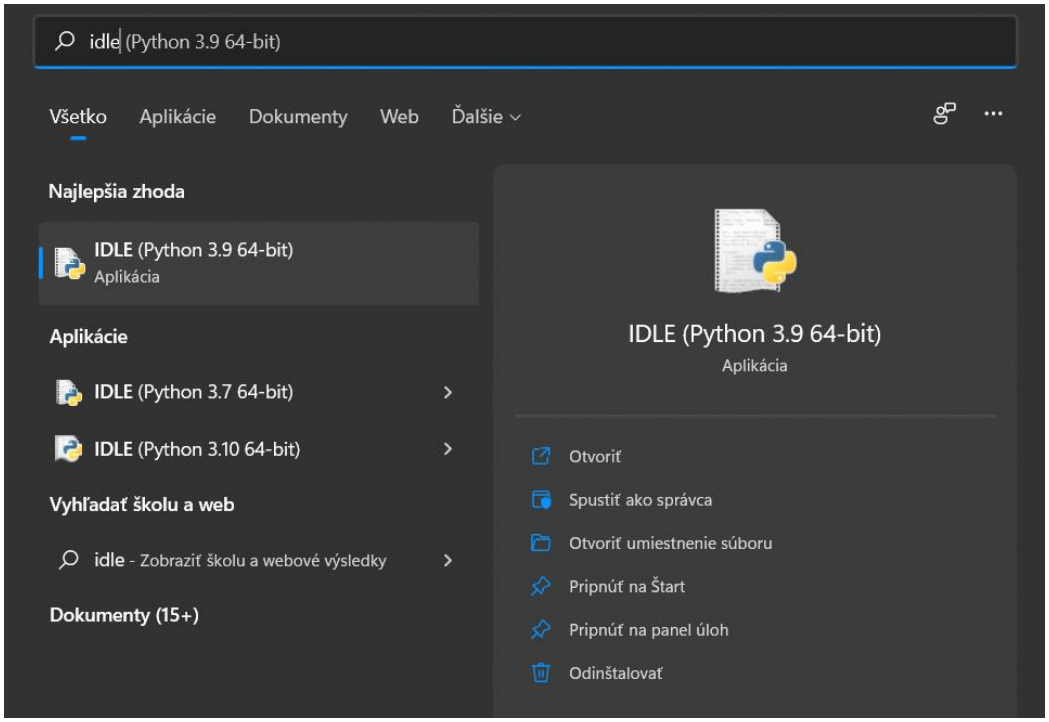
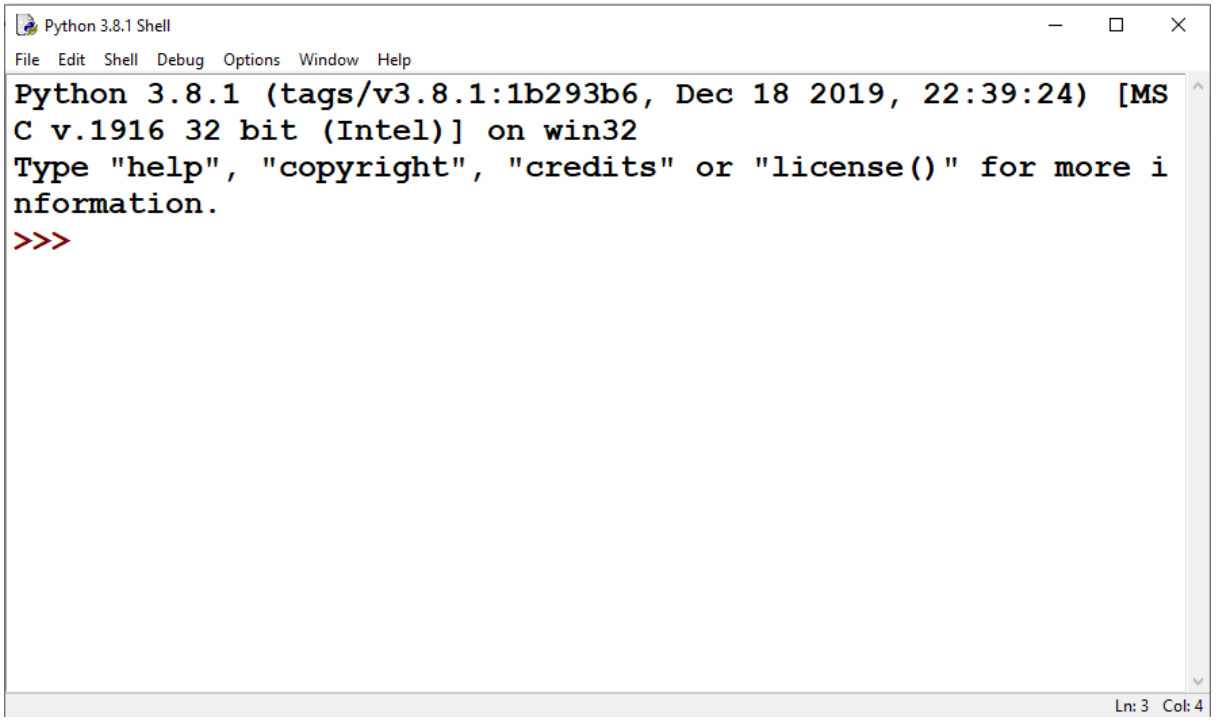
1.1 Python

