

Funkcja **print()**, zmienne, konwersja typów, komentarze, funkcja **input()**

1. Funkcja **print**

- Funkcja **print()** jest funkcją wbudowaną. Wypisuje określony komunikat na ekranie
- Aby wywołać funkcję, musisz użyć nawiasu otwierającego i zamykającego, poprzedzając je pożądaną nazwą funkcji. Możesz przypisać argument do funkcji umieszczając go właśnie pomiędzy nawiasami.

- Argumenty musisz rozdzielać przecinkami, np.

```
print("Ala", "ma", "kota", ",", ",", "kot", "wabi", "sie", "'Brutus'")
```

- "Pusta" funkcja **print()** wyświetli na ekranie/w oknie konsoli pustą linię.
- Linie tekstu w języku Python można ograniczać za pomocą cudzysłówów lub apostrofów (cudzysłówów podwójny lub pojedynczy ;)) np.

```
print("Ala ma kota")
print()
print()
print('Ala ma kota')
```

- Umieszczony wewnątrz linii tekstu ukośnik wsteczny:) `\` jest specjalnym znakiem mówiącym o tym, że następujący po nim znak ma specjalne znaczenie, np.

`\n` znak nowej linii(przenosi dalszą część wyniku do nowej linii)

`\t` znak tabulacji

```
print('Ala\n \tma \n\nkota')
print('Ala ma \'kota')
```

- Argumenty przekazywane pozycyjnie to te, których znaczenie określone jest przez ich pozycję, np. drugi argument umiejscowiony jest na wyjściu po pierwszym, trzeci po drugim, itd.
- Argumenty w postaci słów kluczowych to te, których znaczenie nie jest określone przez ich pozycję, ale przez słowa specjalne (słowa-klucze) używane do ich identyfikacji.
- Parametry **end** oraz **sep** mogą być używane do formatowania tekstu na wyjściu funkcji **print()**. Parametr **sep** ustanawia separator, który ma być wyświetlony pomiędzy argumentami

```
print("Ala", "ma", "kota", sep="-")
print("Ala", "ma", "kota", sep="\n")
```

a parametr **end** ustala co ma być wyświetlone na końcu tekstu wyjściowego

```
print("Jak", "nazywa się kot Ali", end=" ??")
print("Jak", "nazywa się kot Ali", sep="\n", end=" ??")
```

Zadanie 1

Przeanalizuj wszystkie powyższe przykłady i zastanów się nad nimi, spróbuj pozmienić, eksperymentuj – to najlepsza forma nauki :) :) :)

Zadanie 2

Wykorzystaj funkcję **print** napisz **jedną** linijkę kodu aby uzyskać oczekiwany wynik. (punkt **a** i **b** to jedna linijka wyniku, punkt **c** to trzy linijki wyniku. Zastanów się, że ten sam efekt można uzyskać na różne sposoby.

Oczekiwany wynik:

- a) Nazywam się Bond 'James Bond'.
- b) Zapoznaj się z feedback'kiem
- c) Twoja ksywa to
 "Tiger"
 znaczy się 'tygrys'

2. Zmienne

- Zmienna to nazwana lokalizacja zarezerwowana do przechowywania wartości w pamięci. Zmienna jest tworzona lub inicjowana automatycznie po przypisaniu jej wartości po raz pierwszy.
- Każda zmienna musi mieć unikalną nazwę - identyfikator. Poprawna nazwa zmiennej musi mieć postać niepustej sekwencji znaków, musi zaczynać się od **liter** lub **podkreślnika** (**_**) i nie może być słowem kluczowym zarezerwowanym przez Pythona (np. słowo kluczowe **for** jest słowem zarezerwowanym). Nazwy zmiennych w języku Python mogą być tworzone z rozróżnieniem wielkich i małych liter, np. zmienne PierwszyNapis i pierwszynapis to dwie różne zmienne.
- W Python nie musisz w nim deklarować zmiennych. Aby przypisać wartości do zmiennych, możesz użyć prostego operatora przypisania w postaci "znaku równości" (=), np.

```
zm1 = 1  
zm2 = "to jest pies"
```

- Można używać złożonych operatorów przypisania (operatorów skrótów), aby modyfikować wartości przypisane do zmiennych, np.

```
zm1 = 2  
zm1 += 3  
zm1 /= 2 * 3
```

- Możesz łączyć tekst i zmienne za pomocą operatora + i używać funkcji print(), aby otrzymać ciągi znaków razem ze zmiennymi

```
s1 = "Bond"  
print("jestem", s1, "James " + s1)
```

