Programko.sk

Python 1

ZAČÍNAME S PYTHONOM

Python je vysoko-úrovňový jazyk (rovnako ako C++ alebo Java). To, aspoň pre nás, znamená, že má relatívne jednoduchú syntax, teda je čitateľný a ľahko sa učí. Tieto jazyky sa potom kompilujú alebo interpretujú na jazyk strojový, ktorý sa skladá z jednotiek a núl. Strojový kód je potom prečítaný počítačom a vykonaný. Python je taktiež jazyk interpertovaný, teda sa prekladá do strojového kódu za behu programu, čo mierne uberá na jeho rýchlosti, ale zasa iné výhody, ako napríklad jednoduchú prácu, či rýchle písanie jednoduchých skriptov.

INŠTALÁCIA PYTHONU / IDLE – VERZIA 3.8.2

Stránka: https://www.python.org/downloads/release/python-382/

1. Prescrollujeme na úplny spodok stránky a vyberieme váš operačný systém (pre windows: „Windows x86-64 executable installer“, pre mac: „macOS 64-bit installer“)

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		f9f3768f757e34b342dbc06b41cbc844	24007411	SIG
XZ compressed source tarball	Source release		e9d6ebc92183a177b8e8a58cad5b8d67	17869888	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	f12203128b5c639dc08e5a43a2812cc7	30023420	SIG
Windows help file	Windows		7506675dcbb9a1569b54e600ae66c9fb	8507261	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	1a98565285491c0ea65450e78afe6f8d	8017771	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	b5df1cbb2bc152cd70c3da9151cb510b	27586384	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	2586cdad1a363d1a8abb5fc102b2d418	1363760	SIG
Windows x86 embeddable zip file	Windows		1b1f0f0c5ee8601f160cfad5b560e3a7	7147713	SIG
Windows x86 executable installer	Windows		6f0ba59c7dbeba7bb0ee21682fe39748	26481424	SIG
Windows x86 web-based installer	Windows		04d97979534f4bd33752c183fc4ce680	1325416	SIG

2. Následne sa vám stiahne inštalačný súbor, ktorý spustíme

3. (Windows inštalátor) vyberieme „Install Now“

4. Keď sa dokončí inštalácia, klikneme „Close“

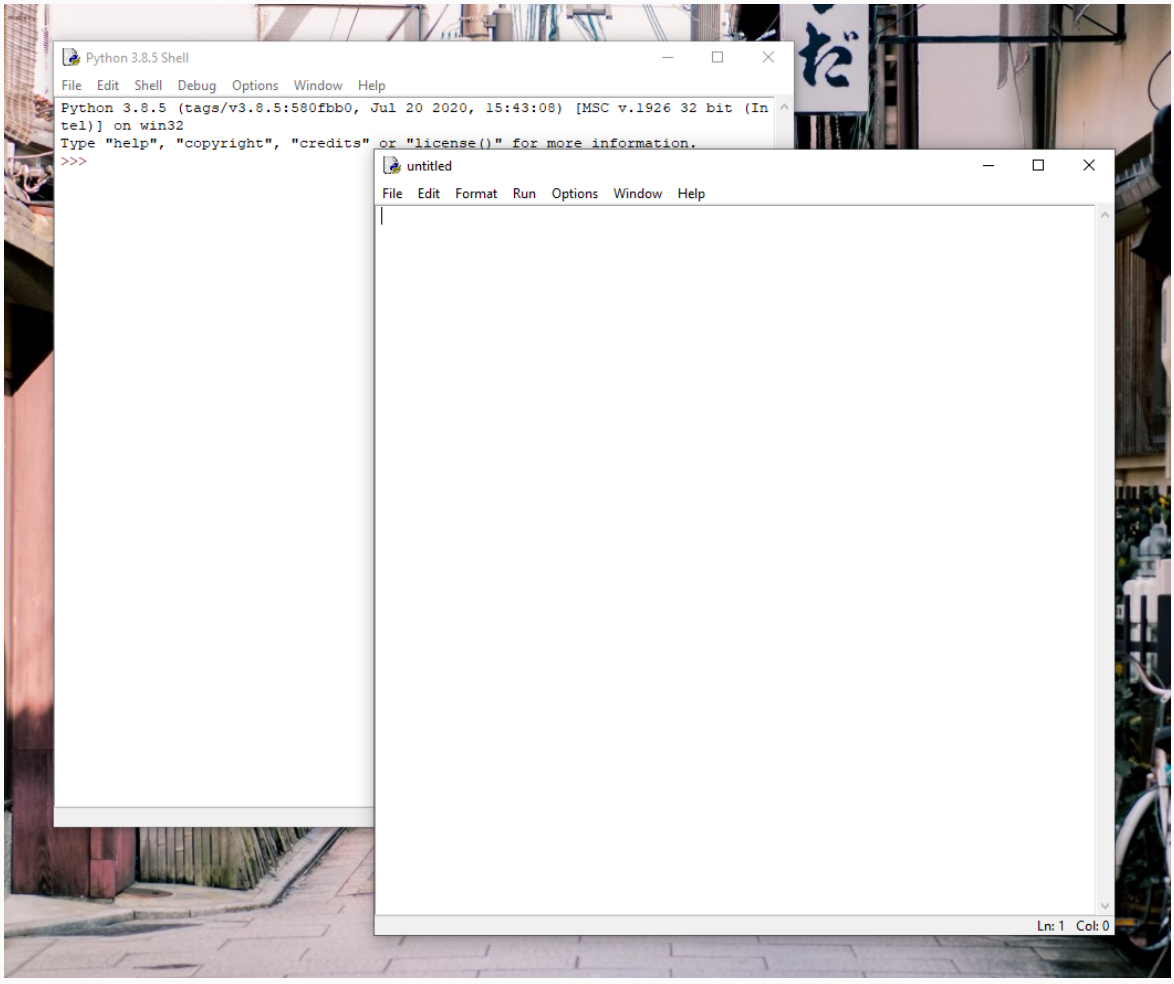
5. Teraz by ste mali mať na počítači nainštalovaný python spolu s jednoduchým IDE (Integrated development environment – prostredie kde sa píše kód programovacieho jazyka), nayvaným IDLE

