Python - wprowadzenie

3.Podstawy python

- **3.1 Pierwsze kroki - rozgrzewka**
- **3.2 Zmienne**
- **3.3 Typy danych**
- **3.4 Intigery i Floaty**
- **3.5 Strings**
- **3.6 Listy**
- **3.7 Tuple**
- **3.8 Sety**
- **3.9 Słowniki**
- **3.10 Porównania i Booleany**
- **3.11 Operatory logiczne**
- **3.12 Instrukcje warunkowe (if, elif,else)**
- **3.13 Pętle (while, for loops)**
- **3.14 Funkcje własne (User defined functions)**
- **3.15 Podstawowa obsługa błędów**

****

3.Podstawy python

****

3.1 Pierwsze kroki - rozgrzewka

W tej części pierwsze interakcje z Pythonem Dla każdego z przykładu dodajemy 2 swoje

In [1]:
```
5 #wyświetlmy numer
```

Out[1]: 5

In [2]:
```
1+1 #dodawanie
```

Out[2]: 2

In [3]:
```
2-1 #odejmowanie
```

Out[3]: 1

In [4]:
```
1*120 #mnożenie
```

Out[4]: 120

In [5]:
```
235/12 #dzielenie
```

Out[5]: 19.583333333333332

In [6]:
```
8**2 #pierwiastek
```

Out[6]: 64

In [7]:
```
type(5) #sprawdzanie typu
```

Out[7]: int

In [8]:
```
type(True) #sprawdzanie typu 1
```

Out[8]: bool

In [9]:
```
type("Dom") #sprawdzanie typu 2
```

Out[9]: str

In [10]:
```
print(1+2+3+5) #funkcja drukująca/ podglądowa
```

```
11
```

In [11]:
```
print("Ala kupiła kota a kot był czarny") #funkcja drukująca/ podglądowa 1
```

```
Ala kupiła kota a kot był czarny
```

In [12]:
```
print("Ala kupiła kota a kot był czarny",1+2+3+5/89) #funkcja drukująca/ podglądowa 2
```

```
Ala kupiła kota a kot był czarny 6.056179775280899
```

In [13]:
```
print("Ala kupiła kota a kot był czarny",1+2+3+5/89,sep='\n') #funkcja drukująca/ podglądowa 3 z użyciem separatora lini
```

```
Ala kupiła kota a kot był czarny
6.056179775280899
```

3.2 Zmienne

W tej części omówienie zmiennych

In [14]:
```
a = 2 #przypisanie intigera 5 do zmiennej a
```

In [15]:
```
print(a) #podglądamy a
```

```
2
```

In [16]:
```
a=66 #zmianiamy/ nadpisujemy zmienną a
```

In [17]:
```
print(a) #podglądamy a
```

```
66
```

In [18]:
```
b = 2356 #przypisanie intigera 2356 do zmiennej b
```

In [19]:
```
print(b) #podglądamy b
```

```
2356
```

In [20]:
```
b=4 #zmianiamy/ nadpisujemy zmienną b
```

In [21]:
```
print(b) #podglądamy b
```

```
4
```

In [22]:
```
print(a,b) #podglądamy a i b
```

```
66 4
```

In [23]:
```
print("Pies miał",a,"łatek i",b,"łapy",sep='\n') #funkcja drukująca/ podglądowa 3 z użyciem separatora lini
```

```
Pies miał
66
łatek i
4
łapy
```

