

Programowanie aplikacji internetowych w Python/Django

Backend - Python

Agnieszka Rudnicka
rudnicka@agh.edu.pl

Python - wstęp

Python - geneza

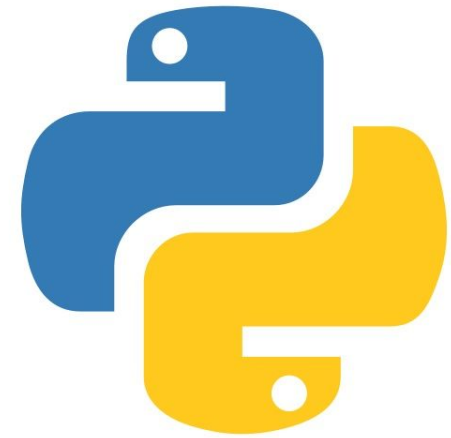
Pierwsza implementacja 1989, Guido van Rossum

"Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office (a government-run research lab in Amsterdam) would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)." Guido van Rossum



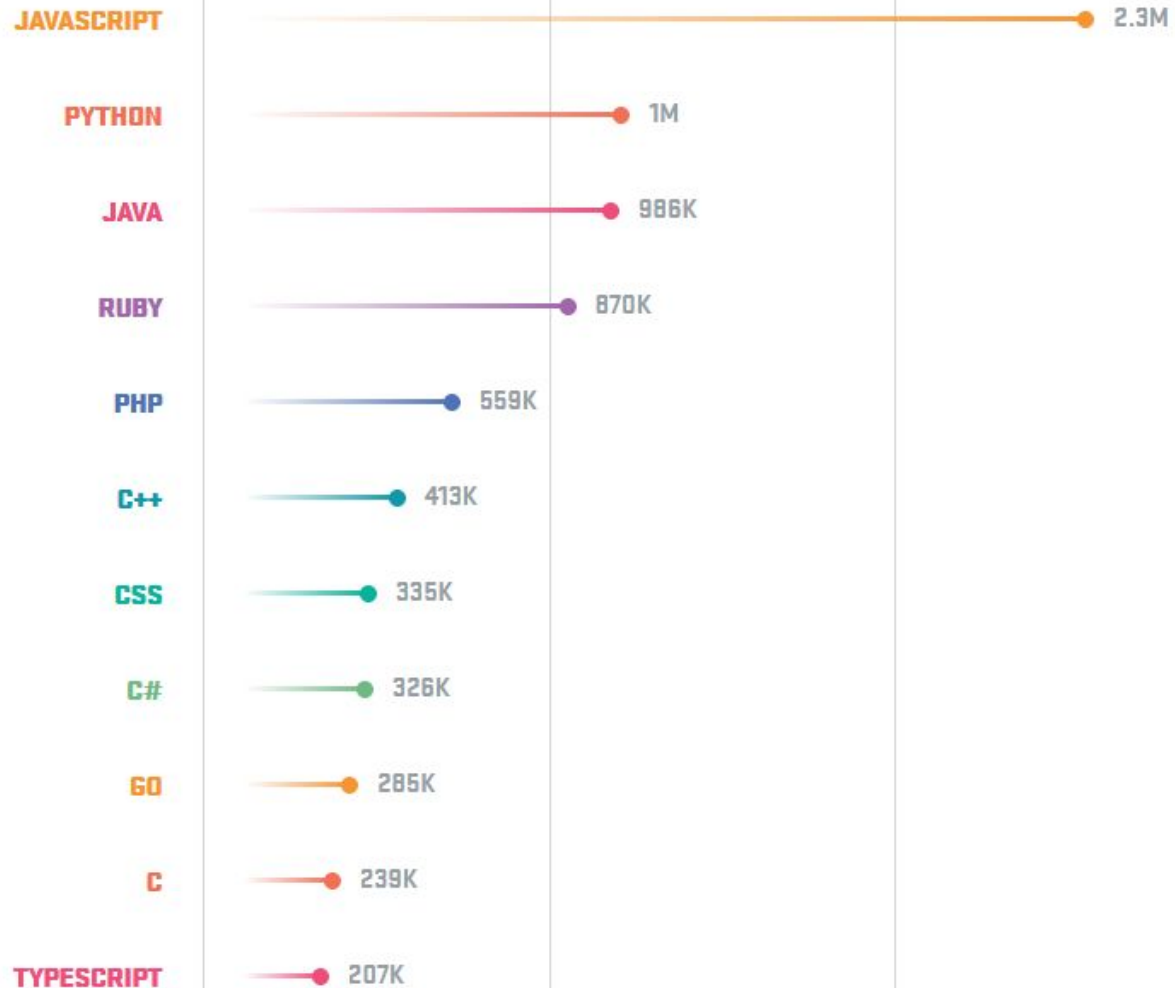
Python - cechy rozpoznawcze

- » wcięcia jako element składni
- » podobny do pseudokodu, przez co czytelny dla osób nieznających języka
- » w pełni obiektowy
- » bardzo wysokiego poziomu
- » dynamiczna typizacja (duck typing)
- » język skryptowy (interpretowany)
- » liberalny interpreter
- » bardzo dużo zasobów: frameworków, bibliotek, tutoriali