6. Nájdite program IDLE, buď cez štart alebo na pracovnej ploche, mal by mať meno podobné „IDLE (Python 3.8)“

VYTVORENIE NOVÉHO PROGRAMU

1. Keď máme otvorené IDLE, klikneme na File – New File

2. Malo by sa vám otvoriť nové, prázdne okno s názvom untitled



3. Sem začneme písať náš kód. Pred jeho spustením však musíme súbor uložiť (ctrl + s a vyberieme miesto kam ho uložíme), následne program spustíme tlačítkami Run – Run Module, alebo stlačením klávesy F5

PRVÝ PROGRAM – HELLO WORLD!

1.1 Príklad

Vypíšte na riadok výstupu v konzole reťazec: Hello World!

Použijeme príkaz:

print() - vypíše daný text (to čo je v zátvorkách – nazývaný ako vstup) na konzolu (nazývaný ako výstup) a posunie kurzor na ďalší riadok

PRÍKAZ INPUT()

1.2 Príklad

Nechajte užívateľa zadať jeho meno a vypíšte

reťazec: “Meno“ je super.

Použijeme príkaz:

input() - žiada vstup od užívateľa na konzole po stlačení klávesy enter sa vstup pošle do programu, vracia string (reťazec)

TYPY HODNÔT A ZAKLADNÉ OPERÁCIE

Každá hodnota v jazyku python má nejaký dátový typ, aby jazyk vedel ako s ňou narábať. Napríklad „Hello World!“ z prvej úlohy je typu string, teda reťazec znakov. Okrem string-u existuje kopa ďalších, nám však pre začiatok budú stačiť tieto základné:

- **int** - integer - celočíselné číslo
- **float** - floating point - desatinné číslo
- **str** - string, reťazec - slovo veta atď..
- **bool** – Boolean - logická hodnota, má stavy True alebo False

Vyskúšajte

print(3)

print(‘3’)

Všimnite si rozdiel v týchto dvoch zápisoch. Prvý vypisuje číslo, druhý reťazec.

