli jest pętla "while". Pętla ta jest wykonywana aż do momentu kiedy określony warunek jest prawdziwy. w przykładzie, póki zawartość zmiennej ’cyfra’ jest różna od 10. Tak więc nie ma odgórnego założenia ile razy ta pętla się wykona, jak było to w przypadku pętli "for".

```
cyfra = 0
while cyfra != 10:
    print(cyfra)
    cyfra += 1
```

Podobnie jak pierwszy przykład dotyczący pętli "for", powyższy wyświetli kolejno wszystkie cyfry w zakresie od 0 do 9.

Pętla while - wciśnij przycisk aby zakończyć program

Dotychczasowe programy wykonywały się tylko raz. Aby skorzystać z nich ponownie, należało uruchomić go ponownie. Znane nam z życia programy tak się nie zachowują. Decyzję o zakończeniu wykonywania danego programu podejmuje użytkownik. Poniższy kod pozwala na wykonywanie w pętli programu dopóki nie zostanie wciśnięta litera ’q’ na klawiaturze.

```
znak=""
print("Aby zakończyć program wciśnij \"q\"")
while znak != "q":
    print("Nie wcisnąłeś \"q\"")
    znak = input()
```

Polecenie brake i pętla nieskończona

Polecenie służy to natychmiastowego przerwania wykonywanej pętli.

```
print("Aby zakończyć program wciśnij \"q\"")
while True:
    znak = input()
    if znak != "q"
        print("Nie wcisnąłeś \"q\"")
    else:
        break
```

```
print("Aby zakończyć program wciśnij \"q\"")
while True:
    znak = input()
    if znak == "q"
        break
    print("Nie wcisnąłeś \"q\"")
```

Pętla nieskończona i polecenie break

Dzięki dotychczasowej wiedzy możemy stworzyć proste menu

```
while True:
    print("Aby zakonczyc program wcisnij \"q\"")
    print("Aby wyswietlic imie autora wcisnij \"i\"")
    print("Aby wyswietlic nazwe przedmiotu i kierunku
        wcisnij \"n\"")
    znak = input()
    if znak == "i":
        print("Marcin")
    elif znak == "n":
        print("Programowanie na kognitywistyce")
    elif znak == "q":
        break
    else:
        print("Nieprawidlowy znak")
```

Pętle zagnieżdzone

Pętle zagnieżdżone, czyli pętle w pętli. Służą do tego, aby daną pętlę wykonać wielokrotnie.

```
for i in range(1,4):
    for j in range(1,5):
        print("To jest ", i ," wykonanie zewnętrznej
            pętli i ", j, " wewnętrznej pętli")
```

```
C:\python>python trzecie.py
To jest 1 wykonanie zewnętrznej pętli i 1 wewnętrznej pętli
To jest 1 wykonanie zewnętrznej pętli i 2 wewnętrznej pętli
To jest 1 wykonanie zewnętrznej pętli i 3 wewnętrznej pętli
To jest 1 wykonanie zewnętrznej pętli i 4 wewnętrznej pętli
To jest 2 wykonanie zewnętrznej pętli i 1 wewnętrznej pętli
To jest 2 wykonanie zewnętrznej pętli i 2 wewnętrznej pętli
To jest 2 wykonanie zewnętrznej pętli i 3 wewnętrznej pętli
To jest 2 wykonanie zewnętrznej pętli i 4 wewnętrznej pętli
To jest 3 wykonanie zewnętrznej pętli i 1 wewnętrznej pętli
To jest 3 wykonanie zewnętrznej pętli i 2 wewnętrznej pętli
To jest 3 wykonanie zewnętrznej pętli i 3 wewnętrznej pętli
To jest 3 wykonanie zewnętrznej pętli i 4 wewnętrznej pętli
```

Czyszczenie ekranu

```
import os
os.system("cls")
```

Powyższe linijki kodu wyczyszczą ekran konsoli. Przykład użycia z tekstem:

```
import os
print("To nie będzie widoczne")
os.system("cls")
print("To będzie widoczne")
```

Do czego służy linia ’import os’?