Python

TIOBE
index,
Github

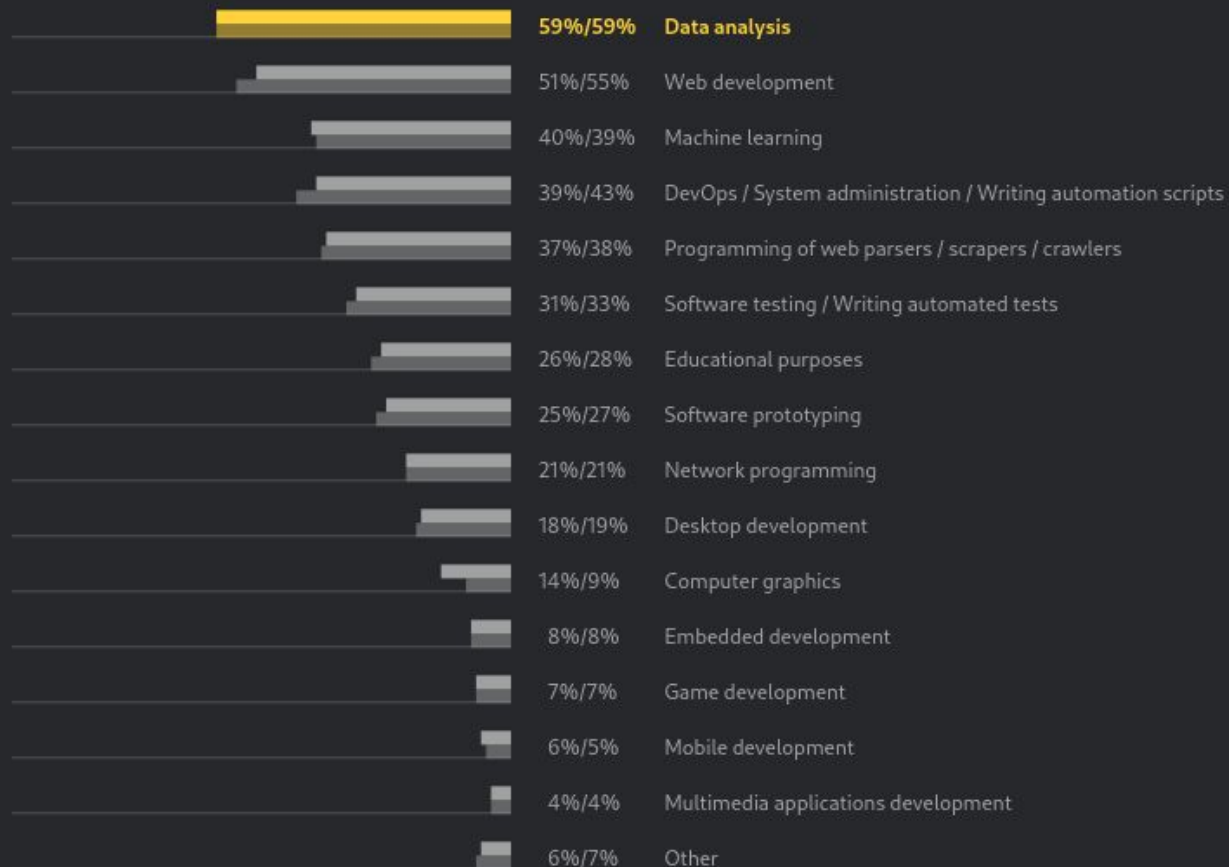


What do you use Python for? > 100%

Main Secondary Combined