Voľne stiahnuteľný - https://www.python.org/downloads/

Po inštalácii spúšťame cez idle.



1. Konzola



Za >>> napíš a potvrď ENTERom:

125	125+2	125//2 – div
125*2	125/2	125%2 - mod

1.2 Základné údajové typy v Python

Celé čísla

- int ako celé čísla, napr. 0, 1, 15, -123456789, …

Reálne čísla, desatinnú časť oddeľujeme . nie ,

- float ako desatinné čísla, napr. 0.0, 3.14159, 2.0000000001, 33e50, …

Reťazce znakov - udávané v ‘’alebo “”. Môžu obsahovať ľubovoľné znaky.

- str ako znakové reťazce, napr. 'a', "abc", '', "I'm happy"

1.2.1 int - základné celočíselné operácie

oba operandy musia byť celočíselného typu

- + súčet, napr. 1 + 2 má hodnotu 3
- - rozdiel, napr. 2 - 5 má hodnotu -3
- * násobenie, napr. 3 * 37 má hodnotu 111
- // celočíselné delenie, napr. 22 // 7 má hodnotu 3
- % zvyšok po delení, napr. 22 % 7 má hodnotu 1
- ** umocňovanie, napr. 2 ** 8 má hodnotu 256

1.2.2 float - základné operácie s desatinnými číslami

aspoň jeden operand musí byť desatinného typu (okrem delenia /)

- + súčet, napr. 1 + 0.2 má hodnotu 1.2
- - rozdiel, napr. 6 - 2.86 má hodnotu 3.14
- násobenie, napr. 1.5 * 2.5 má hodnotu 3.75
- / delenie, napr. 23 / 3 má hodnotu 7.666666666666667
- `//` delenie zaokrúhlené nadol, napr. `23.0 // 3` má hodnotu `7.0`
- `%` zvyšok po delení, napr. `23.0 % 3` má hodnotu `2.0`
- `**` umocňovanie, napr. `3 ** 3.` má hodnotu `27.0`
- x**(1/n) n-ta odmocnina z x

float - funkcie

- abs(cislo)

- round(cislo, pocet): zaokrúhľovanie - round, round(3.8)=4, round(3.4)=3, round(482,-1)=480, round(482, -2)=500. Záporné hodnoty určujú na desiatky/stovky. Round(a,2) - zaokrúhli na 2 desatinné miesta

1.2.3 str - operácie so znakovými reťazcami

`+` zreťazenie (spojenie dvoch reťazcov), napr. `'a' + 'b'` má hodnotu `'ab'`

`*` viacnásobné zreťazenie toho istého reťazca, napr. `3 * 'x'` má hodnotu `'xxx'`

ord`(a)` - ordinálna hodnota znaku ktorý je uložený v a

chr`(97)` - premení číslo 97 na znak - ‘a’

s.upper() - metóda - a=’a’ a=a.upper() - spraví z malého veľké aj pre celu vetu

s.lower()-z veľkých na malé.

len(a) - vráti dĺžku reťazca

in - zistí či sa nachádza podreťazec v reťazci, vráti true alebo false

rez reťazca

Video na Youtube:

1. https://youtu.be/XLkZxJPwOes

s=’ahoj milacik’ s[2:6] - ‘oj m’ ak nedáme s[:6] - od začiatku po 6 alebo s[2:] - od druhého(treti znak) po koniec

v rezoch môže byt aj [2:7:2] tretí je krok o koľko.

meno[::-1] – od konca - musíme ísť aj s indexami od väčšieho po menšie.