Instalacja dodatkowych pakietów - PIP

W najnowszych wersjach Pythona instalacja dodatkowych pakietów (a więc elementów rozszerzających jego możliwości) sprowadza się jedynie do użycia menedżera PIP. Do dalszej części zajęć potrzebujemy modułów ’colorama’ i ’termcolor’. Aby je zainstalować uruchamiamy linię komend i wpisujemy:

```
pip install colorama
```

a następnie:

```
pip install termcolor
```

Kolorowanie tekstu - z użyciem modułu termcolor i colorama

```
from colorama import init
from termcolor import colored
init()
print(colored("Cześć!", "green"))
```

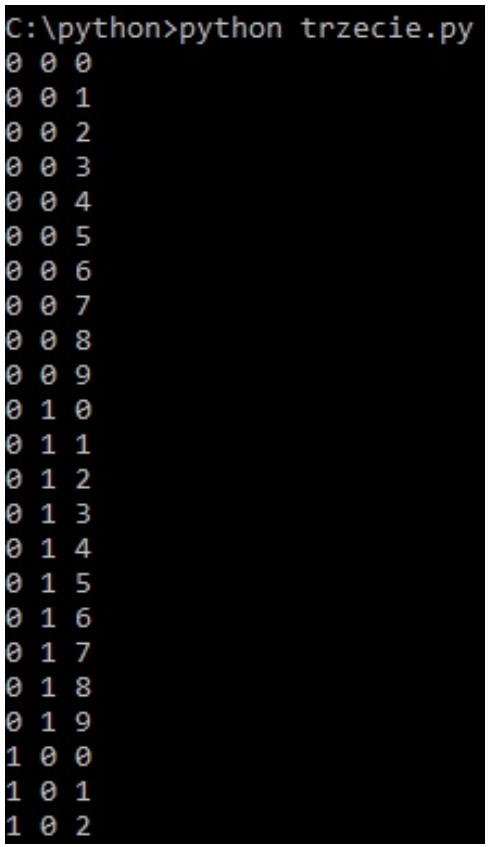
Standardowo Python nie umożliwia kolorowania tekstu. Aby umożliwić kolorowanie tekstu musimy zaimportować zewnętrzne moduły. Służą do tego dwie pierwsze linijki kodu. init() uruchamia moduł colorama.
Do samego kolorowania tekstu służy funkcja "colored". Jej pierwszym argumentem jest wyświetlany tekst a drugim nazwa koloru wyświetlanego tekstu.

Zadania

1. Rozbuduj kalkulator w poprzednich zajęć. Tym razem program nie powinien kończyć się po jednokrotnym wykonaniu operacji arytmetycznej. Niech o końcu działania decyduje użytkownik.
2. Przygotuj program, który policzy sumę liczb od 0 do 50 (wykorzystaj pętlę).
3. Przygotuj program, który wyświetli liczby od 0 do 100, poza: 4, 34, 57-74, 78-90, 92 (wykorzystaj jedną pętlę).
4. Za pomocą poznanych metod kolorowania tekstu wyświetl kolorową choinkę z kolorowymi bombkami.

Zadania

6. Przygotuj program wyświetlający wszystkie liczby z zakresu od 0 do 999. Nie korzystaj z jednej pętli "for i in range(0,1000)". Wykorzystaj trzy pętle. Każda cyfra danej liczby powinna być wyświetlana osobno.



7. Zmodyfikuj powyższy program tak, aby zliczał ilość wystąpień cyfry 4.

Zadania

8. Przygotuj program, który liczy sumę liczb z zakresu od 25 do 50. Wykonaj to w dwóch wariantach: za pomocą pętli for i while.
9. Przygotuj program, który liczy sumę liczb parzystych z zakresu od 25 do 50. Wykonaj to w dwóch wariantach: za pomocą pętli for i while.
10. Przygotuj program, który wyświetla tabliczkę mnożenia.
11. Przygotuj program, który wyświetla instrukcję zrobienia jajecznicy. Niech na początku pyta z ilu jajek ma być przygotowana. Elementy, które się powtarzają powinny być wyświetlane za pomocą pętli.
12. Wstaw 2, 3 i 4 zadanie do menu ze slajdu 12.
13. Wstaw jak najwięcej zadań do menu ze slajdu 12.