• 2019

• 2018



Źródło:

jetbrains.com

Python

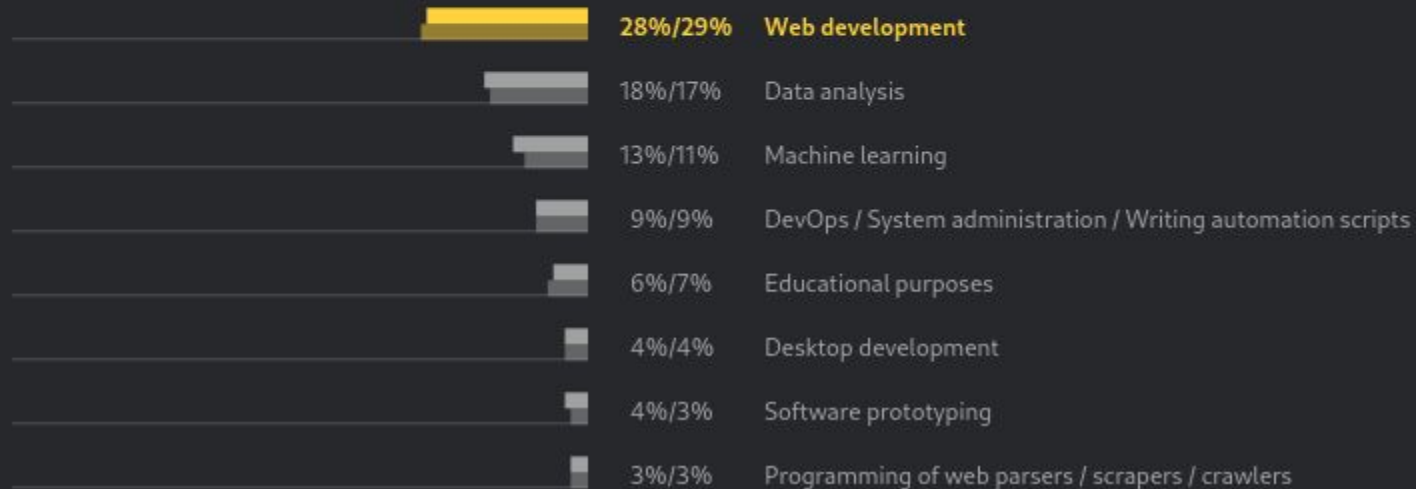
What do you use Python for the most?

