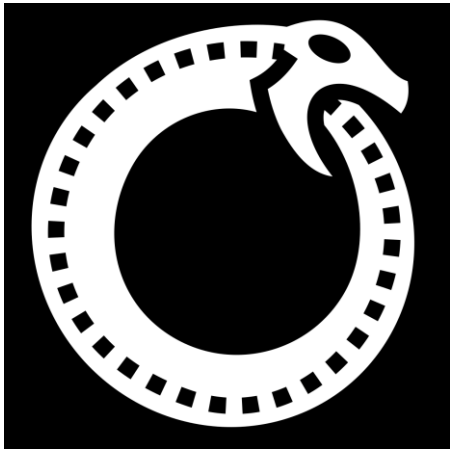


Pętla FOR w Pythonie

M@я3k Pų𝔡€£kØ

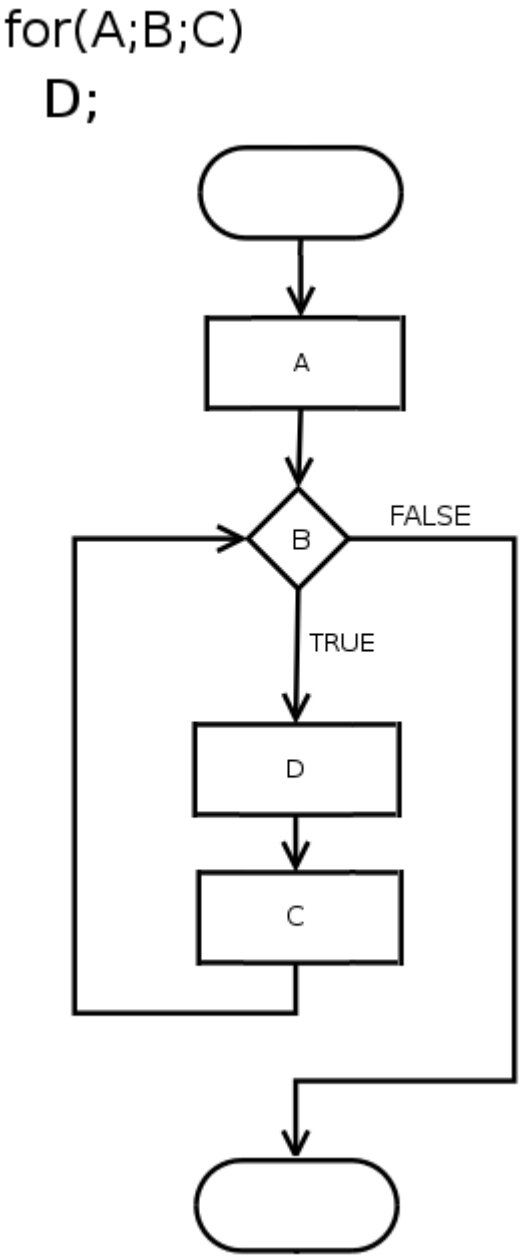
Programowanie w Pythonie

Spis treści

- Pętla w programowaniu.
- Pętla for w Pythonie
- Cechy pętli for
- Instrukcja range
- Pętle zagnieżdżone
- Wyjście z pętli

Pętla w programowaniu

- Pętla to element programowania, który umożliwia wykonywanie danego fragmentu kodu wielokrotnie.



Pętla for w Pythonie

- Pętla **for** służy do iterowania (kolejnego powtarzania operacji) po elementach sekwencji, wykonując powtarzalny blok kodu dla każdego elementu.

```
for zmienna in sekwencja:
    wykonywane_operacje

liczby = [1,2,3]
for l in liczby:
    print(l)
```

Cechy pętli for

- Automatyczne zakończenie:
  - Pętla `for` kończy się po przetworzeniu wszystkich elementów w sekwencji.
- Zmienna iteracyjna:
  - W każdej iteracji zmienna przechowuje bieżący element z sekwencji.
- Użycie z range():
  - Funkcja `range (start, stop, step)` jest często używana do generowania sekwencji liczb, co pozwala na wykonanie pętli określoną liczbę razy.
- Wcięcie:
  - Blok kodu pętli jest definiowany za pomocą wcięć, podobnie jak w instrukcjach `if`.
- Dwukropek:
  - Na końcu linijki z `for` należy postawić dwukropek :
- Zazwyczaj używa się kluczowego słowa `IF` (lub `IF-ELSE`) do wyboru wykonywanego kodu.

Instrukcja range

- Instrukcja `range` służy do kontroli wykonywania pętli `for`.
  - Generuje sekwencje liczb całkowitych z ustalonego zakresu.
  - Używana do wykonania pewnej liczby przejść (iteracji).
- `range` ma trzy główne postacie:

`range(stop)`	Tworzy sekwencję liczb od **0** do **stop-1**
`range(start, stop)`	Tworzy sekwencję liczb od **start** do **stop-1**
`range(start, stop, step)`	Tworzy sekwencję liczb od **start** do **stop-1** z krokiem **step**

Przykłady pętli for

- Generuje liczby od 0 do 9

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

- Generuje liczby od 1 do 10

```
for i in range (1, 11):
    print (i)
```

- Generuje co drugą liczbę z zakresu od 1 do 10

```
for i in range (1, 11, 2):
    print (i)
```

