# Python: komentarze, typy zmiennych, operatory,  instrukcje warunkowe.

## Komentarze

Komentarze są to fragmenty kodu źródłowego, które nie są wykonywane i mają za zadanie objaśnić dany fragment kodu. W języku Python komentarz jest to kod po prawej stronie znaku: **#.**

```
# przykładowy komentarz
```

## Zmienne

Zmienne są to konstrukcje programistyczne pozwalające na przechowywanie danych oraz operowanie na nich. Występują różne typy zmiennych w zależności od danych jakie przechowują. Mogą to być wartości liczbowe (całkowite i rzeczywiste), logiczne (True/False), ciągi znaków lub bardziej skomplikowane struktury. W języku Python w odróżnieniu od wielu innych języków wysokiego poziomu, programista nie musi określać typu zmiennej – robi to za niego interpreter w momencie deklaracji.

## Zmienne tekstowe

Zmienne tekstowe, inaczej zwane zmiennymi typu string lub po prostu stringami służą do przechowywania w pamięci ciągów znaków. Deklaracja odbywa się użyciem pojedynczych lub podwójnych cudzysłowów:

```
zmienna_tekstowa = "przykladowy tekst"
zmienna_tekstowa = 'przykladowy tekst'
```

Możliwe jest także zadeklarowanie bloku tekstu:

```
blok_tekstu = '''Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Proin nibh augue, suscipit a, scelerisque sed, lacinia in, mi. Cras vel lorem.
Etiam pellentesque aliquet tellus. Phasellus pharetra nulla ac diam. Quisque
semper justo at risus. Donec venenatis, turpis vel hendrerit interdum, dui
ligula ultricies purus, sed posuere libero dui id orci. Nam congue, pede vitae
dapibus aliquet, elit magna vulputate arcu, vel tempus metus leo non est.'''
```

Tutaj także możemy zamiennie używać """ i '''.

## Zmienne liczbowe

W języku Pyathon występują dwa typy zmiennych liczbowych: całkowite i rzeczywiste:

```
zmienna_calkowita = 1
zmienna_rzeczywista = 1.0
```

## Operatory arytmetyczne

W języku Python do wykonywania operacji arytmetycznych używamy standardowych operatorów znanych z lekcji matematyki: + - / *

```
skladnik_a = 2
skladnik_b = 3
suma = skladnik_a + skladnik_b
print(suma)

Inne przydatne operator:
// - dzielenie całkowite
% - dzielenie modulo (reszta z dzielenia)
** - potęgowanie
== - operator porównania

Potęgowanie może odbywać się także za pomocą funkcji pow():
pow(2, 3)
```
gdzie pierwszy argument to podstawa, a drugi to wykładnik.

## Tablice

Tablice służą do przechowywania w pamięci sekwencji danych, możemy sobie je wyobrazić jako zmienne ustawione w szereg.

```
tablica = [1, 2]
tablica.append(3)
print(tablica[0]) # wypisze 1
print(tablica[1]) # wypisze 2
print(tablica[2]) # wypisze 3
```

Dodawanie (potocznie: *wrzucanie)* wartości do tablicy odbywa się podczas deklaracji albo z użyciem funkcji *append()* wywoływanej „na danej tablicy”. Żeby dostać się do przechowywanych wartości używamy nawiasów kwadratowych i indeksu.

**Uwaga: Indeksowanie tablicy w języku Python zaczyna się od 0!**

Przydatnym narzędziem jest metoda *len(),* która zwraca długość tablicy:

```
tablica = [1, 2, 3]
print(len(tablica)) # 3
```

Ćwiczenie: sprawdzić jaki będzie wynik użycia operatora dodawania na dwóch tablicach.