In [24]:
```
print(a+b) #dodawanie zmiennych
```

```
70
```

In [25]:
```
print(a-b) #odejmowanie
```

```
62
```

In [26]:
```
print(a*b) #mnożenie
```

```
264
```

In [27]:
```
print(a/b) #dzielenie
```

```
16.5
```

In [28]:
```
print(a**b) #pierwiastek
```

```
18974736
```

In [29]:
```
c= a+b #przypisywanie zmiennych do innej zmiennej
```

In [30]:
```
print(c) #podgląd zmiennej c
```

```
70
```

In [31]:
```
type(a) #podgląd typu zmiennej a
```

Out[31]:
```
int
```

In [32]:
```
f="Kalkulator"
```

In [33]:
```
print(f) #podgląd zmiennej f
```

```
Kalkulator
```

In [34]:
```
type(f) #podgląd typu zmiennej f
```

Out[34]:
```
str
```

In [35]:
```
a =1 #łańcuch przypisań zmiennych 1
```

In [36]:
```
b=a #łańcuch przypisań zmiennych 2
```

In [37]:
```
c=b #łańcuch przypisań zmiennych 3
```

In [38]:
```
d=c #łańcuch przypisań zmiennych 4
```

In [39]:
```
print(d) #podgląd zmiennej d
```

```
1
```

In [40]:
```
b=8
```

In [41]:
```
print(d) #podgląd zmiennej d
```

```
1
```

In [42]:
```
print(b) #podgląd zmiennej d
```

```
8
```

3.3 Typy danych

W tej części omówienie typów danych

In [43]:
```
print(type("tekst")) #przegląd typów danych
print(type(1))
print(type(1.0))
print(type(True))
print(type("1.0"))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'str'>
```

In [44]:
```
a="tekst" #przypisanie danych do zmiennych
b=1
c=1.0
d=True
e="1.0"
```

In [45]:
```
print(type(a)) #przegląd typów danych po przypisaniu do zmiennych
print(type(b))
```

```
print(type(c))
print(type(d))
print(type(e))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'str'>
```

3.4 Intigers & Floats

W tej części omówienie Intigerów i Floatów

In [46]:
```
5
```

Out[46]:
```
5
```

In [47]:
```
5.0
```

Out[47]:
```
5.0
```

In [48]:
```
25986
```

Out[48]:
```
25986
```

In [49]:
```
a=5
```

In [50]:
```
b=25986
```

In [51]:
```
c=-500000
```

In [52]:
```
print(a,b)
```

```
5 25986
```

In [53]:
```
print(type(a),type(b))
```

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

Dodawanie

In [54]:
```
1+2
```

Out[54]: 3

In [55]:
```
1+258984+a
```

Out[55]: 258990

In [56]:
```
a+a+a+a+a+a+b+c
```

Out[56]: -473984

In [57]:
```
1.0+2.0
```

Out[57]: 3.0

Odejmowanie

In [58]:
```
b-c-a
```

Out[58]: 525981

In [59]:
```
b+c-a
```

Out[59]: -474019

In [60]:
```
2896-698
```

Out[60]: 2198

In [61]:
```
2896.00-698.00
```

Out[61]: 2198.0

Mnożenie

In [62]:
```
a*b
```

Out[62]: 129930

In [63]:
```
b*c
```

Out[63]: -12993000000

In [64]:
```
d=a*b
```

In [65]:
```
d
```

Out[65]:
```
129930
```

Dzielenie

In [66]:
```
b/c
```

Out[66]:
```
-0.051972
```

In [67]:
```
d/a
```

Out[67]:
```
25986.0
```

In [68]:
```
print(type(b/c))
print(type(d/a))
```

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

In [69]:
```
print(round(b/c,2))
print(round(b/c,1))
print(round(b/c,0))
print(round(b/c))
```

```
-0.05
-0.1
-0.0
0
```

In [70]:
```
print(type(round(b/c,2)))
print(type(round(b/c,1)))
print(type(round(b/c,0)))
print(type(round(b/c)))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'int'>
```

In [71]:
```
d/30
```

Out[71]: 4331.0

In [72]:
```
a/5
```

Out[72]: 1.0

In [73]:
```
f=a/b
```

In [74]:
```
f.as_integer_ratio()
```

Out[74]: (7098723956634169, 36893488147419103232)

In [75]:
```
f
```

Out[75]: 0.00019241129839144154

Jak zmieniaja się dane w zależności od operacji i typu danych Intigery i Floaty

In [76]:
```
a=66
b=66.00
c=82.84
d=82
e=a+d
f=b+c
```

In [77]:
```
print(a)
print(type(a))
print(b)
print(type(b))
print(c)
print(type(c))
print(d)
print(type(d))
print(e)
print(type(e))
print(f)
print(type(f))
```

```
66
<class 'int'>
66.0
<class 'float'>
82.84
<class 'float'>
82
<class 'int'>
148
<class 'int'>
148.84
<class 'float'>
```