Rezy sa dajú využiť aj pri zoznamoch (poliach)

1.3 Tvorba premenných v Python

Názov premennej môže byť ľubovoľný, mal by vystihovať obsah premennej, mali by sme sa vyhnúť slovám, ktoré sú príkazy Pythonu.

Premenné používame na to, aby sme si v priebehu programu mohli pamätať hodnoty, ktoré používame, alebo hodnoty, ktoré budeme potrebovať. Hodnoty premenných môžeme v behu programu meniť.

Pri tvorbe premennej používame príkaz priradenia **=** ktorý znamená, do premennej, ktorá je na ľavej strane uloží hodnotu, ktorá je na pravej strane.

`a=125` - do premennej a uloží hodnotu 125

Napíšme a do konzoly vypíše nám hodnotu 125.

Ak chceme zistiť typ premennej napíšeme **type (a)** – vypíše nám <class 'int'>

poznáme 3 typy premenných:

`int` sú **celé čísla**, napr. `0`, `1`, `15`, `-123456789`, …

`float` sú **desatinné čísla**, napr. `0.0`, `3.14159`, `2.0000000001`, `33e50`, …

`str` sú **znakové reťazce**, napr. `'a'`, `"abc"`, `''`, `"I'm happy"`

Priklad:

Do premennej *m* priraďte svoje meno a nechajte ho napísať 1000 krát na obrazovku

```
m=’tomas’
m*1000
```

Do premennej p priraďte slovo ‘ahoj’ a spojte ju s premennou s vašim menom.

p+m

po prípade s medzerami (' '+m+' '+p+' ') a ak chceme 100 krát - (' '+m+' '+p+' ')*100

ak chceme zistiť typ premennej použijeme príkaz `type(názov_premennej)`

Aktualizácia premenných

- meno_premennej += hodnota # meno_premennej = meno_premennej + hodnota sucet
- meno_premennej -= hodnota # meno_premennej = meno_premennej - hodnota
- meno_premennej *= hodnota # meno_premennej = meno_premennej * hodnota
- meno_premennej /= hodnota # meno_premennej = meno_premennej / hodnota
- meno_premennej //= hodnota # meno_premennej = meno_premennej // hodnota - div
- meno_premennej %= hodnota # meno_premennej = meno_premennej % hodnota - mod
- meno_premennej **= hodnota # meno_premennej = meno_premennej ** hodnota - mocnina

Príklady v konzole

```
m=25
m+=1 >>> 26
m+=m >>>52
m = a = b = 0
a, b = 15, ‘ahoj‘
```

```
>>> a-15, b-ahoj
a, b= b, a >>> výmena obsahu
x,y=5,6
x, y = y, x+y>>> 6, 11
```

`meno, x, y = 'A', 180, 225` / v premennej meno bude hodnota ´A´ v x bude 180 a v y 225

`r = 'bod {} na súradniciach ({},{})'.format(meno, x, y)` // zátvorky nahradí hodnotami `meno, x, y`

`r`

'bod A na súradniciach (180,225)'

1.4 Vstup a výstup

Príkaz na načítanie hodnoty z klávesnice **príkaz vstupu** sa nazýva `input()`, všetky hodnoty, ktoré pomocou neho načítame sú typu `str - reťazce znakov` a preto ich podľa potreby pretypovávame príkazmi `int - celé čísla, float - reálne čísla,` každú hodnotu načítanú z klávesnice cez príkaz input musíme potvrdiť ENTERom.

Príklady:

`p = input (‘nepovinny text do vystupu uzivatelovi‘)` - načíta hodnotu typu `str`

`cislo=int(input(‘zadaj cele cislo’))` načíta celé číslo

`cislo=float(input(‘zadaj realne cislo’))`

Príkaz slúžiaci na vypisovanie textov, alebo hodnôt na obrazovku sa nazýva `print()`, vypisovaný text ohraničíme apostrofmi/úvodzovkami a výpis hodnoty premennej oddelíme od textu čiarkou dole.

príklady:

`print(p)` - vypíše hodnotu premennej `p`

`print(round(a),' ',b)` - vypíše zaokrúhlenú hodnotu premennej `a` potom medzeru a hodnotu premennej `b`