Základné matematické operácie v pythone fungujú rovnako ako v matematike, i keď niektoré, pre jednoduchší zápis na klávesnici sa zapisujú trocha inak ako môžete byť zvyknutí. Toto sú tie ktoré budeme používať:

- **+ sčítanie 2 + 3** má hodnotu 5
- **- odčítanie 3 – 2** má hodnotu 1
- *** násobenie 2 * 3** má hodnotu 6
- **/ desatinne delenie 5 / 2** má hodnotu 2.5
- **// celočíselné delenie 22 // 7** má hodnotu 3
- **% zvyšok po delení 22 % 7** má hodnotu 1
- **** umocňovanie 2 ** 8** má hodnotu 256

Vyskúšajte

Vyskúšajte si postupne vypísať, s pomocou príkazu print(), rôzne aritmetické operácie pre rôzne čísla, aby ste sa utvrdili v tom ako fungujú.

NEPOVINNÉ PARAMETRE

Niektoré funkcie, ktoré v kóde voláme potrebujú na spustenie nejaké premenné (napríklad print() potrebuje string alebo číslo, ktoré sa vypíšu). Tieto premenné nazývame parametre. Okrem toho existujú aj nepovinné parametre ktoré majú nejakú pre-definovanú hodnotu, a teda ich môžeme, ale nemusíme zadávať.

Príkazy ktoré už poznáme:

print('Ahoj číslo na ktoré som myslel bolo: ', 3, sep='', end='')

Print vieme rozdeliť na 3 časti. Každá táto časť je rozdelená čiarkou. Prvá časť je string druhá int, 3. a 4. sú nepovinné parametre. Sep (ako separátor) je to string ktorý ma print napísať namiesto čiarok z vstupu (v našom príklade nemá napísať nič ale defaultne to býva medzera) a end je string ktorý sa má dať na koniec (v našom prípade tam nemá dať nič ale defaultne tam je symbol konca riadka takzvaný '\n')

age = input('Zadaj svoj vek prosím: ')

Inputu môžeme takisto nepovinne zadať string, ktorý sa vypíše ako správa užívateľovi, predtým ako si program vypýta nejaký vstup. Väčšinou daný string býva správa, ktorá užívateľovi povie, čo má zadať. V tomto prípade bude program vyzerať tak, že program najprv vypíše "Zadaj svoj vek prosím: ", následne počká na užívateľa kým niečo zadá a potom to uloží do premennej age.

Vyskúšajte

Vyskúšajte si rôzne variácie príkazu print/input, zadajte rôzne hodnoty pre sep/end, aby ste sa zoznámili s tým ako fungujú

Napríklad:

```
print('Ahoj svet', end='!')
print('tieto', 'slová', 'sú', 'oddelené', 'pomlčkami', sep='-')
x = input('Zadaj x:')
print(x, ' je tvoje zadané číslo')
```

FOR CYKLUS

Občas narazíme na situácie, kedy je potreba zopakovať nejaký úkon niekoľkokrát za sebou. Pre tieto účely môžeme použiť for cyklus. Pre použitie for cyklu sú potreba 2 veci: riadiaca premenná a prvky, ktoré prechádzame. Program, ktorý chceme zopakovať označujeme ako telo for cyklu a musí byť odsadené (klávesou tab). For cyklus postupne priradzuje prvky for cyklu do riadiacej premennej a spustí svoje telo toľko krát koľko definujeme. V nasledujúcom príklade používame riadiacu premennú i a prvky, ktoré prechádzame sú definované pomocou range(10). For cyklus v nasledujúcom príklade postupne vypíše čísla od 0 do 9.

Vyskúšajte

```
for i in range(10):
 print(i)
```

Tento kód vypíše čísla od 0 po 9 pod seba

RANGE

Je príkaz ktorý typicky používame vo forcykle. Definujem postupnosť čísel, ktorú forcyklus prechádza. V najjednoduchšom prípade, range(10), prechádza čísla od 0 po 9, teda cyklus zopakuje 10 krát.

range(start, stop, step) - Vracia postupnosť čísiel začínajúc od štart (ak nezadáme nič tak od 0) zväčšujúca sa o step (ak nezadáme nič tak o 1) a končí na čísle definované stop, čo je jedinný povinný parameter.

To znamená že ak napíšeme príkaz range(10) zadávame vlastne iba parameter stop, teda pokiaľ má range siahať.

Vyskúšajte

```
range(10)
range(2, 5)
range(2, 11, 1)
range(4, 100, 2)
range (10, 0, -1)
```

Vyskúšate si rôzne variácie range

(zadaním do forcyklu ako v predošlom príklade) a sledujte aké čísla vám jednotlivé postupnosti vypíšu, aby ste sa zoznámili s tým ako range funguje

PRÍKLADY

2.1 Príklad

Vytvorte program, ktorý si vypýta od užívateľa jeho meno
(so správou aby užívateľ vedel čo má zadať) a následne vypíše toto meno.

