Wizualizacja danych - wykład 2

dr Piotr Jastrzębski

Wprowadzenie do języka Python - kontynuacja

Podstawowe typy danych w Pythonie

- liczby (`int`, `float`, `complex`): `22`, `5.2`, `2j + 9`
- łańcuchy znaków (`str`): `'tekst1'`, `"tekst2"`
- lista (`list`): `[3, 22, 'tekst', False]`
- krotka (`tuple`): `(6, 17, 'tekst', False)`
- słownik (`dict`): `['klucz': 'wartość', 23: 33, 'status': False]`
- typ logiczny (`bool`): `True`, `False`

Liczby

```
print(type(5))
```

```
## <class 'int'>
```

```
print(type(4.5))
```

```
## <class 'float'>
```

```
print(type(55+3j))
```

```
## <class 'complex'>
```

```
print(type(4e+4))
```

```
## <class 'float'>
```

```
print(type(40000))
```

```
## <class 'int'>
```

Łańcuchy znaków - stringi

```
str = 'Hello World!'

print(str)
```

```
## Hello World!
```

```
print(str[0])
```

```
## H
```

```
print(str[2:5])
```

```
## llo
```

```
print(str[2:])
```

```
## llo World!
```

```
print(str * 2)
```

```
## Hello World!Hello World!
```

```
print(str + 'WMII')
```

```
## Hello World!WMII
```

Podstawowa instrukcja wyjścia - funkcja `print`

Składnia wg dokumentacji:

```
print(*objects, sep=' ', end='\n', file=sys.stdout,
      flush=False)
```

- `objects` - to co ma być wyświetlone
- `sep` - separator, domyślnie znak spacji
- `end` - co co ma być wyświetlone na końcu, domyślnie znak końca linii
- `file` - określa gdzie mają być `objects` wyświetlone, domyślnie `sys.stdout` (domyślny ekran)
- `flush`- określa czy "wyjście" ma być buforowane przed przekazaniem do `file`, domyślne `False`

```
print(1, 2, 3, 4)
```

```
## 1 2 3 4
```

```
print(1, 2, 3, 4, sep='*')
```

```
## 1*2*3*4
```

```
print(1, 2, 3, 4 ,sep='#', end='&')
```

```
## 1#2#3#4&
```

```
print('x', 'y', 'z', sep='', end='')
print('a', 'b', 'c' , sep='', end='')
```

```
## xyzabc
```

```
print('a', 'b', '\n', 'c')
```

```
## a b
##  c
```

\t - przesunięcie do następnego “tab”=8 spacji

```
print('sdf', 3456, -2, sep='\t')
```

```
## sdf  3456    -2
```

\r - przesunięcie do lewej strony po każdym wyświetleniu

```
print(345, 'y', 'abc', sep='\r')
```

```
## abc
```

```
a = 16
b = 2.25
z = 45
print('{:5d} {:6.3f} {:10d}'.format(a, b, z))
```

```
##    16  2.250         45
```

```
print('{2:5d} {1:6.3f} {0:10d}'.format(a, b, z))
```

```
##    45  2.250         16
```

Input - podstawowe wejście

```
name= input('Podaj imię \n')
print('typ:', type(name))
```

```
Podaj imię
Jan
typ: <class 'str'>
```

```
number= int(input('Podaj liczbę \n'))
print('typ:', type(number))
```

```
Podaj liczbę
32
typ: <class 'int'>
```

Operacje arytmetyczne

```
print(5+3)
```

```
## 8
```

```
print(4*5.2)
```

```
## 20.8
```

```
print(9-7)
```

```
## 2
```

```
print(25%7)
```

```
## 4
```

```
print(4/5)
```

```
## 0.8
```

```
print(4//5)
```

```
## 0
```

```
print(4/5.0)
```

```
## 0.8
```

```
print(4//5.0)
```

```
## 0.0
```

```
print(3**0)
```

```
## 1
```

```
print(0**0)
```

```
## 1
```

```
print(4/0)
```

```
ZeroDivisionError: division by zero
```

Operacje na stringach

```
print('raz'+' '+'dwa')
```

```
## raz dwa
```

```
print('tekst'*3)
```

```
## teksttekstttekst
```

Operatory przypisania

- = standardowy
- +=, -=, *=, /=, %=. **=, //=

```
a = 5
a += 1
print(a)
```

```
## 6
```

```
a **= 2
print(a)
```

```
## 36
```