Ćwiczenie 1

1. Napisz program wypisujący liczby od **n** do **m**. Liczby **n** i **m** wczytaj z klawiatury.
2. Napisz program wypisujący kolejny potęgi liczby 2 od 1 do n. Liczbę **n** wczytaj z klawiatury.
3. Wczytaj liczbę **x** z klawiatury. Użyj operatorów // i %, aby obliczyć sumę jej cyfr.
4. Dany jest pierwszy wyraz a1 = 2 i iloraz q = 3. Oblicz **n-ty** wyraz ciągu geometrycznego przy pomocy operatora potęgowania **.
5. Liczba mieszkańców miasta wynosi 1000. Co roku rośnie o 3 %. Oblicz, ile osób tam będzie mieszkało po **n** latach.
6. Zmienna **x** = 1. W każdej iteracji pętli zwiększ ją o 50 % (x *= 1.5) przez 10 kroków. Wypisz wartość **x** po każdej zmianie.
7. Pobierz liczbę całkowitą **n**. Oblicz **n!** (silnię).
8. Wczytaj liczbę **a** i procent **p**. Oblicz jak rośnie kwota **a** przy wykorzystaniu procentu składanego.
   - Procent składany polega na tym, że liczbę **a** zwiększamy za każdym razem o wartość procentu **p**.

Pętle zagnieżdżone

- W języku Python możemy umieszczać jedne pętle w drugich:

```
for i in range(1,11):
    for j in range(1,11):
        k = i*j
        if j == 10:
            print(k)
        else:
            print(k, "\t", end = " ")
```

- Pętla zewnętrzna kontroluje liczbę pełnych wykonani pętli wewnętrznej.
- Pętla wewnętrzna jest wielokrotnie wykonywana w całości dla każdej iteracji pętli zewnętrznej. Musi zakończyć wszystkie swoje iteracje, zanim pętla zewnętrzna przejdzie do kolejnej iteracji.

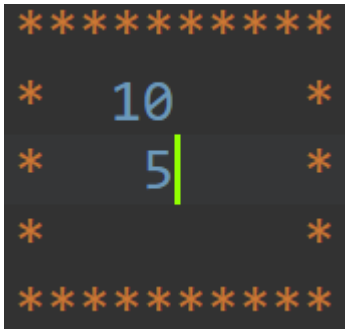
Zastosowanie pętli zagnieżdżonych

- Pętle zagnieżdżone są używane dla danych wielowymiarowych. Umożliwiają efektywne przejście przez elementy tablic czy list wielowymiarowych.
  - Macierze wielowymiarowe
- Są użyteczne do tworzenia wszystkich możliwych kombinacji elementów z dwóch lub więcej zbiorów.
- Gdy potrzebne są pewne wielokrotne zestawy działań dla każdego elementu zewnętrznej pętli.
  - Na przykład, w przypadku, gdy masz listę użytkowników, a dla każdego użytkownika musisz przeglądać listę jego zamówień
- Zagnieżdżenie może sprawić, że kod będzie bardziej zrozumiały niż alternatywne, „spłaszczone” rozwiązania.

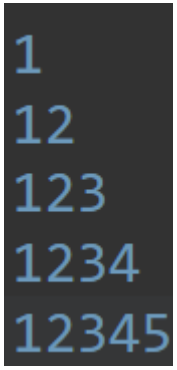
Ćwiczenie 2

1. Napisz program wczytujący liczby **n** i **m**. Następnie narysuje prostokąt złożony z gwiazdek o wymiarach **n** na **m**.

2. Napisz program wczytujący liczby **n** i **m**. Następnie narysuje ramkę złożoną z gwiazdek o wymiarach **n** na **m**.

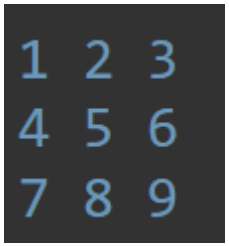


3. Napisz program wypisujący liczby w trójkącie. Wczytuje liczbę n wypisuje liczby według wzoru na rysunku.

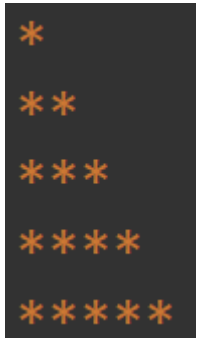


4. Napisz program rysujący szachownicę. Wczytuje rozmiar wiersza i kolumny, oraz znaki jakie ma rysować naprzemian.

5. Napisz program wypisujący liczby występujące na wszystkich kostkach domina.
   - (Przykład: 0-0, 0-1, ... 5-5, 5-6, 6-6)
6. Napisz program wczytujący liczbę n i rysujący kwadrat **n** x **n**. W kolejny rzędach mają być kolejne liczby od 1 do n



7. Napisz program wczytujący liczbę n i wypisujący liczby od 1 do $\mathbf{n}^2$. Rysuje je w kwadracie **n** x **n**.
8. Napisz program wczytujący liczbę **n** i rysujący trójkąt z gwiazdek o wymiarach **n** x **n** według wzoru na rysunku.



Wyjście z pętli

- Instrukcja `break` wyskakuje z najbardziej wewnętrznej pętli for:

```
for n in range(10, 1, -1):
    for x in range(n, -n, -1):
        if x == 0:
            print("nastąpiło dzielenie przez zero")
            break
        if n % x == 0:
            print("Dzielnikiem ", n, " jest ", x)
```

- Instrukcja break nie służy do obsługi wyjątków, a do natychmiastowego przerwania najbliższej pętli, przenosząc sterowanie do pierwszej instrukcji po niej.

Powtórzenie

1. Do czego służą pętle w językach programowania?
2. Jak jest realizowana pętla for w Pythonie?
3. W jaki sposób zdefiniować dokładną ilość wykonywanych operacji?
4. Jak zrealizować obsługę wielowymiarowych struktur danych?
5. Jak wyjść z pętli w razie nagłego błędu?