In [1]:
```
import sys
sys.version_info
```

Out[1]:
```
sys.version_info(major=3, minor=5, micro=2, releaselevel='final', serial=0)
```

In [7]:
```
23 ** 4
```

Out[7]:
```
279841
```

In [19]:
```
def hovadina():
    """nic"""
    pass

help(hovadina)
```

```
Help on function hovadina in module __main__:

hovadina()
    nic
```

In [47]:
```
pom = [0,4,1]
pom.append(4)
pom
```

Out[47]:
```
[0, 4, 1, 4]
```

In [49]:
```
pom = {'1':'a', '2':2}
pom['1']
```

Out[49]:
```
'a'
```

In [56]:
```
pom = []
dir(pom)
```

Out[56]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

In [57]:
```
help(dir)
```

```
Help on built-in function dir in module builtins:

dir(...)
    dir([object]) -> list of strings

    If called without an argument, return the names in the current scope.
    Else, return an alphabetized list of names comprising (some of) the attribu
tes
    of the given object, and of attributes reachable from it.
    If the object supplies a method named __dir__, it will be used; otherwise
    the default dir() logic is used and returns:
      for a module object: the module's attributes.
      for a class object:  its attributes, and recursively the attributes
        of its bases.
      for any other object: its attributes, its class's attributes, and
        recursively the attributes of its class's base classes.
```

Python sa da pouzit ako kalkulacka. Njcastejsie operatory, ktore sa daju pouzit na pracu s cislami su + - * / // %

- Kalkulacka

```
 + - * / ** // %
< > == and or
_
```

- Reťazce

```
" "" ''' """ \n
```

```
 + * [] [-1] [:] [len(_)+vela] [:len(_)+vela]
```

- konverzia: `int, str`
- Zoznam (list)

```
prakticky vsetky operacie ako s retazcom
```

```
 list(), range, spojenie
append
in
```

- Slovník (dict)

```
{}, dict()
print, input, len, dir, help, type (__class__)
while, if, for
```

Logicke operatory sa zapisuju slovne and or not a nie znakmi & alebo |

In [58]:
```
1 and True
```

Out[58]:
```
True
```

In [61]:
```
4 & 8
```

Out[61]:
```
0
```

Na zapis retazcov je niekolko roznych sposobov. Okrem inych aj viacriadkove

In [ ]:
```
'dssds'
```

In [ ]:
```
"sdsd"
```

In [ ]:
```
'''asasa\tsas
asasasas
asasa'''
```

In [ ]:
```
"""sdsd\tsdsd
sdsds"""
```

In [65]:
```
a = "aaaa"
a
```

Out[65]:
```
'aaaa'
```

In [66]:
```
b = 'bbbbb'
b
```

Out[66]:
```
'bbbbb'
```

k znakom retazcu a podretazcom sa pristupuje ako k prvkom pola

In [62]:
```
pom = "hatlanina"
pom[:len(pom)+5]
```

Out[62]:
```
'hatlanina'
```

retazce su immutable

In [67]:
```
id(a)
```

Out[67]:
```
139696185203488
```

In [68]:
```
"aaas " + 'asasas'
```

Out[68]:
```
'aaas asasas'
```

In [69]:
```
id(a + 'asasas')
```

Out[69]:
```
139696185239728
```

In [70]:
```
a[3] = 'f'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-70-c36d518ac4e2> in <module>()
----> 1 a[3] = 'f'

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

zoznamy su mutable

In [71]:
```
a = [1,2,3]
a
```

Out[71]:
```
[1, 2, 3]
```

In [72]:
```
id(a)
```

Out[72]:
```
139696185224008
```

In [73]:
```
a[0] = 'pes'
a
```

Out[73]:
```
['pes', 2, 3]
```

In [74]:
```
id(a)
```

Out[74]:
```
139696185224008
```

tuple je vlastne immutable list

In [75]:
```
a = (1,2,3)
print(a, id(a))
```

```
(1, 2, 3) 139696294668832
```

In [76]:
```
a[0] = 'pes'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-76-f9d9b81d922b> in <module>()
----> 1 a[0] = 'pes'

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

In [77]:
```
 b = (2,3,4)
a + b
```

Out[77]:
```
(1, 2, 3, 2, 3, 4)
```

In [79]:
```
a = list()
a.append(76)
a
```

Out[79]:
```
[76]
```

In [80]:
```
76 in a
```

Out[80]:
```
True
```

In [ ]:
```
list(range(3,8,2)) # list sa da pouzit na konverziu iteratora na zoznam
```

In [ ]:
```
a = {}
a['ff'] = 5
a
```

In [ ]:
```
len(a)
```

In [ ]:
```
dir(a) # dir vrati zoznam nazvov funckii objektu
```

In [ ]:
```
a.__class__
```

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

In [ ]:
```
help(len)
```

In [ ]:
```
# Fibonacciho postupnost
a, b = 0, 1
while b < 10:
    print(b)
    a, b = b, a+b

print(9) # kod bez odsadenia je uz mimo bloku
    print(8) # bezdovodne odsadenie sposoby chybu
```

In [ ]:
```
words = ['cat', 'window', 'defenestrate'] # for cyklus sa pouziva na iteraciu ce
for w in words:
    print(w, len(w))
```

In [ ]:
```
words = ['cat', 'window', 'defenestrate']
for i in range(3): # to iste ako range(0, len(words))
    print(i, words[i])
```

In [ ]:
```
a,b = (1,2) # pomocou ciarky sa da priradit hodnota viacerym premennym naraz
print(a,b)
```

Funkcia enumerate vytvori zo zoznamu iterator, kde kazdy prvok je dvojica (index, prvok zoznamu). Kontruktor list z iteratoru spravi zoznam

In [ ]:
```
list(enumerate(words))
```

In [ ]:
```
words = ['cat', 'window', 'defenestrate']
for i, w in enumerate(words): # pri iteracii sa daju priradit viacere premenne n
    print(i, w)
```

In [ ]:
```
x = int(input("Please enter an integer: ")) # if, elif, else niesu nicim prekvap
if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```