In [78]:
```
type(a+1)
```

Out[78]:
```
int
```

In [79]:
```
type(a+1.0)
```

Out[79]:
```
float
```

In [80]:
```
type(b)
```

Out[80]:
```
float
```

In [81]:
```
int(b)
```

Out[81]:
```
66
```

In [82]:
```
b=int(b)
```

In [83]:
```
b
```

Out[83]:
```
66
```

In [84]:
```
f
```

Out[84]:
```
148.84
```

In [85]:
```
type(f)
```

Out[85]:
```
float
```

In [86]:
```
f=int(f)
```

In [87]:
```
f
```

Out[87]:
```
148
```

In [88]:
```
type(f)
```

Out[88]:
```
int
```

3.5 Strings

W tej części omówienie Stringów

In [89]:
```
a="test"
a
```

Out[89]:
```
'test'
```

In [90]:
```
type(a)
```

Out[90]:
```
str
```

In [91]:
```
b="Ćwiczenie"
c="Pomidor"
d="Kawa"
```

In [92]:
```
print(d)
print(type(d))
```

```
Kawa
<class 'str'>
```

Łączenie tekstu

In [93]:
```
a+b
```

Out[93]:
```
'testĆwiczenie'
```

In [94]:
```
b+c+a
```

Out[94]: 'ĆwiczeniePomidortest'

In [95]:
```
print(type(a+b))
```

```
<class 'str'>
```

In [96]:
```
print(c+" i "+"zielony ogórek"+" "+"to dobry zestaw do kanapki"+" A do tego do śniadania musi być "+d)
```

```
Pomidor i zielony ogórek to dobry zestaw do kanapki A do tego do śniadania musi być Kawa
```

In [97]:
```
print(c+" i "+"zielony ogórek"+" "+"to dobry zestaw do kanapki"+"\nA do tego do śniadania musi być "+d)
```

```
Pomidor i zielony ogórek to dobry zestaw do kanapki
A do tego do śniadania musi być Kawa
```

Ekstrakcja tekstu (NIE DZIAŁA!!!)

In [98]:
```
b-c
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[98], line 1
----> 1 b-c

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

In [99]:
```
c-d
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[99], line 1
----> 1 c-d

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Replikacja tekstu

In [100…
```
c*b #nie można mnożyć tekstu przez tekst
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[100], line 1
----> 1 c*b #nie można mnożyć tekstu przez tekst

TypeError: can't multiply sequence by non-int of type 'str'
```

In [101…
```
c*5
```

Out[101]:
```
'PomidorPomidorPomidorPomidorPomidor'
```

In [102…
```
d*555
```

Out[102]:
```
'KawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKa
waKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaK
awaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawa
KawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKaw
aKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKa
waKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaK
awaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawa
KawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKaw
aKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKa
waKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaK
awaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawa
KawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKaw
aKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKa
waKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaK
awaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawa
KawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawaKawa'
```

In [103…
```
d*555.0
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[103], line 1
----> 1 d*555.0

TypeError: can't multiply sequence by non-int of type 'float'
```

Jak zmieniaja się dane w zależności od operacji i typu danych Intigery, Floaty i Stringi

In [104…
```
a=8
b="Kawa"
c=8.0
print(a)
print(type(a))
print(b)
print(type(b))
print(c)
print(type(c))
```

```
8
<class 'int'>
Kawa
<class 'str'>
8.0
<class 'float'>
```

In [105…
```
a+b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[105], line 1
----> 1 a+b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [106…
```
c+b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[106], line 1
----> 1 c+b

TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

In [107…
```
str(c)
```

Out[107]:
```
'8.0'
```

In [108…
```
c=str(c)
```

In [109…
```
c+b
```

Out[109]:
```
'8.0Kawa'
```

In [110…
```
a*b
```

Out[110]:
```
'KawaKawaKawaKawaKawaKawaKawaKawa'
```

In [111…
```
c*b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[111], line 1
----> 1 c*b

TypeError: can't multiply sequence by non-int of type 'str'
```

