STRING – metódy a funkcie

retazec = 'postupnostZnakov#alebo Cisel<?@'

retazec[i] ...prístup k i-temu znaku reťazca (**začína sa od 0**)

retazec[od:po] ...podreťazec reťazca, začína „od“ a končí tesne pred „po“

retazec[:] kópia celého reťazca = retazec[0:len(retazec)]

len(retazec) ...vráti dĺžku reťazca

retazec.strip() ...vráti reťazec **bez medzier** a ' \n' na začiatku a na konci reťazca

retazec.lower() ...vráti reťazec, v ktorom prevedie všetky písmená **na malé**

retazec.upper() ...vráti reťazec, v ktorom prevedie všetky písmená **na veľké**

podretazec **in** retazec ...vráti **True**, ak reťazec **obsahuje** podreťazec

retazec.replace(co, cim) ...vráti reťazec, v ktorom **nahradí** všetky 'co' iným reťazcom

retazec.count(podretazec) ...zistí **počet výskytov** podreťazca v reťazci

retazec.find(podretazec) ...zistí **index prvého výskytu** podreťazca v reťazci (**-1** ak nenájde)

retazec.format(hodnoty) ... vráti reťazec, v ktorom nahradí '{ }' zadanými hodnotami

```
>>> 'Moje meno je {} a mám {} rokov!'.format('Adam', 24)
'Moje meno je Adam a mám 24 rokov!'
```

retazec.split('oddelovac') ...vráti list stringov

```
>>> osoba='ja,ty,on,ona,ono'
>>> osoba.split(',')
['ja', 'ty', 'on', 'ona', 'ono']
```

```
>>> meno, priezvisko = input('zadaj meno a priezvisko: ').split()
```

LIST – metódy a funkcie

zoznam = [3, 2.5, 'slovo', True, 'prvok']

zoznam[i] ...prístup k i-temu prvku zoznamu (**začína sa číslovať od 0**)

zoznam[od:po] ...časť zoznamu = tzv. **rez**, začína „od“ a končí tesne pred „po“

zoznam[:] kópia celého zoznamu

zoznam[i]=nova_hodnota ...**nahradenie** i-teho prvku zoznamu novou hodnotou

len(zoznam) ...vráti počet prvkov v zozname

sum(zoznam) ...vráti súčet prvkov v zozname čísel

max(zoznam), min(zoznam) ...vráti maximálny, minimálny prvok zo zoznamu

zoznam.append(hodnota) ...pridá do zoznamu ďalšiu hodnotu (na koniec zoznamu)

zoznam.sort() ...usporiada prvky zoznamu od najmenšieho po najväčší

zoznam.pop(i) ...**odoberie** (a vráti) i-ty prvok zoznamu (ak bez i, odoberie posledný)

zoznam.remove(hodnota) ...**odoberie** prvý výskyt hodnoty v zozname (skráti dĺžku o 1)

zoznam.insert(i, hodnota) ...**vsunie** na i-tu pozíciu v zozname prvok hodnotu

zoznam.count(hodnota) ...vráti **počet výskytov** hodnoty v zozname

zoznam.index(hodnota) ...vráti **index prvého výskytu** hodnoty v zozname (spadne ak ju v zozname nenájde)

štandardné funkcie

print(premenna, end='\n'**)** ...vypíše do shellu obsah premennej

input('otázka, výzva'**)** ...načíta od užívateľa z klávesnice vstup,
hodnotu zvyčajne uložíme do nejakej premennej
napr. meno = **input(**'Ako sa voláš?'**)**

type(premenna**)** **...vráti typ premennej (int, float, str, bool, list...)**

int(hodnota**)** ...z danej hodnoty vyrobí celé číslo (ak sa dá)

float(hodnota**)** ...z danej hodnoty vyrobí desatinné číslo (ak sa dá)

str(hodnota**)** ...z danej hodnoty vyrobí reťazec (string)

list(hodnota**)** ...z danej iterovateľnej hodnoty vyrobí zoznam

```
>>> list('ahoj')
['a', 'h', 'o', 'j']
```

abs(cislo) ...vráti absolútnu hodnotu čísla

round(cislo) ...zaokrúhli číslo (na celé, výsledkom je integer)

round(cislo,pocet_miest) ...zaokrúhli číslo na daný počet desatinných miest

range(n) ...vygeneruje postupnosť čísel 0, 1, ... n-1

užívateľom definované funkcie

```
def nazov1():
    prikaz
    prikaz
    ...
```

```
def nazov2(parametre):
    prikaz
    prikaz
    ...
```

```
def nazov3(parametre):
    prikaz
    prikaz
    return navratova_hodnota
```

najprv si musíme definovať svoju funkciu:

- vhodne ju pomenujeme,
- môže mať aj nejaké vstupné parametre (oddelené čiarkou),
- funkcia môže iba niečo vykonať (tzv. procedúra),
- alebo môže aj vrátiť nejakú hodnotu/-ty (príkazom **return**)

potom ju môžeme zavolať (aj v inej funkcii, aj viackrát, s konkrétnymi vstupmi)

nazov1() **nazov2(hodnoty)** **vysledok = nazov3(hodnoty)**

```
def pozdravMa(koho):
    print('Ahoj '+koho)

pozdravMa('Adam')
```

```
def pocitaj(a,b):
    return a+b

sucet=pocitaj(2,3)
print(sucet)
```

DICTIONARY – metódy a funkcie

slovnik = { kluc1: hodnota1, kluc2: hodnota2, kluc3: hodnota3 **}**

...nemá žiadne dva rovnaké kľúče, poradie prvkov tu neexistuje!