2.2 Príklad

Za pomoci for cyklu napíšte program ktorý do riadku vypíše prvých n čísel od jedna.

Teda pre N = 10 vypíše:

1 2 3 4 5 6 7 8 9 10

Je treba použiť, for cyklus, range a print, s tým že range treba upraviť, aby nezačínala od 0 ale od 1.

2.3 Príklad

Vytvorte program, ktorý dostane od užívateľa slovo, uloží si ho do premennej a

následne ho 10 krát vypíše do riadku a dá medzi každé slovo čiarku

Treba použiť for cyklus a print so správne zadaným parametrom end

2.4 Bonusový príklad

Vylepšite príklad 2.1 tak, aby užívateľ zadal aj číslo, koľko krát sa má dané slovo vypísať a následne ho for cyklom toľko krát vypíšete.

Hint: ak z **input()** získame číslo, uloží sa nám do premennej ako string a teda ho potom neviem použit vo funkcii **range()**, vypíše nám to error. Aby sme daný string pretypovali na integer, musíme použiť funkciu **int()**, teda napríklad, ak máme premennú cislo napíšeme **range(int(cislo))**.

PÍSANIE KÓDU DO BLOKOV

V predošlej hodine sme sa učili ako používať for cyklus, a ako ste si určite všimli, kód vnútri musí byť odsadený klávesou tab. Toto je z dôvodu že python musí nejak rozoznať, ktorý kód patrí k čomu. V prípade for cyklu, aby vedel ktorý kód sa má opakovať vo for cykle x krát a ktorý už nie. Neskôr budeme odsadzovanie kódu používať aj napríklad v podmienkach, či funkciách, na teraz si ho natrénujeme pri for cykloch. Ako sme už spomínali, odsadzujeme tabom, čo je presná hodnota ktorú interpreter potrebuje aby chápal, či je kód odsadený alebo nie.

Vyskúšajte

Odsadenie klávesou tab:

```
for i in range(10):
    print("Tento kód funguje")
```

Odsadenie jednou medzerou:

```
for i in range(10):
 print("Tento kód nefunguje")
 print("Tento kód nefunguje")
```

Druhý príklad vám vyhodí error, pretože si interpreter nebude istý kam jednotlivé príkazy patria

Vyskúšajte

```
for i in range(10):
    print("Tento kód sa opakuje")
    print("Tento kód sa taktiež opakuje")

for i in range(10):
    print("Tento kód sa opakuje")
print("Tento kód sa už neopakuje")
```

Vyskúšajte si napísať rôzne kódy odsadené tabom, aby ste sa zoznámili s tým, ako presne blokovanie funguje. Teoreticky môže byť v jednom for cykle koľko riadkov len chcete. Dôležité je, aby boli všetky správne odsadené.

PRÍKLADY

3.1 Príklad

Napíšte program, ktorý si vypýta od užívateľa dve slová (musíte dvakrát použiť príkaz input) a vypíše ich 5 krát, potom vypíše reťazec „medzera medzi for cyklami“ a následne zadané slová vypíše ešte 5 krát. T.j. musíte použiť dva for cykly.

3.2 Príklad

Napíšte for cyklus, ktorý si v každej iterácii vypýta od užívateľa slovo a následne ho vypíše. For cyklus môže bežať ľubovoľne veľa krát.

VYUŽÍVANIE RIADIACEJ PREMENNEJ FOR CYKLU

Určite ste si všimli, že vo forcykloch vždy zadávame nejakú premennú (väčšinou nazvanú i, ale môžeme ju nazvať ako len chceme). Táto premenná drží informáciu o tom v koľkátom cykle for cyklu sme. Okrem toho, ju vieme využiť na rôzne výpočty a vďaka funkcionalite range, má táto premenná veľa využití.

