Wprowadzenie do języka Python - wykład 3

Dokończenie list

`list.sort()` - sortuje listę (o ile elementy można posortować)

```
a = ['abc','xyz','abc','efg']
a.sort()
print(a)
```

```
## ['abc', 'abc', 'efg', 'xyz']
```

`list.reverse()` - odwraca kolejność elementów na liście (nie ma nic związku z sortowaniem!)

```
a = [4, 5, -2, 7.3, 9, -22, 23]
a.reverse()
print(a)
```

```
## [23, -22, 9, 7.3, -2, 5, 4]
```

`list.copy()` - tworzy kopię listy

Spójrzmy na przykład jak działa operator przypisania dla list.

```
a = [4, 5, -2, 7.3, 9, -22, 23]
b = a
b[2] = 100
print(b)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

```
print(a)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

Różnica z użyciem copy.

```
a = [4, 5, -2, 7.3, 9, -22, 23]
b = a.copy()
b[2] = 100
print(b)
```

```
## [4, 5, 100, 7.3, 9, -22, 23]
```

```
print(a)
```

```
## [4, 5, -2, 7.3, 9, -22, 23]
```

Lista jako stos

```
stack = [3, 4, 5, 8, 9]
stack.append(6)
stack.append(7)
print(stack)
```

```
## [3, 4, 5, 8, 9, 6, 7]
```

```
print(stack.pop())
```

```
## 7
```

```
print(stack)
```

```
## [3, 4, 5, 8, 9, 6]
```

Lista jako kolejka

```
from collections import deque

queue = deque(["aw", "tg", "kj"])
queue.append("gg")
print(queue)
```

```
## deque(['aw', 'tg', 'kj', 'gg'])
```

```
print(queue.popleft())
```

```
## aw
```

```
print(queue)
```

```
## deque(['tg', 'kj', 'gg'])
```

List Comprehensions

```
squares = []
for x in range(5):
    squares.append(x ** 2)

print(squares)
```

```
## [0, 1, 4, 9, 16]
```

```
squares = [x**2 for x in range(5)]
print(squares)
```

```
## [0, 1, 4, 9, 16]
```

Krotka - tuple

```
krotka = 123, 'abc', True
krotka2 = (123, 'abc', True)
print(krotka[2])
```

```
## True
```

```
krotka[0] = 1
```

```
## TypeError: 'tuple' object does not support item assignme
```

https://docs.python.org/3.8/library/stdtypes.html#tuple

Zbiór - set

```
cyfry = {'raz', 'dwa', 'raz', 'trzy', 'raz', 'osiem'}
print(cyfry)
```

```
## {'dwa', 'trzy', 'raz', 'osiem'}
```

https://docs.python.org/3.8/library/stdtypes.html#set

Słownik

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
print(tel)
```

```
## {'jack': 4098, 'sape': 4139, 'guido': 4127}
```

```
tel['jack']
```

```
## 4098
```

```
del tel['sape']
tel['irv'] = 4127
print(tel)
```

```
## {'jack': 4098, 'guido': 4127, 'irv': 4127}
```

https://docs.python.org/3.8/library/stdtypes.html#mapping-types-dict

Napisy

- trochę podobne do listy
- typ sekwencyjny do przechowywania znaków, ale w odróżnieniu od listy jest niezmienny
- w języku Python nie ma oddzielnego typu znakowego
- apostrofy i cudzysłów można stosować zamiennie, ale konsekwentnie

Inne nazwy: - string, napisy, łańcuchy znaków

Abstrakcyjnie:

- na końcu każdego napisu jest znak “zerowy” - będzie widać lepiej w C++

Tablica znaków ASCII
https://upload.wikimedia.org/wikipedia/commons/5/5c/ASCII-Table-wide.pdf

```
a = "Olsztyn"
print(a)
```

```
## Olsztyn
```

```
print(a[3])
#a[2] = 'w'
```

```
## z
```

```
a = "Olsztyn"
b = "Gdańsk"
print(a + b)
```

```
## OlsztynGdańsk
```

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

```
## OlsztynOlsztyn
```

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

```
## OlsztynOlsztyn
```

Specjalne funkcje

- `chr()` zamienia liczbę całkowitą na znak
- `ord()` zamienia znak na liczbę całkowitą odpowiadającą pozycji w tabeli znaków
- `len()` - długość napisu
- `str()‘` - rzutuje argument na napis

Porządek leksykograficzny

Mądra definicja z wikipedii:

Relację leksykograficzną $\preccurlyeq$ między ciągami $\alpha, \beta \in X^*$ ustala się następująco:

- jeśli istnieje wskaźnik j taki, że $\alpha(j) \neq \beta(j)$, to znajdujemy najmniejszy i o tej własności. Wówczas
  - $\alpha \preccurlyeq \beta$ gdy $\alpha(i) \preccurlyeq \beta(i)$ lub $\beta \preccurlyeq \alpha$ gdy $\beta(i) \preccurlyeq \alpha(i)$ (tzn. relacja między ciągami jest zgodna z relacją między odpowiednimi elementami)
- jeśli taki j nie istnieje, to
  - jeśli oba są skończone i tej samej długości, to $\alpha = \beta$
  - jeśli oba ciągi są nieskończone, to $\alpha = \beta$
  - jeśli są różnej długość np. β jest dłuższy od α (w szczególności β może być nieskończony), to $\alpha \preccurlyeq \beta$

Przykłady:

```
print("A" < "a")
```

```
## True
```

```
print("Abc" < "aTw")
```

```
## True
```

```
print("vccx" < "123")
```

```
## False
```

```
print("ABC" < "AB")
```

```
## False
```

```
print("AB" < "ABC")
```

```
## True
```

Fomatowanie napisów

- trzy różne konwencje
- niektóre rzeczy nie działają w każej wersji 3.x
- warto zastanowić się czy warto używać tych konstrukcji? czasem może lepiej skorzystać z funkcji print?

styl printf

Zaczerpnięty z języka C - stare.

https://docs.python.org/3.9/library/stdtypes.html#old-string-formatting

```
a = "abc"
str = "a to %s" % a
print(str)
```

```
## a to abc
```

```
b = 4
c = 5
str2 = "%d + %d = %d" % (b, c, b + c)
print(str2)
```

```
## 4 + 5 = 9
```

Dodatkowe:

https://gist.github.com/pjastr/02d01dba3d5f5c3e60ed74cb32c913ed
https://gist.github.com/pjastr/cbe8418eb4798b92d7fcba4f48d32845
https://gist.github.com/pjastr/e7d5fcbebd578c1df122d307e0051707
https://gist.github.com/pjastr/00c1df223918f975d678d8455b4f5b0a

styl format

https://docs.python.org/3.9/library/string.html#formatstrings

```
a = "abc"
str = "a to {}".format(a)
print(str)
```

```
## a to abc
```

```
b = 4.2
c = 5
str2 = "{0} + {1} = {2}".format(b, c, b + c)
print(str2)
```

```
## 4.2 + 5 = 9.2
```

```
b = 4.2
c = 5
str2 = "{0:f} + {1:d} = {2:e}".format(b, c, b + c)
print(str2)
```

```
## 4.200000 + 5 = 9.200000e+00
```

Dodatkowe przykłady:

https://docs.python.org/3.9/library/string.html#format-examples
https://gist.github.com/pjastr/d42d937c7b00b80b5dfe309b4ac0e854
https://gist.github.com/pjastr/90f1accf7f54d8c74ac036d59a24a9dd
https://gist.github.com/pjastr/adfd7371dcbe4e4034bfb12dcfe30129
https://gist.github.com/pjastr/2980fb68a484dcb5ff595774eec4195c

f-Strings

https://docs.python.org/3.9/reference/lexical_analysis.html#f-strings

```
a = "abc"
str = f"a to {a}"
print(str)
```

```
## a to abc
```

```
b = 4.2
c = 5
str2 = f"{b} + {c} = {b+c}"
print(str2)
```

```
## 4.2 + 5 = 9.2
```

```
b = 4.2
c = 5
str2 = f"{b:f} + {c:d} = {b+c:e}"
print(str2)
```

```
## 4.200000 + 5 = 9.200000e+00
```

Dodatkowe

- podział stałych https://docs.python.org/3.9/library/string.html?highlight=string#module-string
- funkcje wbudowane dot. napisów https://docs.python.org/3.9/library/stdtypes.html#string-methods

Funkcje

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

```
def printme(str):
    """Funkcja wyświetlająca string"""
    print(str)
    return


printme("abc")
```

```
## abc
```

```
print(printme.__doc__)
```

```
## Funkcja wyświetlająca string
```

Przekazywanie przez referencję

```
def changeme(lista):
    print("Przed zmianą: ", lista)
    lista[2] = 50
    print("Po zmianie: ", lista)
    return


mylist = [10, 20, 30]
changeme(mylist)
```

```
## Przed zmianą:  [10, 20, 30]
## Po zmianie:  [10, 20, 50]
```

```
print("Poza funkcją: ", mylist)
```

```
## Poza funkcją:  [10, 20, 50]
```

```
def changeme(lista):
    lista = [2, 3, 4]
    print("Wewnątrz funkcji: ", lista)
    return


lista = [10, 20, 30]
changeme(lista)
```

```
## Wewnątrz funkcji:  [2, 3, 4]
```

```
print("Poza funkcją: ", lista)
```

```
## Poza funkcją:  [10, 20, 30]
```

```
def changeme():
    global lista
    lista = [2, 3, 4]
    print("Wewnątrz funkcji: ", lista)
    return


changeme()
```

```
## Wewnątrz funkcji:  [2, 3, 4]
```

```
print("Poza funkcją: ", lista)
```

```
## Poza funkcją:  [2, 3, 4]
```

Obowiązkowy argument

```
def printme(str):
    print(str)
    return


printme()
```

```
## TypeError: printme() missing 1 required
positional argument: 'str'
```

Keyword argument

```
def kwadrat(a):
    return a*a


print(kwadrat(a=4))
```

```
## 16
```

Domyślny argument

```
def sumsub(a, b, c=0, d=0):
    return a - b + c - d


print(sumsub(12, 4))
```

```
## 8
```

```
print(sumsub(3, 4, 5, 7))
```

```
## -3
```

```python
def srednia(first, *values):
    return (first + sum(values)) / (1 + len(values))


print(srednia(2, 3, 4, 6))
```

```
## 3.75
```

```python
print(srednia(45))
```

```
## 45.0
```

```
def f(**kwargs):
    print(kwargs)


f()
```

```
## {}
```

```
f(pl="Polish", en="English")
```

```
## {'pl': 'Polish', 'en': 'English'}
```

Funkcje matematyczne

Link do dokumentacji
https://docs.python.org/3/library/math.html

```
import math

a=0
b=math.sin(2*math.pi)
print(b)
```

```
## -2.4492935982947064e-16
```

```
print(math.isclose(a,b, rel_tol=1e-09, abs_tol=1e-09))
```

```
## True
```