slovnik[kluc**]** ...prístup k hodnote, ktorá prislúcha kľúču

slovnik[kluc**]**=nova_hodnota ...**nahradenie** hodnoty pre zvolený kľúč

kluc **in** slovnik ...vráti **True**, ak v slovníku existuje kľúč

```
for kluc in slovnik:
    print(kluc)
```

```
for kluc in slovnik:
    print(kluc, slovnik[kluc])
```

len(slovnik**)** ...vráti počet prvkov v slovníku

slovnik.keys() ...postupnosť kľúčov
slovnik.values() ...postupnosť hodnôt
slovnik.items() ...postupnosť dvojíc kľúč a hodnota

```
for kluc, hodnota in slovnik.items():
        print(kluc, hodnota)
```

SÚBOR – metódy a funkcie

textový súbor = postupnosť riadkov (reťazcov ukončených znakom '\n')

subor = open(*'nazov_suboru.txt'***, 'r',** encoding='utf_8**)**

...otvorenie súboru na čítanie (**r = read**), zápis (**w = write**), pridávanie (**a = append**)

subor.readline() ...**prečíta jeden riadok** súboru (musí byť „otvorený“)
subor.read() ...prečíta obsah **celého súboru**

subor.write(hodnota**)** ...do otvoreného súboru **zapíše hodnotu** (vždy string)
print(co, file=subor, end=';'**)** ...iný spôsob **zápisu** do určeného súboru

subor.close() ...po ukončení práce je nutné súbor **zavrieť**

```
f=open('moj.txt', 'r')
for riadok in f:
    print(riadok)
f.close()
```

```
from random import *
subor=open('nahodne_cisla.txt', 'w')
for i in range(10):
    print(randint(10, 99), file=subor)
subor.close()
```

podmienený príkaz IF

```
if podmienka:
    prikazYES
    prikazYES
dalsie prikazy programu
```

```
if podmienka:
    prikazYES
    prikazYES
else:
    prikazNO
    prikazNO
dalsie prikazy programu
```

najprv sa zistí, **či platí** *podmienka*:

- **ak áno** (*podmienka=True*), tak sa pokračuje príkazmi v prvej časti (prikazYES), po ich vykonaní sa pokračuje *ďalšími príkazmi programu*,
- **ak nie** (podmienka=False), tak sa pokračuje príkazmi v **else** vetve (príkazmiNO), ak nemá else vetvu ihneď pokračuje ďalšími príkazmi programu

a **<** b ... menšie	a **>** b ... väčšie
a **<=** b ... menšie alebo rovné	a **>=** b ... väčšie alebo rovné
a **==** b ... rovnaké	a **!=** b ... rôzne

nieco **in** premenna, nieco **not in** premenna, a **<** b **<** c, (a **or** b) **and** c

vetvenie ELIF

```
if podmienka1:
    prikazy_ak_je_splnena_podmienka1
elif podmienka2:
    prikazy_ak_nie_je_splnena_podmienka1_a_zaroven_plati_podmienka2
elif podmienka3:
    prikazy_ak_neplatili_predchadzajuce_a_zaroven_plati_podmienka3
else:
    prikazy_ak_neplatila_ziadna_z_podmienok
```

```
if vyska<100:
    print('Trpaslík')
elif vyska<150:
    print('Nižší vzrast')
elif vyska<180:
    print('Normálna výška')
elif vyska<200:
    print('Vyšší vzrast')
else:
    print('Dlháň')
```

pre výšku do 100 vypíše 'Trpaslík'

pre výšku <100, 150) 'Nižší vzrast'

pre výšku <150, 180) 'Normálna výška'

pre výšku <180, 200) 'Vyšší vzrast'

pre výšku >=200 vypíše 'Dlháň'

cyklus FOR

riadiaca premenná cyklu, postupne nadobúda hodnoty z postupnosti

je buď číselná, generovaná, vymenovaná prvkami, reťazec, viac reťazcov, zoznam, n-tica, súbor...

for *pocitadlo* **in** *postupnost*:
    prikaz
    prikaz
dalsie prikazy programu

opakuje sa toľkokrát, koľko je členov postupnosti

range(po**)**
range(od, po**)**
range(od, po, krok**)**

```
for i in range(5):
    print(i)
for i in 0,1,2,3,4:
    print(i)
for i in '01234':
    print(i)
for i in [0,1,2,3,4]:
    print(i)
```

pass ...prázdny príkaz (neurob nič)
break ...vyskočenie z cyklu (najčastejšie po splnení nejakej podmienky)
continue ...na tomto mieste sa ukončí vykonávanie danej iterácie cyklu a pokračuje ďalšou hodnotou riadiacej premennej

cyklus WHILE

kým je podmienka splnená, vykonávajú sa príkazy v tele cyklu, opakovane

while *podmienka*:
    prikaz
    prikaz
dalsie prikazy programu

opakuje sa

ak dôjde k zmene a **podmienka prestane platiť**, po jej overovaní už do cyklu nevojde a program pokračuje ďalšími príkazmi
(keď **podmienka neplatí už od začiatku** – ani raz sa nevojde do cyklu a príkazy v ňom sa vôbec nevykonajú, ihneď sa pokračuje ďalšími príkazmi)

...pozor na nekonečný cyklus

```
while 1==1:
    print('pisem')
print('toto sa nikdy nedostane na rad')
```

pass ...prázdny príkaz (neurob nič)
break ...vyskočenie z cyklu (najčastejšie po splnení nejakej podmienky)
continue ...na tomto mieste sa ukončí vykonávanie danej iterácie cyklu a pokračuje ďalšou hodnotou riadiacej premennej