`print(round(a,2))` - vypíše zaokrúhlenú hodnotu premennej `a` na dve desatinné miesta

`for i in range(10):`

`print(i, end=' ')` – ostanu v riadku a oddelene medzerou

`print('195',"\n",' ahoj')` – `odriadkuje "\n"` – odriadkuje akoby vo výpise stlačil ENTER

Príklady:

1. Vytvorte program ktorý vypíše na obrazovku `Ahoj Svet`
2. Vytvorte program, ktorý po načítaní strany štvorca vypíše jeho obvod a obsah.
3. Vytvorte program, ktorý vypočíta obvod a obsah kruhu. Pi môžete použiť ako konštantu 3.14.
4. Vytvorte program, ktorý načíta počet sekúnd a vypíše koľko je to hodín a minút.

1.5 Podmienky

Slúžia na vetvenie chodu programu. Pomocou logických podmienok napr. a>b vyhodnotia či je podmienka pravdivá lebo nie.

Ak je podmienka pravdivá (True) vykonajú sa príkazy, ktoré sú napísané hneď pod if, ak podmienka nie je splnená (False) vykonávajú sa príkazy, ktoré sú napísané pod else.

1.5.1 Odsadenie príkazov

Všimnite si, že po : za podmienkou alebo po else sa príkazy odsunú do prava, to znamená patria tomuto príkazu. Python totiž nemá begin | end, alebo { | } ako začiatok a koniec príkazov pre danú štruktúru, ale podľa odsadenia určuje k čomu dané príkazy patria. Toto odsadenie sa dá nastaviť v configure IDLE, ale je to veľkosť jedného stlačenia klávesu Tab.


```
if podmienka:    # ak podmienka platí, vykonaj 1. skupinu príkazov
   prikaz
   prikaz
   ...
else:            # ak podmienka neplatí, vykonaj 2. skupinu príkazov
   prikaz
   prikaz
   …
```

body < 90	je menšie ako
body <= 50	je menšie alebo rovné
body == 50	rovná sa
body != 77	nerovná sa
body > 100	je väčšie ako
body >= 90	je väčšie alebo rovné
40 < body <= 50	je väčšie ako … a zároveň menšie alebo rovné …
a < b < c	a je menšie ako b a zároveň je b menšie ako c

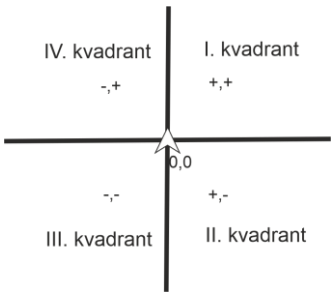
1.5.2 Podmienky môžeme spájať AND, OR:

AND - logický súčin True vyhodnotí vtedy ak sú všetky podmienky vyhodnotené ako True, ak čo i len jedna je False, celkový výsledok dáva False.

OR - logický súčet, True vyhodnotí už vtedy, čo i len jedna podmienka je splnená.

Príklady:

1. Zistite či je zadané číslo párne.
2. Zistite, či výťah unesie človeka, ak poznáme váhu človeka a nosnosť výťahu.
3. Načítajte 2 čísla a zistite či sa rovnajú, alebo ktoré z nich je väčšie.
4. Zistite či sa dá zostrojiť trojuholník ak poznáte jeho tri strany.
5. Predstavte si že ste v 2D priestore, vaša poloha je na súradniciach 0,0. Načítajte súradnice x,y v ľubovoľnom rozsahu -/+ celých čísel - predstavujú asteroid a určite z ktorého kvadrantu na vás asteroid letí.



1.6 ELIF

Ak potrebujeme viacnásobné vetvenie, alebo teda rozdeliť chod programu podľa viacerých hodnôt, môžeme použiť príkaz `elif.`

Ako prvé vyhodnotenie použijeme podmienku `if` a ďalšie podmienky začíname na úrovni `if`, ale s príkazom `elif.`

```
if podmienka_1:      # ak podmienka_1 platí, vykonaj 1. skupinu príkazov
   prikaz
   ...
elif podmienka_2:    # ak podmienka_1 neplatí, ale platí podmienka_2, ...
```

```
    prikaz
    ...
elif podmienka_3:       # ak ani podmienka_1 ani podmienka_2 neplatia, ale platí podmienka_3, ...
    prikaz
    ...
else:                   # ak žiadna z podmienok neplatí, ...
    prikaz
    ...
```