Funckja **type()** zwraca typ zmiennej

```
zm1=1  
print(type(zm1))
```

```
zm1=12.11  
print(type(zm1))
```

```
zm1="Ala"  
print(type(zm1))
```

Zadanie 3

Jaki jest wynik poniższego fragmentu kodu – zanim wykonasz koda zastanów się?

```
a) zm1=2
   zm1=3
   print(zm1)
```

```
b) a = '2'
   b = "4"
   print(a + b)
```

```
c) a = 3
   b = 2
   a /= 2 * b
   print(a)
```

3. Komentarze

```
zm1=12      # przypisałem zmiennej zm1 wartość 12
"""
    A to jest komentarz wielolinijkowy
    Czyli blok komentarza
"""
```

4. Konwersja typu, funkcja input

Konwersja

- Funkcja **int()** przyjmuje jeden argument (np. ciąg znaków: `int(string)`) próbuje dokonać jego konwersji na liczbę całkowitą
- Funkcja **float()** przyjmuje jeden argument (np. ciąg znaków: `float(string)`) i próbuje przeprowadzić konwersję tego argumentu na liczbę rzeczywistą
- Funkcja **str()** przyjmuje jeden argument (np. liczbę `str(int)`) i próbuje przeprowadzić konwersję tego argumentu na ciąg znaków.

```
zm1="12"
print(type(zm1))
```

```
zm2=int(zm1)
print(type(zm2))
```

```
zm3=float(zm1)
print(type(zm3))
```

```
zm3/=5
print(zm3)
print(type(zm3))
```

```
print(type(str(zm3)))
print(type(str(zm2)))
```

5. Funkcja input()

- Kiedy funkcja **input()** jest wywoływana, wykonywanie programu zostaje zatrzymane, symbol zachęty miga (zachęca użytkownika do podjęcia działania, gdy konsola przechodzi w tryb wprowadzania), dopóki użytkownik nie wprowadzi danych i/lub nie naciśnie klawisza Enter.

```
imie = input("Jak masz na imię ? ")
print("Witaj " + imie + " Jak się masz")
print()
print("Naciśnij klawisz Enter, aby zakończyć program.")
input()
print("KONIEC.")
```

- Funkcja **input()** **zawsze zwraca ciąg znaków**. Możesz dodawać łańcuchy do siebie nawzajem za pomocą konkatenacji (operator +). Sprawdź ten kod:

```
n1=input("Wprowadź pierwszą liczbę: ")    # Wprowadź 33
n2=input("Wprowadź drugą liczbę: ")       # Wprowadź 44
print(n1 + n2)                           # program zwróci 3344
```

6. Wyrażenia i operatory arytmetyczne

Wyrażenie jest kombinacją wartości, której przypisywana jest wartość, np. 1 + 2.

Operatory są specjalnymi symbolami lub słowami kluczowymi które mogą działać na wartościach i wykonywać operacje, np. operator + dodaje wartości np. 2+3+4

Operatory arytmetyczne w Pythonie: to

+	dodawanie
-	odejmowanie
/	dzielenie - zawsze zwraca liczbę zmiennoprzecinkową
//	dzielenie całkowite - zwraca liczbę wynikającą z podziału, ale zaokrągloną w dół do najbliższej liczby całkowitej
%	modulo - dzieli lewy argument przez prawy argument i zwraca pozostałą część operacji
**	potęgowanie - lewy argument podniesiony do potęgi prawego argumentu

Operator jednoargumentowy posiada tylko jeden argument, np. -2, lub +5.

Operator dwuargumentowy posiada dwa argumenty, np. 5 + 1, lub 11 % 5.

Hierarchia priorytetów:

jednoargumentowe + oraz - posiadają najwyższy priorytet

następnie: **, następnie: *, /, // oraz %,

następnie z najniższym priorytetem: dwuargumentowe + oraz -.

Wyrażenia w nawiasach zawsze są obliczane jako pierwsze, np.

```
9*3 - 2 * (3 * (3 + 2))+3 # Wynik 0.
```

Operator potęgowania używa łączenia prawostronnego, np.

```
2**3**2 #wynik 512
```

Zadanie 3

Jaki jest wynik poniższego fragmentu kodu – zanim wykonasz koda zastanów się?

```
print((2 ** 4), (2 * 3.), (2 * 4))  
print((-2 / 4), (6 / 2), (2 // 4), (-2 // 4))  
print((2 % -4), (2 % 4), (2 ** 3 ** 2))
```

poeksperymentuj z innymi przykładami.