Vyskúšajte

Suma prvých desať čísel:

```
suma = 0
for i in range(10):
    suma = suma + i
    print(suma)
```

Výpis párnych čísel od 0 po 10

```
for i in range(0, 11, 2):
    print(i)
```

Poskúšajte rôzne variácie range a pripomeňte si ako funguje

3.3 Príklad

Napíšte for cyklus, ktorý vypíše všetky nepárne čísla od 0 po 10 (veľmi podobne ako v predchádzajúcom príklade)

3.3.1 Bonusový príklad

Vylepšite príklad 3.3 tak, aby užívateľ zadal aj číslo, pokiaľ má program čísla vypisovať. Teda ak užívateľ zadá 5, budú sa vypisovať nepárne čísla od 0 po 5

Hint: ak z **input()** získame číslo, uloží sa nám do premennej ako string a teda ho potom neviem použit vo funkcii **range()**, vypíše nám to error. Aby sme daný string pretypovali na integer, musíme použiť funkciu **int()**, teda napríklad, ak máme premennú **cislo = input('Zadaj číslo >> ")** napíšeme **range(int(cislo))**.

3.4 Príklad

Napíšte for cyklus, ktorý vynásobí všetky čísla od 1 po 10 a vypíše tento výsledok

Hint: dávajte si pozor, treba správne upraviť range, ak necháme defaultne **range(11)**, prvé číslo bude vždy nula a ak násobíme nulou, vyjde nám vždy na konci nula.

PODMIENKY - IF

IF pri programovaní používame ak chceme aby program vybral čo sa má stať ďalej na základe nejakej podmienky. V pythone sa pre túto potrebu využíva vetva if (po anglicky „ak“) a ak treba tak vetva else (po anglicky „inak“). Vetva if potrebuje zadať hodnotu typu boolean ktorá ak je true (pravda) tak sa vykoná kód vnútri nej a ak false (nepravda) tak sa kód jednoducho preskočí, prípadne sa vykoná vetva else.

Kód, vnútri if alebo else musí byť odsadený rovnako ako sme to robili pri for cykloch, aby python vedel čo kam patrí. Vyskúšajte

Jednoduchý if, ktorý vždy vypíše „Číslo = 5“

```
cislo = 5
if (cislo == 5):
      print("číslo = 5")
```

if ktorý reaguje na input od užívateľa

```
slovo = input("zadajte slovo: ")
if slovo == "auto":
      print("zadané slovo je 'auto'")
else:
      print("zadané slovo nie je 'auto'")
```

Všimnite si, že za podmienku vždy píšeme **==**, narozdiel od priradovania do premenej so znakom **=**. Jednoduché = totiž vždy znamená priradenie do premennej, a nevracia žiadnu hodnotu. **Operátor == je equivalent opýtania sa „Rovná sa?“**, takže ak napíšeme napríklad **cislo == 5**, python sa pri príchode na danú podmienku spýta „Je hodnota v premennej číslo rovná 5?“, ak áno, prevedie sa kód vnútri, ak nie, prevedie sa vetva else (ak existuje) a program pokračuje ďalej.

== nazývame logický operátor, čo je operátor ktorý nám vracia hodnotu boolean.

LOGICKÉ OPERÁTORY

a < 50	je true (pravda), ak a (premenná) je menšie ako 50
a <= 50	je true, ak a je menšie alebo rovné ako 50
a >= 50	je true, ak a je väčšie alebo rovné ako 50
a == 50	je true, ak a je rovné 50
a != 50	je true, ak a nerovná sa 50
50 < a < 60	je true, ak a je medzi 50 a 60 (okrem 50 a 60)
50 <= a <= 60	je true, ak a je medzi 50 a 60 (vrátane 50 a 6)

Výrazov, ktoré vracajú hodnotu typu Boolean môžeme v jednej podmienke použiť aj viac a pre ich spojenie používame tieto kľúčové slová.

and ak je splnená podmienka na ľavo a aj na pravo tak platí, inak neplatí

50 < a and a < 60

or ak je splnená jedna z podmienok tak platí, inak neplatí

a < 50 or 60 < a

not negácia, ak podmienka neplatí tak negácia platí, a naopak

not a == 50 je to isté ako **a != 50**

Vyskúšajte

Vyskúšajte si rôzne podmienky, aby ste si osvojili prácu s logickými operátormi, napríklad:

```
cislo = int(input("zadajte číslo: "))
if 3 < cislo < 5:
      print("zadané číslo je menšie 5")
      print("zadané číslo je väčšie ako 3")
else:
      print("zadané číslo je väčšie ako 3 a menšie ako 5")

cislo1 = int(input("zadajte číslo: "))
cislo2 = int(input("zadajte číslo: "))
if cislo1 == 5 and cislo2 == 5:
      print("obe čísla sú 5")
else:
      print("obe čísla nie sú 5")
```

PRÍKLADY

4.1 Príklad

Napíšte program, ktorý si od užívateľa vypýta slovo, uloží ho do premennej. Ak je slovo „voldemort“, vypíše „psst toto meno nesmieme vypisovať“.