## Formatowanie tekstu

Łączenie tekstu i parametrów może odbywać się na dwa sposoby: poprzez użycie funkcji *format* lub za pomocą operatora dodawania:

```
wiek = 20
imie = 'Adam'
print('{0} ma {1} lat'.format(imie, wiek))
```

```
wiek = 20
imie = 'Adam'
print(str(imie) + " ma " + str(wiek) + " lat")
```

W drugim przypadku wykonaliśmy tzw konkatenację (łączenie) stringów. Konieczne było użycie funkcji str(), która zamienia typ liczbowy na ciąg znaków.

## Znak końca linii

Każde wywołanie funkcji *print()* zakończone jest domyślnie przejściem do nowej linii – na końcu dopisywany jest „niewidoczny" znak końca linii: ***\n*** **.** Aby nie przechodzić do następnej linii do wywołania funkcji dopisujemy dodatkowy parametr *end:*

```
print('a', end='')
print('b', end='')
```

Znak konca linii możemy także wstawić celowo, aby wyświetlony tekst był podzielony na linie wg naszego uznania:

```
print('To jest pierwsza linia\nTo jest druga linia')
```

```
Inne, podobne znaki specjalne:
\t – tabulacja
\r – powrót karetki
\\ - ukośnik
```

Więcej: https://pl.python.org/docs/ref/node9.html

Stringi są to w rzeczywistości tablice znaków, więc możemy tutaj zastosować też techniki znane z tablic:

```
napis = "abcdefghijk"
print(napis[2]) # c


napis = "abcdefghijklmnop"
print(napis[3:7]) # defg

napis = "abcdefghijklmnop"
print(napis[3:]) # defghijklmnop
```

Metoda *count()* pozwala na zliczenie występowania danej wartości:

```
napis = "abrakadabra"
print(napis.count("a"))  # 5
print(napis.count("ab")) # 2
```

Metoda *index()* zwraca indeks pierwszego wystapienia zadanej wartości:

```
napis = "abcdeabcde"
print(napis.index("a")) # 0
print(napis.index("d")) # 3
```

Do zamiany znaków na wielkie bądź małe litery używamy metody *upper()/lower():*

```
napis = "Witaj"
print(napis.upper()) # WITAJ
print(napis.lower()) # witaj
```

Metoda *split()* przydaje się gdy chcemy podzielić tekst na pojedyncze fragmenty (słowa, zdania, linie etc):

```
napis = "Ala ma kota."
tablica_slow = napis.split(" ")
print(tablica_slow) # ['Ala', 'ma', 'kota']
```

Do usuwania tzw białych znaków z początku i końca ciągu znaków służy metoda *strip():*

```
napis = "    Ala ma kota.    "
tablica_slow = napis.strip()
print(tablica_slow) # Ala ma kota.
```

**Zadanie 1:**
**Zmodyfikuj poniższy kod w taki sposób, aby aby wyświetlone wartości odpowiadały tym zadanym w komentarzach:**

```
s = "Jaki powinien byc ten napis?"

# Powinien byc dlugi na 17 znakow
print("Dlugosc napisu = {}".format(len(s)))

# Pierwsza litera 'a' w tekscie powinna miec indeks nr 1
print("Indeks pierwszej litery 'a' = {}".format(s.index("a")))

# W napisie musza byc dwie litery 'a'
print("'a' wystepuje {} razy".format(s.count("a")))

# Dzielenie napisu na kawalki
print("Pierwszych piec znakow to {}".format(s[:5]))
print("Nastepne piec znakow to {}".format(s[5:10]))
print("Dwunastym znakiem jest {}".format(s[12]))

print("Ostatnie piec znakow to {}".format(s[-5:]))

# Zamien wszystkie male litery na duze
print("Uzywajac duzych liter: {}".format(s.upper()))

# Zamien wszystkie litery na male
print("Uzywajac malych liter: {}".format(s.lower()))

# Rozdziela napis na trzy czesci,
# z ktorych kazda zawiera tylko jedno slowo
print("Napis rozdzielony na slowa: {}".format(s.split(" ")))
```

## Instrukcje warunkowe

Instrukcje warunkowe pozwalają na wykonywanie różnych obliczeń w zależności od tego czy zdefiniowane wyrażenie logiczne jest prawdziwe czy fałszywe.