In [112…
```
a+b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[112], line 1
----> 1 a+b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [113…
```
a=str(a)
```

In [114…
```
a+b
```

Out[114]:
```
'8Kawa'
```

In [115…
```
a+c
```

Out[115]:
```
'88.0'
```

3.6 Listy

W tej części omówienie List Listy mogą składać się z:

- floatów
- stringów
- intigerów
- Booleanów

Listy deklarujemy poprzez nawiasy kwadratowe, możemy dokonywać na nich różnych operacji oraz pobierać konkretne elementy z listy.

In [171…
```
lista1=[1,2,3,4]
```

In [172…
```
lista1
```

Out[172]:
```
[1, 2, 3, 4]
```

In [173…
```
lista2=[1,2,True,"string",1.0]
```

In [174…
```
lista2
```

Out[174]:
```
[1, 2, True, 'string', 1.0]
```

In [175…
```
type(lista1)
```

Out[175]:
```
list
```

In [176…
```
len(lista2) #zliczanie elementów listy
```

Out[176]:
```
5
```

Działania na listach

Na listach działaja takie same operacje jak w przypadku stringów, floatów etc ,aczkolwiek występują pewne ograniczenia

- brak możliwości łączenia tekstu
- brak możliwości ekstrakcji

Poza tym listy można:

- łączyć
- sortować
- odwracać
- usuwać konkretne elementy z list

In [177…
```
sum(lista1)
```

Out[177]:
```
10
```

In [178…
```
2*lista1
```

Out[178]:
```
[1, 2, 3, 4, 1, 2, 3, 4]
```

In [179…
```
sum(lista2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[179], line 1
----> 1 sum(lista2)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [180…
```
2*lista1
```

Out[180]:
```
[1, 2, 3, 4, 1, 2, 3, 4]
```

In [181…
```
l1 = lista1
l2 = lista2
l3 = ["a","b","dfg"]
```

In [182…
```
l1+l2 # to jest łączenie list a nie "zwyczajowe sumowanie"
```

Out[182]:
```
[1, 2, 3, 4, 1, 2, True, 'string', 1.0]
```

In [183…
```
l1+l2+l3+["c","d","op"]
```

Out[183]:
```
[1, 2, 3, 4, 1, 2, True, 'string', 1.0, 'a', 'b', 'dfg', 'c', 'd', 'op']
```

In [184…
```
sum(l3)# to jest sumowanie listy na wartościah tekstowych które nie działa
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[184], line 1
----> 1 sum(l3)# to jest sumowanie listy na wartościah tekstowych które nie działa

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Wbudowane metody (działania) na listach

In [185…
```
l99=9999
l1.append(l99) # dodanie elementu do listy
```

In [186…
```
l1
```

Out[186]:
```
[1, 2, 3, 4, 9999]
```

In [187…
```
l999=[101,102,103]
l1.extend(l999) # dodanie sekwencji - zbioru elementow
```

In [188…
```
l1
```

Out[188]:
```
[1, 2, 3, 4, 9999, 101, 102, 103]
```

In [189…
```
l1+2 # bezpośrednie dodanie elementu do listy nie jest możliwe w taki sposób
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[189], line 1
----> 1 l1+2 # bezpośrednie dodanie elementu do listy nie jest możliwe w taki sposób

TypeError: can only concatenate list (not "int") to list
```

In [190…
```
l1.insert(3,5) # dodanie elementu do listy na pozycji indexu (x-wartosc indexu,y-wartość dodawana do listy)
print(l1)
```

```
[1, 2, 3, 5, 4, 9999, 101, 102, 103]
```

In [191…
```
l11=["v","t"]
l1.extend(l11)
print(l1)
```

```
[1, 2, 3, 5, 4, 9999, 101, 102, 103, 'v', 't']
```

In [192…
```
l1.pop(2) #usuwanie elementu z listy wg pozycji indexu
l1
```

Out[192]:
```
[1, 2, 5, 4, 9999, 101, 102, 103, 'v', 't']
```

In [193…
```
l1.remove("v") #usuwanie elementu z listy wg wartości
l1
```

Out[193]:
```
[1, 2, 5, 4, 9999, 101, 102, 103, 't']
```

In [194…
```
l1.remove("t") #usuwanie elementu z listy wg wartości
l1
```

Out[194]:
```
[1, 2, 5, 4, 9999, 101, 102, 103]
```

In [195…
```
l1.reverse() #odwracanie listy
l1
```

Out[195]:
```
[103, 102, 101, 9999, 4, 5, 2, 1]
```

In [196…
```
l1.sort() #sortowanie list
l1
```

Out[196]:
```
[1, 2, 4, 5, 101, 102, 103, 9999]
```

In [197…
```
l1.sort(reverse=True)# sortowanie od największej do najmniejszej
l1
```

Out[197]:
```
[9999, 103, 102, 101, 5, 4, 2, 1]
```

In [198…
```
l1.sort(reverse=False)
l1
```

Out[198]:
```
[1, 2, 4, 5, 101, 102, 103, 9999]
```

Wybieranie elementów z list

W pythonie początek listy a także czegokolwiek co może być policzalne wg domyślnej numeracji równy jest 0 a nie 1

Można także wybierać elementy z listy lub czegokolwiek wg domyślnej numeracji od końca np. -1 będzie oznaczać ostatni element z listy

Można także wybierać elementy całymi grupami lub w interwałach za to odpowiada zapis lista[x1,x2,x3]

- x1 - wartość początowa
- x2 - wartość końcowa (UWAGA wartość końcowa nie jest uwzględniana w przedziale)
- x3 - interwał

In [199…
```
l5 = [1,2,3,4,1.89]
```

In [200…
```
l5[1] #czy to jest pierwszy element z listy?
```

Out[200]:
```
2
```

In [201…
```
l5[0] #czy to jest pierwszy element z listy?
```

Out[201]:
```
1
```

In [202…
```
print(l5[0])
print(l5[1])
print(l5[2])
```