4.2 Príklad

Napíšte program, ktorý si od užívateľa vypýta dve slová (napríklad zadajte nové heslo a zopakujte heslo) ak sú slová rozdielne, vypíše „Heslá musia byť rovnaké“

4.3 Príklad

Napíšte program, ktorý si od užívateľa vypýta dve čísla. Ak je súčin týchto čísel väčší ako 1000, vypíše „ok.“, inak vypíše „nabudúce prosím zadajte väčšie čísla“.

Pripomienka: aby sme so zadanými číslami mohli robiť matematické operácie, musíme ich najprv zmeniť zo stringu na int a to príkazom **int()**, teda: **x = int(input(„Vaša správa“))**

4.4 Bonusový príklad - uhádni číslo

S malo pomocou od externej knižnice si môžete vytvoriť jednoduchú hru „Uhádni číslo“ Na začiatok kódu si napíšte tieto dva riadky:

```
from random import randint
ran = str(randint(1, 10))
```

Tieto riadky najprv nahrajú funkciu **randint** z externej knižnice random ktorú následne použijeme aby sme vygenerovali číslo medzi 1 a 10 (môžete si čísla zmeniť na akékoľvek chcete), následne je toto číslo pretypované na string a uložené do premennej ran.

Ako presnejšie ale fungujú na teraz nepotrebujete vedieť, dôležité je že nám táto funkcia vygeneruje náhodné číslo

Čo chceme teraz spraviť je, vypýtať si od užívateľa číslo so správou „Hádaj číslo: “, pomocou podmienky zistiť či je zadané číslo rovnaké ako číslo v premennej „ran“, ak áno vypísať „Uhádli ste!“ inak vypísať „Neuhádli ste“

FUNKCIE

Niekedy sa nám v kóde stane že potrebujeme použiť tie isté riadky kódu viacej krát na rôznych miestach. V takýchto prípadoch môžeme kód zjednodušiť tým že využijeme funkcie, čo nám jednak skráti ale aj zjednoduší kód a spraví ho čitateľnejším. Okrem toho, ak by sme v budúcnosti chceli tento kód upraviť, stačí nám ho zmeniť na jednom mieste.

S funkciami sme sa už stretli (napríklad print() či range()) a my sa teraz naučíme ako si zadefinovať vlastné. Budeme používať kľúčové slovo def ktoré je nasledované menom funkcie pomocou ktorého ju budeme volať.

Vyskúšajte

Vytvorte si funkciu s vlastným názvom

a skúste ju zavolať na rôznych miestach vo svojom kóde

```
def foo():
      print("Začiatok mojej funkcie")
      print("Koniec mojej funkcie")

print("\nPrvé volanie funkcie foo")
foo()
print("\nDruhé volanie funkcie foo")
foo()
```

Funkcií môžete vytvoriť ľubovoľne veľa, prípadne volať

jednu funkciu v druhej:

```
def foo():
      print("Začiatok mojej funkcie")
      print("Koniec mojej funkcie")

def foo2():
      foo()
      print("Začiatok mojej druhej funkcie")
      print("Koniec mojej druhej funkcie")

foo2()
```

PARAMETRE FUNKCIÍ

Aby funkcie vedeli spolupracovať s kódom okolo, vieme im poslať parametre. Takisto sme sa s nimi už stretli napríklad pri **print()**, ktorej posielame string, ktorý sa má vypísať, a ďalšie voliteľné parametre (k tým sa dostaneme nabudúce).

Funkcii musíme pri definícii určiť koľko parametrov bude prijímať.

