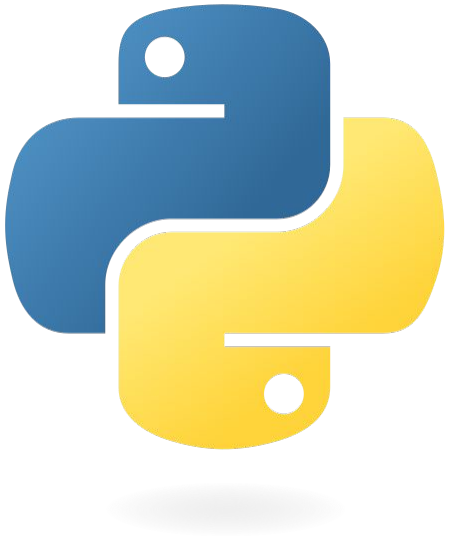
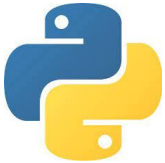


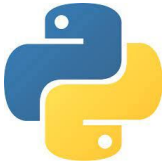
python™

Wprowadzenie



Bibliografia

- https://www.python.org/
- https://www.w3schools.com/python/
- **Algorytmy**
  Maciej M. Sysło, Helion, 2016



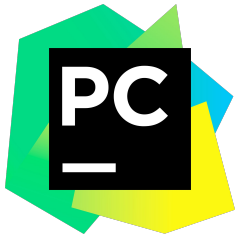
O języku

- język programowania stworzony w 1991 roku przez holenderskiego programistę Guido van Rossum (Google, Dropbox, Microsoft) jako następca języka ABC
- cele Pythona wg van Rossum:
    - łatwy i intuicyjny język, ale jednocześnie równie potężny jak jego konkurenci
    - oparty na zasadzie open source, aby każdy mógł wnieść wkład do jego rozwoju
    - zrozumiały kod w języku angielskim
    - przydatność do rozmaitych codziennych celów, owocująca krótkim czasem programowania
- nazwa pochodzi od serialu komediowego BBC z lat 70 XX wieku “Latający cyrk Monty Pythona”
- język programowania wysokiego poziomu
- język wieloparadygmatowy
- język wieloplatformowy
- język interpretowany
- język typowany dynamicznie

Środowisko programistyczne

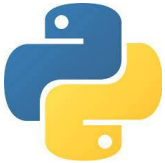
Zintegrowane środowisko programistyczne (ang. *Integrated Development Environment*, IDE) - program służący do tworzenia, modyfikowania i testowania oprogramowania

Pycharm, IntelliJ IDEA od *JetBrains*



Edytory kody źródłowego

Visual Studio Code od *Microsoft*, Notepad++



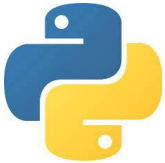
Zmienne

- miejsce w pamięci komputera przechowujące wartość, do którego można odwołać się przy pomocy unikalnego wewnątrz bloku instrukcji identyfikatora (nazwy) (zmienne globalne i lokalne)
- w Python nie ma polecenia do deklarowania zmiennej
- zmienna tworzona jest w momencie przypisania do niej wartości, które może nastąpić:
    - w kodzie poprzez przypisanie konkretnej wartości, wartości innej zmiennej, wyniku wyrażenia - *patrz następne slajdy*
    - w konsoli

```
name=input("Podaj imie ")
print(name)
```

Nazwa (identyfikator) zmiennej

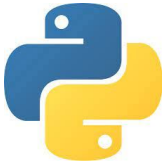
- może składać się z sekwencji znaków alfanumerycznych i podkreślenia
- nie może zaczynać się od liczby
- nie może być słowem kluczowym
- rozróżniane są małe i wielkie litery
- najlepiej w j. angielskim
- sugeruje, co może dana zmienna przechowywać
- format identyfikatora powinien być zgodny z zasadami dobrego stylu programowania w danym języku (konwencją nazewniczą)
    - Camel Case
      ```
      myName = "Marta"
      ```
    - Pascal Case
      ```
      MyName = "Marta"
      ```
    - Snake Case
      ```
      my_name = "Marta"
      ```



Typy danych

Python jest językiem typowanym dynamicznie. Wartości, a nie zmienne, posiadają typ, które do zmiennej przekazywane są podczas przypisania. W związku z czym, zmienne deklarowane są bez typu, ponieważ może on się zmieniać w trakcie wykonywania programu.

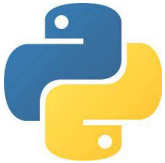
```
x = "Anna" #przypisanie wartosci do zmiennej x
X = 'KOWALSKA'
print(x)
print(x + X)
print(x,X)

print(type(x)) #wyswietlenie typu danych zmiennej x
x = X = 5 #x,X=1,"a"
print("Zmienne x oraz X przechowuja odpowiednio wartosci", x, "i", X, "o typie", type(x), "oraz", type(X))
y = float(x)
print(y, "to typ", type(y))
```



Operatory arytmetyczne

Operatory arytmetyczne są używane z wartościami liczbowymi do wykonywania typowych operacji matematycznych:

OPERATOR	NAZWA	PRZYKŁAD
+	dodawanie	`x + y`
-	odejmowanie	`x - y`
*	mnożenie	`x * y`
/	dzielenie	`x / y`
%	modulo (reszta z dzielenia)	`x % y`
**	potęgowanie	`x ** y`
//	dzielenie bez reszty	`x // y`



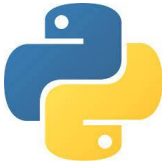
Operatory przypisania

Operatory przypisania służą do przypisywania wartości do zmiennych:

`zmienna ∘ = wyrażenie` to to samo co **`zmienna = zmienna ∘ wyrażenie`**

gdzie ∘ jest operatorem dwuargumentowym (+ , - ,/ ,// ,* ,** ,% ,& ,| ,^ ,>> ,<<), a wyrażeniem może być zmienna lub liczba

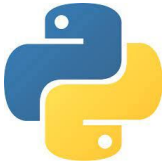
np. `x/=5`, co jest jednoznaczne z zapisem `x=x/5`



Operatory porównania

Operatory porównania służą do porównywania dwóch wartości (w wyniku porównywania dwóch wartości zwracana jest wartość 1(true) lub 0 (false):

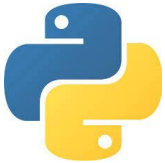
OPERATOR	NAZWA	PRZYKŁAD
==	równe	`x == y`
!=	różne	`x != y`
>	większe	`x > y`
<	mniejsze	`x < y`
>=	większe lub równe	`x >= y`
<=	mniejsze lub równe	`x <= y`



Operatory logiczne

Operatory logiczne służą do łączenia instrukcji warunkowych:

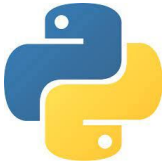
OPERATOR	OPIS	PRZYKŁAD
and	zwraca prawdę, jeśli oba wyrażenia są prawdziwe	`x < 10 and x % 2 == 0`
or	zwraca prawdę, jeśli jedno z wyrażeń jest prawdziwe	`x < 10 or x % 2 == 0`
not	odwraca wynik - zwraca prawdę, jeśli wynikiem jest fałsz	`not(x < 10 and x % 2 == 0)`



Operatory tożsamości

Operatory tożsamości są używane do porównywania obiektów, nie jeśli są równe, ale jeśli w rzeczywistości są tym samym obiektem, z tą samą lokalizacją w pamięci:

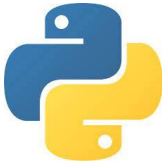
OPERATOR	NAZWA	PRZYKŁAD
is	zwraca prawdę jeśli obie zmienne to te same obiekty	`x is y`
is not	zwraca prawdę jeśli zmienne nie są tymi samymi obiektami	`x is not y`



Operatory członkostwa

Operatory przynależności służą do sprawdzania, czy sekwencja jest prezentowana w obiekcie:

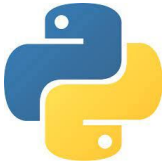
OPERATOR	NAZWA	PRZYKŁAD
in	zwraca prawdę jeśli wartość występuje w obiekcie	`x in y`
not in	zwraca prawdę wartość nie występuje w obiekcie	`x not in y`



Operatory bitowe

Operatory bitowe służą do porównywania (binarnych) liczb:

OPERATOR	NAZWA	OPIS
&	AND, bitowy iloczyn logiczny	ustawia każdy bit na 1, jeśli oba bity to 1
|	OR, bitowa suma logiczna	ustawia każdy bit na 1, jeśli jeden z bitów to 1
^	XOR, bitowa różnica symetryczna	ustawia każdy bit na 1 jeśli tylko jeden z bitów to 1
~	NOT, bitowa negacja	zamienia wartość każdego z bitów z 0 na 1 lub z 1 na 0
<<	przesunięcie bitowe w lewo	przesuń w lewo zaczynając od prawej strony, usuwając bity z lewej strony
>>	przesunięcie bitowe w prawo	odwrotność <<



Styl programowania

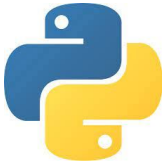
- https://peps.python.org/pep-0008/#introduction - zbiór wytycznych twórców Python dotyczący formatowania kodu w celu ujednolicenia go w skali światowej
- wcięcia (spacje - zazwyczaj 4) na początku używa się do wskazania bloku kodu, a nie tylko dla czytelności jak np. w C++, Java (tutaj do wskazania bloku instrukcji służą klamry)
- brak średnika na końcu instrukcji

C++

```
if (10>1) {
     cout << "10 jest wieksze od 1";
    }
else if (10<1) {
           cout << "10 jest mniejsze od 1";
    }
else {
        cout << "10 i 1 są rowne";
 }
```

Python

```
if 10>1:
    print("10 jest wieksze od 1")
elif 10<1:
     print("10 jest wieksze od
1")
else:
    print("10 i 1 sa rowne")
```



Komentarze

Dobrą praktyką programowania jest stosowanie w kodzie źródłowym komentarzy (najlepiej w j. angielskim) mających na celu:

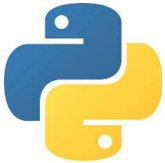
- wyjaśnienie fragmentu kodu
- zwiększenie czytelności kodu
- uniemożliwić wykonanie fragmentu kodu (kompilator/interpreter pomija ten fragment - nie widzi go)

Komentarz jednowierszowy

```
#Printing "Hello, World!"
print("Hello, World!")
```

Komentarz wielowierszowy

```
#Printing
#"Hello, World!"
print("Hello, World!")
"""
print("How big are you?")
print("What is the most beautiful view in the world?")
"""
```



Słowa kluczowe

Słowa mające zastrzeżoną nazwę i specjalnie znaczenie w danym języku programowania. W Python wyróżniamy:

and	False	nonlocal
as	finally	not
assert	for	or
break	from	pass
class	global	raise
continue	if	return
def	import	True
del	in	try
elif	is	while
else	lambda	with
except	None	yield