```
print(l5[3])
print(l5[4])
```

```
1
2
3
4
1.89
```

In [203…
```
l5[-1]
```

Out[203]:
```
1.89
```

In [204…
```
l5[4]
```

Out[204]:
```
1.89
```

In [205…
```
print(l5[-2])
print(l5[3])
```

```
4
4
```

In [206…
```
l6=[1,2,6,99,88.00,66.89,"True","False",True,False]
```

In [207…
```
l6
```

Out[207]:
```
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
```

In [208…
```
l6[:] #wszystkie elementy z listy
```

Out[208]:
```
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
```

In [209…
```
l6[0:99]#wszystkie elementy z listy
```

Out[209]:
```
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
```

In [210…
```
l6[0:2] #wybieramy zakres od pierwszego do 3 elementu (zakres końcowy nie będzie uwzględniony), zobaczymy tylko 2 elementy
```

Out[210]:
```
[1, 2]
```

In [211…
```
l6[2:0] #błędnie logiczny zapis
```

Out[211]:
```
[]
```

In [212…
```
l6[-5:-1]
```

Out[212]: [66.89, 'True', 'False', True]

In [213…
```
l6[-1:-5] #błędnie logiczny zapis
```

Out[213]: []

In [214…
```
l6[0:99]#wszystkie elementy z listy
```

Out[214]: [1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]

In [215…
```
l6[0:6:2] #wybieramy zakres od pierwszego do 7 elementu (zakres końcowy nie będzie uwzględniony)
          # w interwale co 2, zobaczymy tylko 3 elementy
```

Out[215]: [1, 6, 88.0]

In [216…
```
l6[0:9:1]
```

Out[216]: [1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True]

In [217…
```
l6[0:9:3]
```

Out[217]: [1, 99, 'True']

In [218…
```
l6[0::]
```

Out[218]: [1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]

In [219…
```
l6[::]
```

Out[219]: [1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]

In [220…
```
l6[::3]
```

Out[220]: [1, 99, 'True', False]

Podmiana elementów w liście

In [221…
```
l6=[1,2,6,99,88.00,66.89,"True","False",True,False]
l7=[1,2,6,99,88.00,66.89,"True","False",True,False]
```

```
print(l6)
print(l7)
```

```
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
```

In [222…]
```
l7[0]="Ala" #podmiana element po elemencie przez przypisanie
l7[1]="ma"
l7[2]="kota"
l7[3]="od"
l7[4]=6
l7[5]="lat"
l7[6]="i"
l7[7]=2.0
l7[8]="miesiecy"
l7[9]="."
```

In [223…]
```
print(l6)
print(l7)
```

```
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
['Ala', 'ma', 'kota', 'od', 6, 'lat', 'i', 2.0, 'miesiecy', '.']
```

In [224…]
```
l7[0:10]=[0,1,2,3,4,5,6,7,8,9] #podmiana seriami 1
print(l6)
print(l7)
```

```
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [225…]
```
l7[4:10]=[99,99,99,99,99,99] #podmiana seriami 2
l7[0:3]=[99,99,99]
print(l7)
```

```
[99, 99, 99, 3, 99, 99, 99, 99, 99, 99]
```

Dowawanie elementów do listy przez przypisanie zakresami (Bardzo ryzykowne - raczej unikać)

In [226…]
```
len(l7) # sprawdzamy długość listy
```

Out[226]:
```
10
```

In [227…]
```
l7[0:10]=[0,1,2,3,4,5,6,7,8,9,"Test"]
print(l7)
print(len(l7))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'Test']
11
```