Vyskúšajte

Táto funkcia potrebuje jeden parameter a následne ho vypíše

```
def vypis(retazec):
      print(retazec)

vypis("Vypis tento retazec")
```

Táto funkcia potrebuje dva parametre a vypíše ich súčet, prípadne zreťazenie ak sú to stringy.

```
def zrataj(cisloA, cisloB):
      print(cisloA + cisloB)

zrataj(5,  6)
zrataj("a", "b")
```

PRÍKLADY

5.1 Príklad

Napíšte funkciu **pozdrav(meno)** ktorej pošlete jeden parameter, meno, a ona ho pozdraví. Teda ak zavoláme funkciu **pozdrav(“Matej”)**, program vypíše napríklad **„Ahoj, Matej“**

5.2 Príklad

Napíšte funkciu **opakuj(slovo, pocet)** ktorej pošlete dva parametre, nejaké slovo a koľko krát sa má vypísať (Teda string a integer).
Teda ak zavoláme funkciu **opakuj("woop", 3)**, program vypíše:

```
woop
woop
woop
```

5.3 Príklad

Napíšte funkciu **poradie(n)** ktorej pošlete jeden parameter (integer). Táto funkcia potom vypíše všetky čísla od 1 po n ktoré jej bolo zadané.

5.4 Úloha na precvičenie

Napíšte funkciu **stvorec(a)**, ktorá vypíše štvorec so znakov **„#"** a kde a je dĺžka strany tohoto štvorca. Ak zavoláme **stvorec(4)**, python vykreslí:

```
####
####
####
####
```

5.4.1 Úloha na precvičenie

Upravte funkciu z predošlej úlohy tak, aby sme jej vedeli zadať aj znak z ktorého sa kreslí. Ak zavoláme **stvorec(3, "A")**, python vykreslí:

```
AAA
AAA
AAA
```

FUNKCIA RETURN

Ak chceme aby naša funkcia vracala nejakú hodnotu s ktorou by sme mohli potom ďalej pracovať, využijeme slovo **return** (po anglicky, vráť).

Napríklad vytvoríme funckiu ktorá nám bude rátať obvod štvorca, ale nechcem obvod iba vypísať, ale ho následne využiť v ďalšom výpočte, musíme použiť return.

Akonáhle funkcia narazí na príkaz return, hodnota sa vráti a ďalší kód funkcie sa už nevykoná.

Vyskúšajte

Vrátenú hodnotu si vieme potom uložiť do premennej a využiť ju neskôr v kóde

```
def obvod_stvorca(a): # a = dlzka strany
      return a*4

a = 4
o = obvod_stvorca(a)
print("Stvorec s dlzkou strany 4 ma obvod", o)
```

return môžeme použiť aj viacej krát, avšak akonáhle program na jeden narazí, funkcia vráti danú hodnotu a skončí, takže sa vždy vykoná iba jeden, a to prvý na ktorý program príde.

```
def je_parne(cislo):
      if (cislo % 2 == 0):
            return str(cislo) + " je parne"
      return str(cislo) + " nie je parne"

for i in range(100):
      print(je_parne(i))
```

PRÍKLADY

6.1 Príklad

Vytvorte funkciu **vylepsi(slovo)**, ktorej užívateľ pošle slovo, tá k nemu pripojí reťazec **„je super“** a vráti naspäť nový reťazec.

(Stringy vieme sčítavať rovnako ako čísla, teda napríklad **„toto a “ + „este toto“ nám vráti „toto a este toto“**)

6.2 Príklad

Vytvorte dve funkcie:

1. **je_pravouhly(alfa, beta, gamma)** prijíma uhly trojuholníka, ak ich súčet nemôže byť trojuholník (nie je 180), vráti false, ak áno, tak vráti true ak trojuholník je pravouhlý a false ak nie je.

2. **obsah_trojuholnika(a, b, alfa, beta, gamma)** potrebuje dve odvesny trojuholníka a jeho uhly. Následne za pomoci funkcie 1. zistí či je pravouhlý, ak nie je vráti 0 a ak je vyráta jeho obsah z a a b a vráti ho.

PRÁCA S REŤAZCAMI

Ako sme si už vyššie ukázali stringy dokážeme spájať pomocou +, okrem toho ich ale vieme aj násobiť a používať rôzne ďalšie funkcie.

Vyskúšajte

Spájať stringy sme si už vyskúšali vyššie, čo sa ale stane ak ich vynásobíme?

```
print("hello " * 6)
```

Ďalšie základné funkcie sú

len(premenna) - nám vráti dĺžku stringu a ako integer.

premenna[0] - nám vráti znak stringu na 0-tej pozícii, prípadne inej, podľa toho akú zadáme, treba si dať ale pozor, pretože ak je string kratší ako pozícia ktorú zadáme, dostaneme error.