Poniższy program wczytuję liczbę wprowadzoną przez użytkownika i porównuje ją do tej zadeklarowanej w kodzie. W zależności od wprowadzonej wartości wykonuje inny blok kodu (wypisuje inny napis).

```
liczba = 20
wprowadzona_liczba = int(input('Wprowadz liczbę : '))
if wprowadzona_liczba == liczba:
  print('Brawo, to ta liczba.')
elif wprowadzona_liczba < liczba:
  print('Wprowadzona liczba jest mniejsza od {}'.format(liczba))
else:
  print('Wprowadzona liczba jest wieksza od {}'.format(liczba))
print('Done')
```

Funkcja *input()* zczytuje z dane wprowadzone przez użytkownika (w tym przypadku następuje od razu jej przypisanie do zmiennej)

Funkcja *int()* zamienia (rzutuje) typ string na int.

Instrukcja rozpoczyna się od słowa kluczowego *if*, następnie mamy wyrażenie logiczne (zwracające wynik True/False), linia kończy się znakiem dwukropka. Jeśli dane wyrażenie jest prawdziwe to wykona się pierwszy blok kodu (występujący bezpośrednio po *„if-ie”*. Bardzo istotne jest tutaj wcięcie – to ono określa gdzie zaczyna się i kończy dany blok kodu. Następnie występuje instrukcja *elif*, która wykona się jeśli wyrażenie było fałszywe i tutaj analogicznie jak wyżej: następuje sprawdzenie kolejnego warunku i analogicznie jeśli jest on spełniony, nastąpi wykonanie kodu w bloku poniżej. W sytuacji gdy żadne z wyrażeń nie jest prawdziwe wykona się blok kodu pod słowem *else*.

Część *elif* nie jest w tej instrukcji konieczna i może zostać pominięta. Przykład:

```
liczba = 20
wprowadzona_liczba = int(input('Wprowadz liczbę : '))
if wprowadzona_liczba == liczba:
  print('Brawo, to ta liczba.')
else:
  if wprowadzona_liczba < liczba:
    print('Wprowadzona liczba jest mniejsza od {}'.format(liczba))
  else:
    print('Wprowadzona liczba jest wieksza od {}'.format(liczba))
print('Done')
```

## Operatory logiczne

Wyrażenia logiczne możemy łączyć w bardziej skomplikowane wyrażenia zgodnie z regułami logiki:

*and* – operator AND – oba wyrażenia muszą być prawdziwe

*or* – operator OR – przynajmniej jedno z wyrażeń musi być prawdziwe

```
imie = "Jan"
wiek = 23
if imie == "Jan" and wiek == 23:
    print("Nazywasz sie Jan i masz 23 lata.")

if imie == "Jan" or imie == "Robert":
    print("Nazywasz sie Jan lub Robert")
```

*in* - operator sprawdza czy zadana wartość znajduje się innym obiekcie/tablicy

```
imie = "Robert"
if imie in ["Jan", "Robert"]:
    print("Nazywasz sie Jan lub Robert")
```

*not* - zamienia wartość logiczną wyrażenia na przeciwną

**Zadanie 2**

**Zmodyfikuj poniższy kod tak, aby wszystkie wyrażenia zwracały True (były prawdziwe):**

```
# Zmien ponizszy kod
liczba = 10
druga_liczba = 10
pierwsza_tablica = []
druga_tablica = [1,2,3]

if liczba > 15:
  print("1")

if pierwsza_tablica:
  print("2")

if len(druga_tablica) == 2:
  print("3")

if len(pierwsza_tablica) + len(druga_tablica) == 5:
  print("4")

if pierwsza_tablica and pierwsza_tablica[0] == 1:
  print("5")

if not druga_liczba:
  print("6")
```