Príklady:

1. Vytvorte program, ktorý pre zadané číslo zistí koľko je ciferné. Zadanie vyriešte do 5 cifier.
2. Vytvorte program, ktorý si vypýta vek a zaradí človeka do jednej z kategórií:
    - `Slobodný 0 - 5`
    - `Školák 6 - 13`
    - `Teenager 14 - 19`
    - `Produktívny vek 20 - 50`
    - `Už máš rozum 51 - 70`
    - `Ako víno 71 - 99`
    - `Astrálny vek 100`

1.7 Cyklus While

Cyklus s neznámym počtom opakovaní, počet opakovaní zabezpečuje podmienka.
Ak má podmienka hodnotu True vykonávajú sa príkazy v tele cyklu.
Ak podmienka nadobudne hodnotu False. Príkazy v tele cyklu sa preskočia a pokračuje chod programu pod cyklom.
Treba si uvedomiť, že podmienka nehovorí, kedy má cyklus skončiť, ale naopak - kým podmienka platí, vykonávajú sa všetky príkazy v tele cyklu.

```
While podmienka :

     príkaz 1

     príkaz 2

     …
```

1. **Príklad: Nakladanie auta**

Vytvorte program, ktorý vás bude neustále informovať o náklade vozidla počas nakladania. Ak náklad prekročí nosnosť informuje o tom nakladača a ukončí nakladanie. Treba umožniť nakladačovi ukončiť nakladanie, aby neprekročil nosnosť vozidla.

```
nosnost=6000
naklad=0
ukoncenie=0
while (naklad < nosnost) and (ukoncenie != 1):
          print('mozes prilozit',nosnost - naklad,' kg')
          print('priloz naklad')
          naklad += int(input())
          print('aktualna vaha:',naklad)
          ukoncenie=int(input('skoncenie   nakladania   -   1,
pokracovat hcc + Enter'))
```

2. **Príklad: Najväčší spoločný deliteľ**

1. ak sa čísla nerovnajú
2. zistíme ktoré je väčšie a od toho väčšieho odpočítame to menšie

Opakujeme oba kroky, pokiaľ sa čísla nerovnajú

```
a=int(input('zadaj cislo 1'))
b=int(input('zadaj cislo 2'))
x=a
y=b
```

```
while x != y:
          if x > y:
                    x=x-y
          else:
                    y=y-x
print('NSD ',a,'a',b,' je ',x)
```

Príklad: Cifry čísla:

Videa na Youtube:

1. **https://youtu.be/30fuQjtwla4**
2. **https://youtu.be/Hgmgi1a5NkM**

125 % 10 = 5 – vráti nám poslednú cifru čísla

125 // 10 = 12 – vráti nám číslo bez poslednej cifry čísla (zahodí poslednú cifru)

Príklad zistí koľko ciferné je číslo. Zisťujeme, koľko krát odtrhneme poslednú cifru čísla – n=n//10 (odtrhne/zahodí poslednú cifru čísla) odtŕhame poslednú cifru pokiaľ sa číslo n väčšie ako 0 – ak sa stane 0 odtrhli sme všetky cifry.

```
n=int(input('zadaj cislo'))
pocet_cifier=0
while n > 0:
     n=n//10
     pocet_cifier += 1
print('zadal si ',pocet_cifier,' ciferne cislo')
```

Cifry čísla 2:

Upravte program:

- Zistil ciferný súčet čísla napr. 635 = 14
- Zistite počet párnych a nepárnych cifier v čísle
- Zistite či je zadané číslo palindróm - otočte číslo a zistite či je rovnaké ako pred otočením.

druhá odmocnina

```
cislo = float(input('zadaj cislo:'))
x = 1
while x ** 2 < cislo:
    x = x + 0.001
print('odmocnina', cislo, 'je', x)
```

3. **Príklad súboj:**

Vytvorte program, ktorý bude emulovať súboj. Potrebujete vedieť život hrdinu a život príšery. Budete náhodne generovať silu úderu hrdinu a príšery. Následne odpočítate silu úderov od životov a opakujete, pokiaľ sú životy väčšie ako 0.

