

Konsultacje:

- Oficjalne godziny: poniedziałek 18:05-19:05, środa 15:00-16:00 (od bieżącego tygodnia).
- Oprócz tego, można się umawiać (tzn. wcześniej informując) na dowolny inny termin, najchętniej w środę 12:00-15:00 lub po 18:00.
- Konsultacje mają swój zespół (link i hasło na kanale ogólnym).

Wciąż brak oficjalnego stanowiska względem możliwości przejścia na Zooma, ale negocjacje idą w dobrym kierunku :)

Poprzedni wykład:

- Wyrażenia arytmetyczne i ogólne (w tym: kolejność *niektórych* operatorów).
- Przypisania

```
y = 3  
y = 2 * x # 2 * x wyliczone "tu i teraz": y == 6
```

- Obiekty: liczby, napisy, wartości logiczne, ...
- Typy obiektów: int, float, bool, str, ... (pytanie: czemu osobne typy int i float?).
- Podstawy I/O: input() i print().
- Podstawy kontroli sterowania: instrukcja warunkowa if, pętla for (wersja 1).

Drugi rodzaj pętli: while

```
while wyrażenie: # wyrażenie logiczne: True/False  
    ...
```

Kolejne potęgi dwójki < 100 :

```
i = 1  
while i < 100:  
    print(i)  
    i = 2 * i
```

Wczytywanie wartości do skutku:

```
k = 0  
while k <= 0:  
    k = int(input( 'podaj k > 0: ' ))
```

Kontrola sterowania w pętlach

Kontrola nad pętlami:

`break`: przerywa działanie pętli.

`continue`: kończy aktualny obrót pętli.

```
k = 0
while True:
    k = int(input('podaj k > 0: '))
    if k > 0:
        break
```

```
for k in range(0, 100):
    if k % 5 == 1 and k < 30:
        continue
    if k % 5 == 2 and k > 25:
        continue
    print(k)
```

Iteracja: proces powtarzania pewnej operacji (np. pętla w Pythonie).

Potocznie: pojedynczy krok takiego procesu (np. pojedynczy „obrót” pętli).

Obiekt iterowalny (pierwsza przymiarka): obiekt, który reprezentuje ciąg obiektów.

Przykłady: `range` (ciągi liczb), `str` (ciągi pojedynczych znaków).

Obiekty iterowalne

Iterowanie przez obiekty iterowalne.

obj - obiekt iterowalny, np. napis.

```
for x in obj:  
    print(x)
```

Konkretniej:

```
for c in "AbcD":  
    print(c)
```

wynik:

A
b
c
D

Obiekty iterowalne

Wcześniej: `range` traktowaliśmy jako część składni pętli `for`. W rzeczywistości: obiekt iterowalny, podobnie jak napis.

```
my_sequence = range(1, 10)
for element in my_sequence:
    print(element)
```

Inne sposoby konstrukcji `range`: inne (skończone) ciągi arytmetyczne.

```
range(n)           # tak jak range(0, n)
range(-5, 6, 2)    # -5, -3, -1, 1, ..., 5
range(10, 0, -3)   # 10, 7, 4, 1
```

Niektóre inne operacje na (typowych) obiektach iterowalnych:

`len(obj)` - długość ciągu (np. długość napisu: `len('abc') == 3`).

`in`, `not in`: operatory (intuicyjnie: \in, \notin).

```
'x' in 'xyz'           # True  
4 not in range(0, 10, 2) # False
```


Obiekty iterowalne

Typ tuple: *krotka*, skończony ciąg dowolnych obiektów (element Obj^n , gdzie Obj to zbiór wszystkich obiektów).

Krotki długości 2, 3, 4, ...: para, trójka, czwórka, etc.

Konstrukcje krotek (wraz z przypisaniem):

```
three = (1, 'a', 1.0)
test = (0, 0, 0, 0, 0)
```

```
# a nawet prościej:
three = 1, 'a', 1.0
test = 0, 0, 0, 0, 0
```

Krotki są iterowalne:

```
for x in (1, 20, 15, 1):
    print(x)
```

W Pythonie istnieje sposób pisania napisów w ogólnej postaci i podstawiania do nich konkretnych obiektów (tzn. napisów, które je reprezentują).

Operator `%` (w arytmetyce: modulo) służy do formatowania napisów (w „starym” stylu). Ogólna postać:

```
napis % obj
```

gdzie `napis` to napis z pewną ilością „formatek”, a `obj` to krotka obiektów, które mają zostać pod nie podstawione. Przykład:

```
who = 'Ala '  
txt = '%s_ma%s' % (who, 'kota')  # 'Ala ma kota '
```

Więcej: <https://pyformat.info/> „old style”.