In [228…
```
tuple(l7)
```

Out[228]:
```
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'Test')
```

In [229…
```
list(tuple(l7))
```

Out[229]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'Test']
```

3.7 Tuple

Praca z tuplami jest bardzo ograniczona mimo iż wyglądają podobnie i różnią się nawiasami względem list.

Jakie są różnice między listami a tuplami:

- tuple są niezmienialne
- tupli nie można sortować
- tupli nie można odwracać
- iteracja po tuplach jest szybsza
- tuple zajmują mniej pamięci niż listy

In [230…
```
t1=(1,2,6,99,88.00,66.89,"True","False",True,False)
```

In [231…
```
print(t1)
print(type(t1))
print(l6)
print(type(l6))
```

```
(1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False)
<class 'tuple'>
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
<class 'list'>
```

In [232…
```
t1[0:10]=(0,1,2,3,4,5,6,7,8,9)
print(t1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[232], line 1
----> 1 t1[0:10]=(0,1,2,3,4,5,6,7,8,9)
      2 print(t1)

TypeError: 'tuple' object does not support item assignment
```

In [233…
```
t1[1]=2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[233], line 1
----> 1 t1[1]=2

TypeError: 'tuple' object does not support item assignment
```

Zamiana tupli na listy i list na tuple

In [234…
```
type(t1)
```

Out[234]:
```
tuple
```

In [235…
```
l1=list(t1)
print(type(l1))
print(l1)
```

```
<class 'list'>
[1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False]
```

In [236…
```
l1=tuple(t1)
print(type(l1))
print(l1)
```

```
<class 'tuple'>
(1, 2, 6, 99, 88.0, 66.89, 'True', 'False', True, False)
```

3.8 Sety

Sety stanowią listę "UNIKALNYCH" elementów. Podmiana elementów w secie nie jest możliwa można tylko dodawać elementy do setu

In [237…
```
s1={1,1,2,22,2,2,3,3,44,4,4,5}
print(s1)
print(type(s1))
```

```
{1, 2, 3, 4, 5, 22, 44}
<class 'set'>
```

In [238…
```
l1=[1,1,2,22,2,2,3,3,44,4,4,5]
print(l1)
print(type(l1))
```

```
[1, 1, 2, 22, 2, 2, 3, 3, 44, 4, 4, 5]
<class 'list'>
```

Wbudowane metody (działania) na setach

In [239…
```
s1.add(99) # dodanie elementu do setu
print(s1)
```

```
{1, 2, 3, 4, 5, 22, 99, 44}
```

In [240…
```
s1.remove(99) # usunięcie elementu z setu
print(s1)
```

```
{1, 2, 3, 4, 5, 22, 44}
```

In [241…
```
s2={"a","b","a"}
```

In [242…
```
print(s1)
print(s2)
```

```
{1, 2, 3, 4, 5, 22, 44}
{'b', 'a'}
```

In [243…
```
s1.update(s2) # dodanie setów do siebie
print(s1)
```

```
{1, 2, 3, 4, 5, 'b', 'a', 44, 22}
```

In [244…
```
s1.pop() # usunięcie pierwszego elementu z setu
print(s1)
```

```
{2, 3, 4, 5, 'b', 'a', 44, 22}
```

In [245…
```
s1.remove(5) # usunięcie konkretnego elementu z setu
print(s1)
```

```
{2, 3, 4, 'b', 'a', 44, 22}
```

In [246…
```
s1
```

Out[246]: {2, 22, 3, 4, 44, 'a', 'b'}

In [247…
```
s1|s2 # dodanie (złączenie) setów do siebie
```

Out[247]: {2, 22, 3, 4, 44, 'a', 'b'}

In [248…
```
s1&s2 # punkt wspólny setów
```

Out[248]: {'a', 'b'}

In [249…
```
s1-s2 # różnicowanie setów
```

Out[249]: {2, 3, 4, 22, 44}

3.9 Słowniki

Słowniki podobnie jak sety zamknięte są w klamrach, aczkolwiek różnicuje je to że są zapisywane chronologicznie wraz z zapisem pary którym jest klucz oraz wartość.

In [250…
```
sl1={1:"a",2:"b",3:"c",4:"d"}
print(sl1)
print(type(sl1))
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}
<class 'dict'>
```

In [251…
```
s1={"a","b","c","d"}
print(s1)
print(type(s1))
```