Vylepšenia:

- Pred začiatkom súboja hoďte kockou, či a koľko budete mat možností doplniť život. Dopĺňanie umožnite po odčítaní života.
- Pred začatím hoďte kockou kto bude útočiť prvý.
- Pred odčítaním života, hoďte kockou, či úder nebol zablokovaný.
- Program prerobte do tkinter

Vykonávanie cyklu môžeme ukončiť aj príkazom break. Cyklus sa ukončí a pokračuje ďalej chod programu mimo cyklu. Najčastejšie sa používa v spolupráci s if. While True - večný cyklus, kde sme jeho ukončenie zabezpečili príkazom break ak užívateľ zadá 0. V programovaní sa večným cyklom snažíme vyhnúť, je nešťastné riešenie ak stlačíte klávesu na pohyb do prava a postava pôjde už navždy do prava … preto máme zabezpečenie ukončenie cyklu pomocou break.

```
while True:
     cislo = int(input('zadaj cislo: '))
     if cislo % 2 == 0:
          print('parne')
     else:
          print('neparne')
     if cislo == 0:
          break
```

1.8 Cyklus for

Videa na Youtube:

1. https://youtu.be/OUBpUYII_9M
2. https://youtu.be/JM1_f8c3za4

Cyklus for je cyklus so známym počtom opakovaní, to znamená vieme dopredu povedať počet, koľkokrát sa príkazy musia vykonať.

Príkazy, ktoré patria cyklu a majú sa v ňom vykonávať musia byť odsadené pod cyklom.

Cyklus for sa zapisuje:

```
for i in range(10):
    prikaz 1
    príkaz 2
    …
```

for - kľúčové slovo, ktoré určuje, že sa jedná o cyklus for

i - riadiaca premenná, ktorá nadobúda hodnoty, ktoré sa určia v range.

in - v neskôr si povieme, že cyklus for nemusí ísť len po číslach v range, ale aj po ľubovoľnej štruktúre, ktorá má poradie.

range - určuje rozsah a krok ako bude cyklus fungovať, hodnoty, ktoré nastavíme v range, nadobúda riadiaca premenná i

príklady:

range(10) - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

range(0,10) - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

range(5,10) - 5, 6, 7, 8, 9

range(2,10,2) - 2, 4, 6, 8

range(10, 2,-1) - 10, 9, 8, 7, 6, 5, 4, 3

range(začiatok, koniec, krok)

Ak dáme do range len jednu hodnotu, vždy to berie tak, že ideme od 0 po zadanú hodnotu, ktorú nedosiahne, vždy nadobudne hodnotu o 1 menej.

Hodnota, ktorá je napísana ako koniec, nikdy nie je nadobudnutá, vždy nadobudne o jednu menej.

Hodnoty cyklu môžu ísť aj opačne, od väčšieho čísla smerom k menším, ale v tom prípade musí byť záporný krok a začiatok väčší ako koniec.

Príklady:

1. Vytvorte cyklus ktorý vygeneruje čísla 5 - 10

```
for i in range(5,11):
    print(i)
```

2. Vytvorte cyklus, ktorý vygeneruje všetky párne čísla 2 - 22 vrátane 22.
3. Vytvorte cyklus, ktorý vygeneruje čísla 50 - 5.
4. Vygenerujte násobky čísla 7 v rozsahu 70 - 7.
5. Vygenerujte čísla 10, 20, 30, 40, 50, 60, 70 … 150
6. Vytvorte program, ktorý vypíše malú násobilku pre zadané číslo.
7. Vynásobte 2 čísla bez použitia násobenia, použite súčet a cyklus.

8. Vytvorte program pre n!.
9. Vytvorte n členov fibonacciho postupnosti 0,1,1,2,3,5,8,13… prvý člen = 0 druhý člen 1, nasledujúci člen získate súčtom 2 predchádzajúcich.

Premenná cyklu nemusí nadobúdať len celočíselné hodnoty. Cyklus for vie ísť aj po hodnotách str, zoznamoch a súboroch.

príklad zoznam - []:

```
z=[5,”ahoj”,98,99,69,”klasika”]
for hodnota in z:
     print(hodnota)

5
ahoj
98
99
69
klasika
```

príklad str

```
slovo=”Ahojte”
for pismeno in slovo:
     print(pismeno)
```

príklad file - vstup.txt a zdrojový súbor python musia byť v rovnakom priečinku

```
f=open(’vstup.txt’.’r’)
for riadok in f:
     print(f)
```