> 100%

Main Secondary Combined

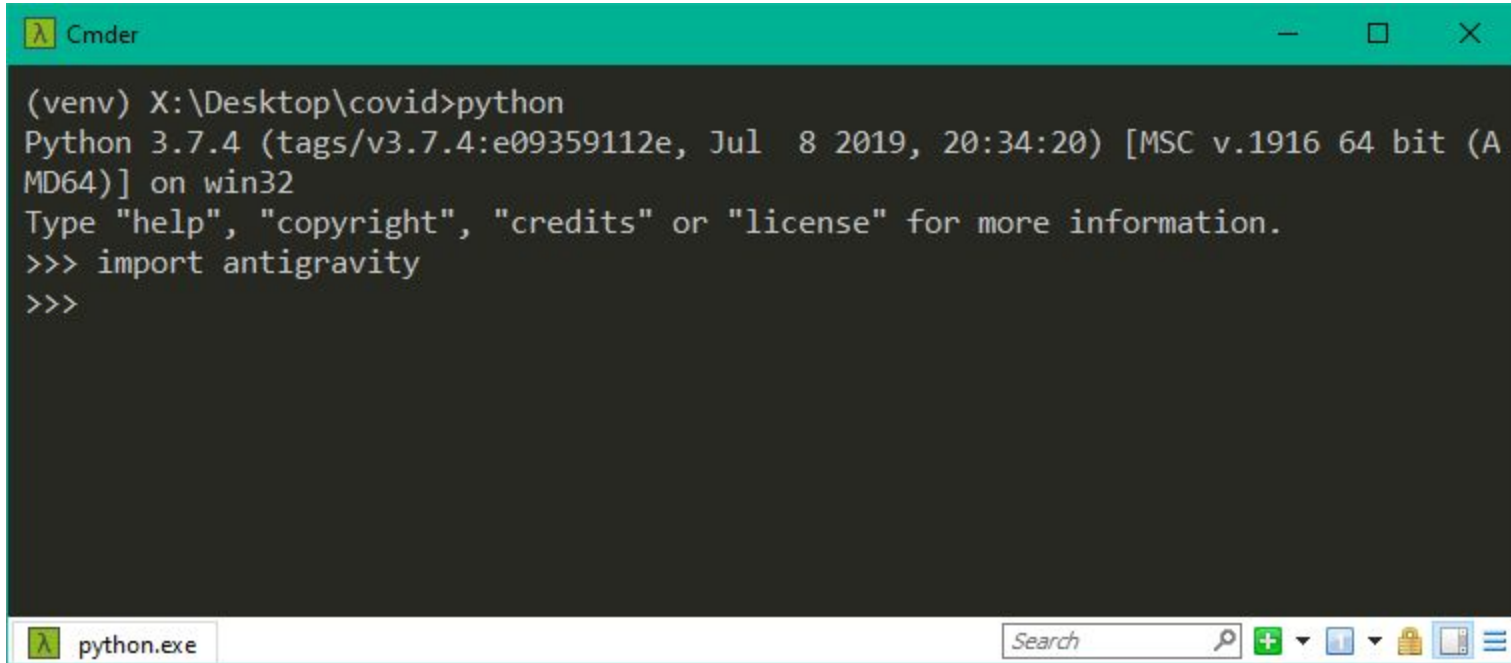
● 2019

● 2018

