



Zaplanuj swoją podróż z Pythonem

infoShare 2019

Agenda

// 17:30 - 19:30 - podstawy Python

// 19:30 - 19:45 - przerwa

// 19:45 - 21:30 - aplikacje w Python



Podstawy Python

Python

// interpretowany - linijka po linijce

// obiektowy

// wszechstronny

// popularny - (Big Data, Web, Blockchain, AI, devops,
hacking, computer vision...)

Tworzenie kodu

// interpreter

// zwykły notatnik

// pliki tekstowe .py

// IDE - dodatkowa funkcjonalność (podpowiedzi,
kolorowanie składni, debugger, testy)

// Python IDLE, PyCharm (CE), VS Code, Sublime, Atom

Uruchamianie kodu

// interpreter (python/python.exe)

// konsola / terminal / wiersz polecenia

// można mieć kilka wersji Pythona

// IDE umożliwiają uruchamianie bezpośrednio

// nie zawsze program zadziała bez IDE

Typy danych

123 - **int** – liczby całkowite

54.45 - **float** – liczby zmiennoprzecinkowe

"Ala" - **str** – łańcuchy znaków (string)

True/False - **bool** – prawda fałsz

None

listy, słowniki, tuple, pliki, własne typy (klasy)

Zmienna

// nazwany obszar pamięci, w którym znajduje się jakaś wartość

// pozwala na ponowne użycie wartości w innym miejscu w kodzie

```
moja_liczba = 124  
nazwisko = "Kowalski"  
czy_obecny = True
```


Operator

Matematyczne:

$+$, $-$, $*$, $/$, $//$, $\%$, $**$

Logiczne:

$==$, $!=$, $<$, $>$, $<=$, $>=$

`in`, `is`, `and`, `or`, `not`

Operator



obliczane jest wyrażenie **po prawej** stronie znaku,
następnie wartość jest przypisywana do zmiennej po
lewej stronie znaku

Atrybuty wbudowane typów

Każdy typ danych posiada zdefiniowane atrybuty (metody i pola), które pozwalają na wykonanie różnych (najpopularniejszych) działań.

`typ.metoda()`

`"ala ma kota".capitalize()`

String (łańcuch znaków)

```
nazwisko = "Kowalski"
```

```
# długość
```

```
len(nazwisko) -> 8
```

```
# Indeksowanie
```

```
nazwisko[0] -> K
```

```
nazwisko[3] -> a
```

```
nazwisko[8] -> błąd, nie ma takiego indeksu!
```

int - float - str

5 - int - liczba całkowita

65.987 - float - liczba zmiennoprzecinkowa

'45' - str - łańcuch znaków

"3434.434" - str - łańcuch znaków

instrukcja warunkowa

if (warunek):

jakiś kod wykonany gdy warunek prawdziwy

elif (inny warunek):

kod wykonany gdy warunek w if był fałszywy

warunek w tym elif musi być prawdziwy aby ten kod wykonać

elif (inny warunek):

elif-ów może być wielu. lub żadnego, kod wew. elif

wykona się tylko gdy wszystkie wyższe warunki były fałszywe

else:

przypadek domyślny, tu nie sprawdzamy warunku, kod w else

będzie wykonany gdy wszystkie w if- elif były fałszywe

else może być tylko jeden lub wcale

Tablica logiczna

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

import

import moduł

from moduł **import** funkcja

from moduł **import** *

string, datetime, copy, math, decimal, random, os, csv,
antigravity

list() []

```
lista = [1, 2, 3]
```

```
lista2 = ["kwiatek", "doniczka", "ziemia", "woda"]
```

```
lista3 = []
```

```
lista4 = [1, "dwa", 3, 4]
```

```
lista5 = list(range(2,5))
```

Możemy indeksować, slice'ować

Do elementu odwołujemy się przez indeks

tuple() ()


Tuple jest typem niezmiennym – raz zdefiniowanego nie można zmienić

```
tuple1 = ("raz", "dwa", "trzy")
```

```
tuple1[0] = "jeden" – spowoduje błąd
```

```
x = "raz",
```

```
y = "raz", dwa"
```



dict() { }

{klucz : wartość}

klucz – musi być typem niezmiennym (string, tuple, liczba), musi być unikalny (tylko jeden w słowniku)

wartość – mogą być powtórzone

Odwołujemy się poprzez klucz a nie indeks!!!

pętla for

for element in kolekcja:
możemy użyć element
...

for wykona się tyle razy ile elementów jest w kolekcji*

* chyba, że pętla zostanie przerwana lub zmodyfikowana

pliki

otwieramy plik

```
plik = open("ścieżka_do_pliku", 'tryb')
```

tryby:

r – tylko do odczytu

w – zapisywanie pliku (stary plik o tej samej nazwie będzie usunięty)

r+ - do odczytu i zapisu

a – dopisywanie do pliku (dane są dopisane do końca istniejącego pliku)

pliki tekstowe odczyt

plik.read() – odczytanie całego pliku, zwracany jest string zawierający cały tekst pliku (włącznie ze znakami \n) – opc. argument – int określająca ilość bajtów do wczytania

plik.readline() – odczytanie jednej linii z pliku, zwracany jest string z linijką testu, włącznie ze znakiem \n

plik.readlines() – odczytuje cały tekst – zwraca listę stringów - linijek

```
for line in plik:  
    print(line, end="")
```

pliki tekstowe zapis

plik.write(string) – zapisuje string do pliku w obecnej pozycji kursora, zwraca liczbę zapisanych znaków – należy pamiętać o znaku \n

plik.writelines(iterable) – zapisuje elementy z kolekcji jako poszczególne linie w pliku

Plik musi być otworzony w trybie do zapisu aby móc go zmieniać!

funkcje - definiowanie

definiowanie:

```
def do_nothing():  
    pass
```

wywołanie:

```
do_nothing()
```


funkcje - argumenty

```
def do_nothing():  
    pass
```

nie ma argumentów

```
def do_nothing(x):  
    pass
```

jeden argument

```
def do_nothing(x, y, z):  
    pass
```

wiele argumentów

funkcje - zwracanie wartości

```
def print_square(x)  
    print(x**2)
```

```
def give_square(x)  
    return x**2
```

aby użyć funkcję zwracającą obiekt należy ten obiekt zapisać w zmiennej

```
>>> wynik = give_square(3)  
>>> print(wynik)  
9
```

funkcje - argumenty domyślne

```
def do_nothing(x, y=10):  
    pass
```

```
def do_nothing(x, y, z=12, w = „Ola”):  
    pass
```

```
def do_nothing(y=10):  
    pass
```

**argumenty domyślne muszą być po argumentach
wymaganych**

funkcje - argumenty domyślne

```
def do_something(x, y, z=12, w =„Ola”):  
    pass
```

```
>>> do_something(1)          <- błąd, zbyt mała liczba argumentów  
>>> do_something(1, 23)  
>>> do_something(1, 2, "trzy")  
>>> do_something(1, 2, 34, "ola")  
>>> do_something(1, 33, w="ola")
```



Workshop



Dziękuję

infoShare 2019