Operaratory porównania

Znak	Znaczenie	Przykład
>	większe niż	`x > y`
<	mniejsze niż	`x < y`
==	równa się	`x == y`
!=	nie równa się	`x != y`
>=	większe lub równe	`x >= y`
<=	mniejsze lub równe	`x <= y`

Operatory logiczne i typ logiczny

Operator	Znaczenie	Przykład
`and`	i	`x and y`
`or`	lub	`x or y`
`not`	negacja	`not x`

```
a = True
b = False
print(a or b)
```

```
## True
```

```
print(type(a))
```

```
## <class 'bool'>
```

Kolejność operatorów

Od ostatniego:

- `lambda`
- `if - else`
- `or`
- `and`
- `not x`
- `in, not in, is, is not, <, <=, >, >=, !=, ==`
- `|`
- `^`
- `&`
- `<<, >>`
- `+, -`
- `*, @, /, //, %`
- `+x, -x, ~x`

- `**`
- `await x`
- `x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`
- `(expressions...)`, `[expressions...]`, `{key: value...},{expressions...}`

Źródło: https://docs.python.org/3/reference/expressions.html#operator-precedence.

Listy

Listy w Pythonie mogą przechowywać elementy różnych typów.

```
list1 = ['raz', 'dwa', 5, 5];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"];
print(list3)

## ['a', 'b', 'c', 'd']
```

```
list4 = ['s', 'ww', True, 5]
print(list4[3])
```

```
## 5
```

```
list4[1] = True
print(list4[1])
```

```
## True
```

```
print(list4[-1])
```

```
## 5
```

```
print(list4[2:])
```

```
## [True, 5]
```

```
print(len([2, 3, 4]))
```

```
## 3
```

```
print([1, 2, 3] + [4, 5, 6])
```

```
## [1, 2, 3, 4, 5, 6]
```

```
print(['Hi!'] * 4)
```

```
## ['Hi!', 'Hi!', 'Hi!', 'Hi!']
```

```
print(3 in [1, 2, 3])
```

```
## True
```

Pytanie do przemyślenia na kolejny wykład

Co oznacza w Pythonie, że wartości przekazywane są przez referencję?

```
a = 5
b = a
b += 2
print(a)
```

```
## 5
```

```
print(b)
```

```
## 7
```

```
list1 = [1, 2, 3, 4]
list2 = list1
list1[2] = 'a'
print(list1)
```

```
## [1, 2, 'a', 4]
```

```
print(list2)
```

```
## [1, 2, 'a', 4]
```

Instrukcje warunkowe

```
a = 5
if a > 0:
    print('liczba dodatnia')
elif a == 0:
    print('zero')
else:
    print('liczba ujemna')

## liczba dodatnia
```

Pętle

```
words = ['kot', 'pies', 'chomik']
for w in words:
    print(w, len(w))
```

```
## kot 3
## pies 4
## chomik 6
```

```
i = None
for i in range(2):
    print(i)
```

```
## 0
## 1
```

```
i = 1
j = 1
while i < 4:
    j = 1
    while j < 4:
        print(i, '*', j, '=', i * j)
        j += 1
    i += 1
```

Bibliografia

- https://pl.wikipedia.org/wiki/Python, dostęp online 12.02.2019.
- https://bulldogjob.pl/news/264-java-php-ruby-jak-wlasciwie-wymawiac-nazwy-technologii. dostęp online 12.02.2019.
- https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html, dostęp online 14.02.2019.
- K. Ropiak, Wprowadzenie do języka Python, http://wmii.uwm.edu.pl/~kropiak/wd/Wprowadzenie%20do%20j%C4%99zyka%20Python.pdf, dostęp online 14.02.2019.
- B. Slatkin, Efektywny Python. 59 sposobów na lepszy kod, Helion 2015.
- https://www.python.org/dev/peps/pep-0008/, dostęp online 14.02.2019.

Bibliografia - cd2

- https://www.flynerd.pl/2017/05/python-4-typy-i-zmienne.html, dostęp online 14.02.2019.
- http://pytolearn.csd.auth.gr/p0-py/01/print.html, dostęp online 15.02.2019.
- https://www.tutorialspoint.com/python3/python_lists.htm, dostęp online 17.02.2019.