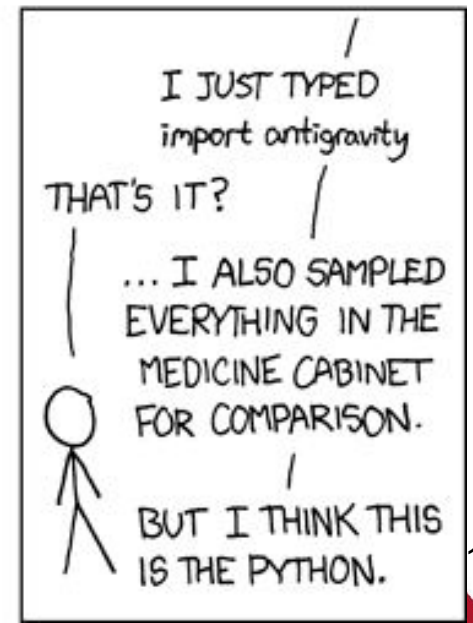
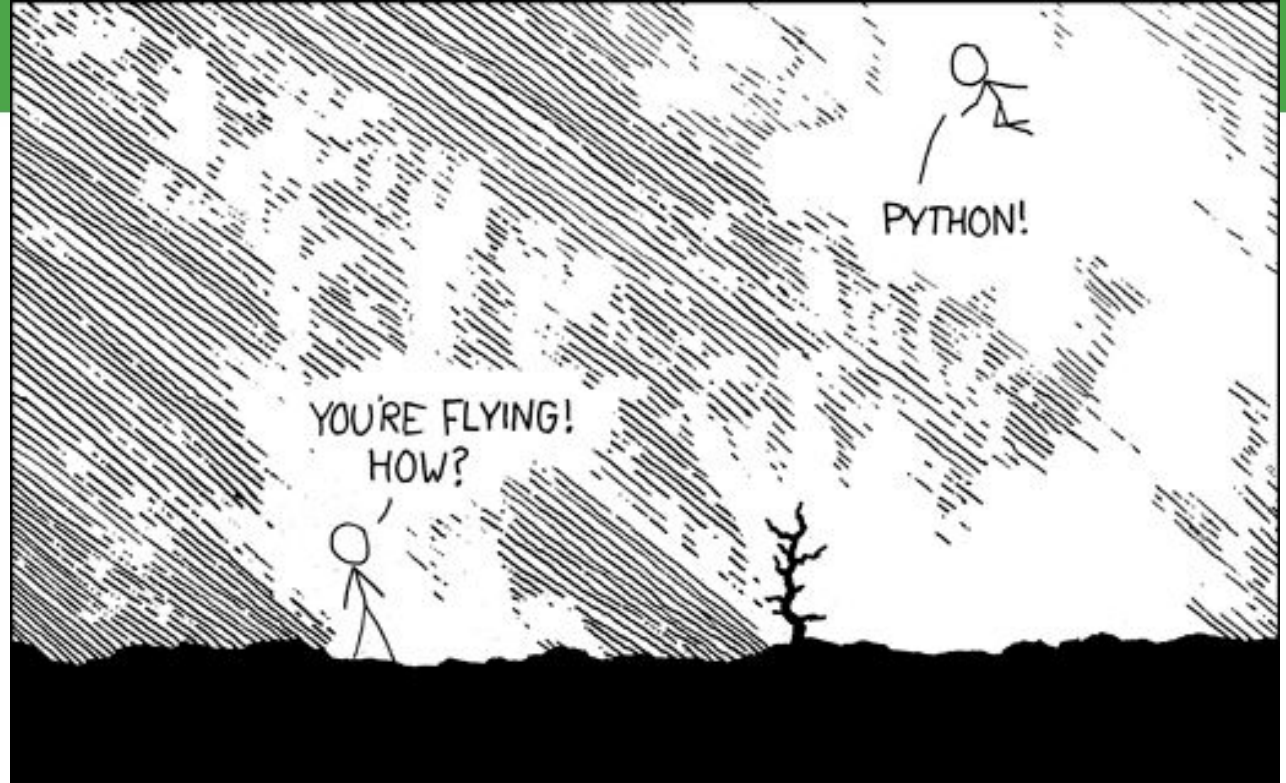