```
{'b', 'd', 'c', 'a'}
<class 'set'>
```

Wbudowane metody (działania) na słownikach

In [252…
```
print(sl1[1]) # wybieranie elementów
print(sl1[3])
```

```
a
c
```

In [253…
```
sl2={5:"a",6:8}
```

In [254…
```
sl1.update(sl2) # dodawanie elementów do słownika
print(sl1)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'a', 6: 8}
```

In [255…
```
sl1.update([(11,"x"),(12,"y")]) # dodawanie elementów do słownika przez tuple
print(sl1)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'a', 6: 8, 11: 'x', 12: 'y'}
```

In [256…
```
sl3={1:"a",2:8}
```

In [257…
```
sl1.update(sl3) # nadpisanie elementów słownika
print(sl1)
```

```
{1: 'a', 2: 8, 3: 'c', 4: 'd', 5: 'a', 6: 8, 11: 'x', 12: 'y'}
```

In [258…
```
sl1.keys() # listowanie kluczy
```

Out[258]:
```
dict_keys([1, 2, 3, 4, 5, 6, 11, 12])
```

In [259…
```
sl1.values()
```

Out[259]:
```
dict_values(['a', 8, 'c', 'd', 'a', 8, 'x', 'y'])
```

In [260…
```
sl1.items()
```

Out[260]:
```
dict_items([(1, 'a'), (2, 8), (3, 'c'), (4, 'd'), (5, 'a'), (6, 8), (11, 'x'), (12, 'y')])
```

In [261…
```
sl1.get(4) # pobieranie elementu ze słownika wg klucza
```

Out[261]:
```
'd'
```

In [262…
```
sl1.pop(1) # usuwanie elementu wg klucza
print(sl1)
```

```
{2: 8, 3: 'c', 4: 'd', 5: 'a', 6: 8, 11: 'x', 12: 'y'}
```

3.10 Porównania i Booleany

Ekwiwalentem True jest 1 natomiast False jest 0

In [263…
```
a=6
b=7
```

```
c=11
d=True
f=False
```

In [264…
```
a == b # równe
```

Out[264]: False

In [265…
```
a == a
```

Out[265]: True

In [266…
```
a != b # różne
```

Out[266]: True

In [267…
```
a > b
```

Out[267]: False

In [268…
```
a < b
```

Out[268]: True

In [269…
```
a*b < c+c+c*2
```

Out[269]: True

3.11 Operatory logiczne

In [270…
```
1 and 1 # operator "i"
```

Out[270]: 1

In [271…
```
True & True # operator "i"
```

Out[271]: True

In [272…
```
(1<2) & (1>3)
```

Out[272]: False

In [273…
```
False or False # operator "lub"
```

Out[273]: False

In [274…
```
True or False
```

Out[274]: True

In [275…
```
True | False # operator "lub"
```

Out[275]: True

In [276…
```
not 1 # operator "nie/inny"
```

Out[276]: False

In [277…
```
not False
```

Out[277]: True

In [278…
```
not((not(9 < 22)) | (37 > 200))
```

Out[278]: True

3.12 Instrukcje warunkowe (if, elif,else)

In [279…
```
a=2
b=3
c=5
```

In [280…
```
if a < b:                    # jeżeli
    print("Tak to prawda")
else:                        # w przeciwnym razie
    print("Nie to nieprawda")
```

Tak to prawda

In [281…
```
if a > b:                    # jeżeli
    print("Tak to prawda")
elif a+b==c:                 # jeżeli od drugiej i kolejnej iteracji
    print("Tak to prawda")
```

```
else:                          # w przeciwnym razie
    print("Nie to nieprawda")
```

```
Tak to prawda
```

In [282…
```
ilość = 140
zniżka = 0
cena = 50
```

In [283…
```
if ilość>100 and ilość < 110:
    zniżka = 0.1
    cena = cena -(cena * zniżka)
    print("Nowa cena to {}".format(cena))
elif ilość>=110 and ilość<120:
    zniżka = 0.2
    cena = cena -(cena * zniżka)
    print("Nowa cena to {}".format(cena))
elif ilość>=120:
    zniżka = 0.3
    cena = cena -(cena * zniżka)
    print("Nowa cena to {}".format(cena))
else:
    print("Brak zniżki")
```