Python - rozgrzewka

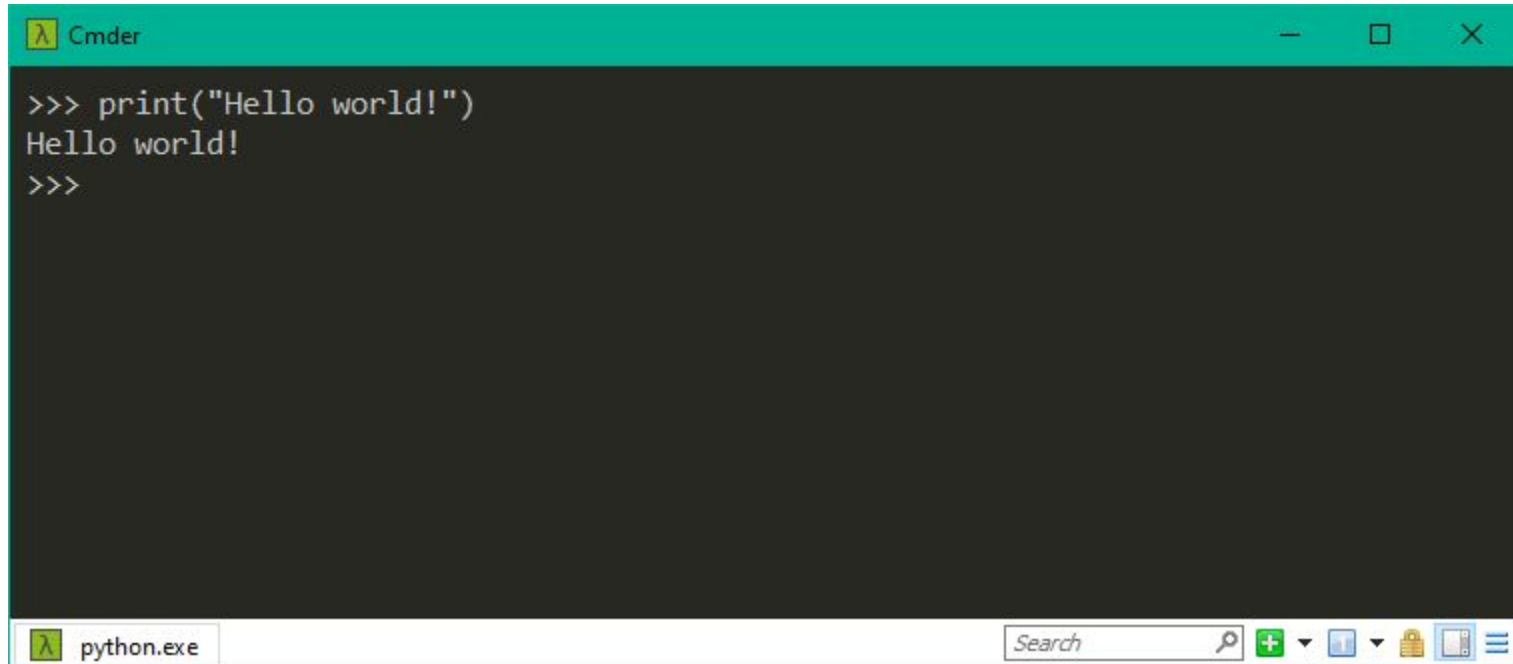
Python - uruchomienie



```
(venv) X:\Desktop\covid>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import antigravity
>>>
```



Python - pierwszy program



```
>>> print("Hello world!")  
Hello world!  
>>>
```

The screenshot shows a Windows Command Prompt window with a teal title bar labeled 'Cmder'. The window has standard minimize, maximize, and close buttons. The command prompt area is black with white text. It shows a Python prompt '>>>' followed by the command 'print("Hello world!")', the output 'Hello world!', and another prompt '>>>'. The taskbar at the bottom shows a taskbar button for 'python.exe' and a search bar with the word 'Search'.

Python - zmienne

Zmienne przypisujemy znakiem równości, nie ma typów!

```
a = 10
```

```
b, c = 20, 30
```

```
d = a + b + c
```

```
print(d)  # co wypisze?
```

```
???
```

Python - typy danych

Napis:

```
color = "yellow" # str
```

Liczby:

```
b, pi = 20, 3.1415 # int, float
```

Lista:

```
marks = [2.0, 3.5, 5.0, 5.0] # list
```

```
print(marks[1]) # wypisze 3.5
```

Słownik: # dict

```
cities = {"Kraków": 771000, "Warszawa": 1777000}
```

```
print(cities["Kraków"]) # wypisze 771000
```

Python - typy prawda/fałsz

Python ewaluuje dane do prawda/fałsz na podstawie zawartości.

Jeśli coś jest puste/ma zerową długość, jest zerem, to będzie fałszem, czyli **False**.

```
bool(0) # zwróci False
```

```
bool(1) # zwróci True
```

```
bool("napis") # zwróci True
```

```
bool("False") # zwróci True
```

```
bool("") # zwróci False
```

```
bool([]) # zwróci False
```

```
if 1: ... # wejdzie do bloku if, ponieważ 1 wyewoluuje do prawdy
```

Python - typ pusty None

Python ma dodatkowo coś co zastępuje null/undefined i inne puste wartości:

None

None w języku Python jest pełnoprawnym obiektem, nie plombą jak w Java/C.

Warto tutaj zauważyć, że ze względu na referencyjność języka typowe NullPointerException są niemożliwe do uzyskania, ponieważ zmienna wskazuje zawsze na jakąś wartość, a None to tylko jedna z możliwości.

Więcej info: <https://realpython.com/null-in-python/>

Python - sprawdzanie warunku

```
x = 10
```

```
if x > 0: # dwukropek jako otwarcie bloku
```

```
    print("x is greater than 0") # wcięcie 4 spacjami!
```

```
elif x == 0:
```

```
    print("x equals 0")
```

```
else:
```

```
    print("too small x")
```

```
print("Koniec") # brak wcięcia - jestem poza blokiem else
```


Python - and/or

Aby uzyskać koniunkcję albo alternatywę warunków można użyć operatorów **and/or**:

```
x = True
```

```
y = False
```

```
bool(x and y) # zwróci False
```

```
bool(x or y) # zwróci True
```

Python - zmiana typu zmiennej

Aby uzyskać zmienną innego typu, np zmienić liczbę na napis używamy konstruktora docelowego typu:

```
x = True
```

```
x = str(x) # zwróci napis "True"
```

```
int(3.14) # zwróci 3
```

```
bool(3) # zwróci True
```

Warto zauważyć, że **nie** mamy tutaj do czynienia z “rzutowaniem” tylko z tworzeniem całkiem nowego obiektu z innego.

Python - pętla for

Po listach, napisach, słownikach można iterować

```
x = "Mój wspaniały napis"
```

```
y = [1, 2, 3, 4, 5]
```

```
for element in y: # element jako zmienna lokalna
```

```
    print(element)
```

```
    if element == 42:
```

```
        break # zakończy pętlę
```

```
# W przypadku iterowania po napisie, co będzie w element?
```

Python - pętla while

Do iterowania można jeszcze użyć pętli while

```
x = "Mój wspaniały napis"
```

```
y = [1, 2, 3, 4, 5]
```

```
flag = True
```

```
while flag: # jak długo bool(flag) jest prawdą, pętla się wykonuje
```

```
    print(element)
```

```
    if element == 42:
```

```
        flag = False # zakończy pętlę, bo warunek będzie fałszem
```

Python - funkcje pomocnicze

Do iterowania przydatne są zwykłe funkcje range, len:

```
x = "Mój wspaniały napis"
```

```
len(x) # zwróci długość napisu x
```

```
y = [1, 2, 3, 4, 5]
```

```
for i in range(10): # będzie zwracać kolejne liczby od 0 do 9
```

```
    pass # instrukcja pusta
```

Oczywiście możliwe są kombinacje `for i in range(len(coś))`, co jest odpowiednikiem `for (int i = 0, i++, i < coś.length())`

Python - operatory pomocnicze

Aby sprawdzić, czy jakaś liczba/litera występuje w napisie/liście:

```
x = "Mój wspaniały napis"
```

```
if "wspe" in x: # in sprawdza zawieranie się w  
    print("wspe is in x")
```

Zaprzeczenie dowolnego operatora/wyrażenia to **not**

```
y = [1, 2, 3, 4, 5]
```

```
if 3 not in y: # not in - zaprzeczenie zawierania się  
    print("3 is missing from this list")
```

```
bool(not False) # jest prawdą :)
```

Python - funkcje

```
def super_function(x, y):  
    print("Got x =", x, "and y =", y)  
    return x + y  
  
super_function(5, 10) # zwróci 15  
super_function(y=10, x=5) # zwróci 15
```

```
def super_function(x=10, y=20): # domyślne argumenty  
    print("Got x =", x, "and y =", y)  
    return x + y  
  
super_function() # zwróci 30
```

Python - materiały pomocnicze

- » [Zanurkuj w Pythonie](#) - klasyk jak chodzi o materiały do Pythona
- » [Python 101](#) - materiały wprowadzające do języka w języku Polskim
- » [The Hitchhiker's Guide to Python](#) - jeden z lepszych materiałów wprowadzających w ekosystem Pythona
- » [PEP 8 — the Style Guide for Python Code](#) - "jak kodzić, żeby inni się doczytali"
- » [Real Python](#) - blog z ogromną ilością nieźle zredagowanych tutoriali i poradników
- » [Awesome Python](#) - moderowana lista ciekawych projektów związanych z językiem Python
- » [Django Girls](#) - tutorial do pythona i potem django
- » [Python Cheat-Sheet](#)