```
Nowa cena to 35.0
```

3.13 Pętle (while, for loops)

while - wykonuje się do zadeklarowanej ilości wystąpień

In [284…
```
a=0
while a <100:
    a = a+1
    print(a)
```

```
93
94
95
96
97
98
99
100
```

In [285…

```python
a=10
while a <30:
    a = a+1
    print(a)
```

```
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

In [286…

```python
a=89
while a <100:
    a = a + 1
    if a < 5:
        print("Mniejsze od 5")
    elif a >= 5 and a < 50:
        print("Przedział 5-50")
    else:
        print("Wieksze od 50")
```

```
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
Wieksze od 50
```

for - wykonuje się dla zadeklarowanej ilości wystąpień, przeważnie w jakimś zestawie danych

In [287…
```
a = [5,2,3,4,5,6]
a=a*1
print(a)
```

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

In [288…
```
for i in a:
    print(a)
```

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

In [289…
```
for i in a:
    print(i)
```

```
5
2
3
4
5
6
```

In [290…
```
b=0
for i in a:
    b += i
    print(b)
```

```
5
7
10
14
19
25
```

In [291…
```
6 // 2
```

Out[291]:
```
3
```

In [292…
```
for i in a:
    if i/2 == round(i/2):
        print(i)
```

```
2
4
6
```

In [293…
```
lista1=[]
lista2=[]
lista3=[1,2,3,4,5,6]

for i in lista3:
    if i/2 == round(i/2):
        lista1.append(i)
    else:
        lista2.append(i)
```

In [294…
```
print(lista1)
print(lista2)
print(lista3)
```

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

3.14 Funkcje własne (User defined functions)

Funkcje tworzymy przez element definiujący "def" i element który będzie zwracać wynik "return"

In [295…
```
def mnozenie1(x,z): # funkcja mnożąca 2 argumenty
    wynik=x*z
    return wynik
```

In [298…
```
mnozenie1(1,2)
```

Out[298]: 2

In [299…
```
def mnozenie2(x,z): # funkcja mnożąca 2 argumenty
    return x*z
```

In [300…
```
mnozenie2(2,6)
```

Out[300]: 12

In [301…
```
lista1=[1,2,3,4,5,6,7,8,9,10]
```

In [302…
```
def srednia(x):
    a = sum(x)
    b = len(x)
    sred = a/b
    return sred
```

In [303…
```
type(srednia(lista1))
```

Out[303]: float

In [304…
```
def konwerterFloatowIntigerowNaStringi(x):
    if type(x)==float:
        return str(x)
    elif type(x)==int:
        return str(x)
    elif type(x)==str:
        return print("Nie ma potrzeby konwersji, pozostaje {}".format(x))
```

In [305…
```
konwerterFloatowIntigerowNaStringi(64)
```

Out[305]: '64'

In [306…
```
def podstawowefunkcjematematyczne(x,y):
    dodawanie = x+y
    odejmowanie = x-y
    mnozenie = x*y
    dzielenie = x/y
    return dodawanie,odejmowanie,mnozenie,dzielenie
```

In [307…
```
podstawowefunkcjematematyczne(64,8)
```

Out[307]:
```
(72, 56, 512, 8.0)
```

In [308…
```
a = konwerterFloatowIntigerowNaStringi(65)
```

In [309…
```
print(a)
print(type(a))
```

```
65
<class 'str'>
```

In [310…
```
l = [1,2,3,"bak",4,"płot"]
```

In [311…
```
def expadnerlist(lista,dodatkowyelement):
    return lista.append(dodatkowyelement)
```

In [312…
```
expadnerlist(l,6)
print(l)
```

```
[1, 2, 3, 'bak', 4, 'płot', 6]
```

3.15 Podstawowa obsługa błędów

Podstawową obsługę błędów wykonujemy używają "try" i "except"

In [313…
```
def podstawowefunkcjematematyczne(x,y):
    try:
        dodawanie = x+y
        odejmowanie = x-y
        mnozenie = x*y
        dzielenie = x/y
        return dodawanie,odejmowanie,mnozenie,dzielenie
    except:
        print("W funkcji musiał pojawić się parametr niezgodny z obsługą funkcji matematycznych")
```

In [314…
```
podstawowefunkcjematematyczne("1",2)
```

```
W funkcji musiał pojawić się parametr niezgodny z obsługą funkcji matematycznych
```