Vyskúšajte

Vyskúšajte si vypísať reťazec po znakoch pomocou for cyklu a len().

```
retazec = "Ahoj, moj den je super, ucim sa totiz python"
for i in range(len(retazec)):
      print(retazec[i])
```

PRÍKLADY

6.3 Príklad

Vytvorte funkciu **znak(slovo, pozicia)**, ktorej užívateľ pošle slovo a pozicia, z ktorej chce znak vypísať. Funkcia následne znak na tejto pozícii vráti, alebo vráti chybovú hlášku podľa vašej fantázie, ak táto pozícia v stringu neexistuje.

6.4 Príklad

Vytvorte funkciu **obsahuje_znak(slovo, znak)**, ktorá prejde zadané slovo for cyklom, a ak narazí na zadaný znak, vráti true, inak po skončení for cyklu vráti false.

6.5 Úloha na precvičenie

Napíšte funkciu **badge(name)**, ktorá zistí dĺžku mena a podľa nej vypíše zadané meno s rámčekom.
Príklad **badge("Ema")**:

```
*******
*     *
* Ema *
*     *
*******
```

PODREŤAZCE

Nové reťazce vieme vytvoriť aj pomocou rezov už existujúcich stringov.

Podreťazec stringu získame pomocou syntaxu slovo[odkiaľ:pokiaľ], pričom prvý znak sa nachádza na pozícii 0 a znak na pozícii pokiaľ sa vo výsledku nezobrazí (všimnite si že veľmi podobne funguje aj funkcia **range()**).

Teda napríklad **slovo[0:1]** vráti prvý znak premennej slovo.

Vyskúšajte

Rôzne rezy na rôznych slovách

a všimnite si, že aj ak je druhé číslo väčšie ako dĺžka reťazcu, funkcia nám vráti jednoducho celý reťazec

```
retazec = "Ahoj, ako sa mas?"
print(retazec[0:1])
print(retazec[2:5])
print(retazec[0:500])
```

Čo sa stane ak nezadáme oba parametre, alebo ani jeden?

```
retazec = "Ahoj, ako sa mas?"
print(retazec[2:])
print(retazec[:5])
print(retazec[:])
```

Ak do rezu zadáme záporné indexy, reťazec sa bude čítať od konca (-1 znamená posledný znak, -2 predposledný a tak ďalej..)

Vyskúšajte

Rezy so zápornými indexami

```
retazec = "Ahoj, ako sa mas?"
print(retazec[-1:])
print(retazec[-5:-1])
print(retazec[:-5])
```

PRÍKLADY

7.1 Príklad

Vytvorte program, ktorému užívateľ zadá reťazec a program vypíše jeho prvé štyri znaky.

7.1.1 Príklad

Vylepšite predošlý program:

Pomocou funkcie **len()** z predošlej hodiny sa najprv spýtajte či je zadané slovo dostatočne dlhé, a ak je kratšie ako 4 znaky, vypíšte ľubovoľnú chybovú hlášku

7.1.2 Príklad

Vylepšite predošlý program ešte raz:

Nechajte užívateľa zadať aj počet znakov ktoré sa majú vypísať, takže ak užívateľ zadá napríklad: **„Nejaké slovo“**

„4“

Program vypíše: **„Neja“**

(Aby ste mohli použiť zadané číslo v reze, musíte ho prekonvertovať na integer funkciou **int()**)

7.2 Príklad

Vytvorte program, ktorému užívateľ zadá reťazec a program ho vypíše postupne, najprv prvý znak, potom prvé dva, a tak ďalej až kým nevypíše celý reťazec

Napríklad užívateľ zadá: **„Ahoj“**

Program vypíše: **„A“**

„Ah“

„Aho“

„Ahoj“

7.3 Príklad

Napíšte funkciu **deleno(retazec, cislo)**, ktorá „podelí reťazec“ daným číslom

Príklad **deleno (“auto”, 2)**:

“au”

Príklad **deleno (“trojkolka”, 3)**:

tro”

(Pozor treba použiť celočíslené delenie **cislo // cislo**, ak by ste použili klasické / výsledkom by mohlo byť necelé číslo a program by nefungoval)

7.4 **Úloha na precvičenie**

Napíšte funkciu **search(word, sentence)**, ktorá funguje nasledovne:

for cyklom prechádza premennú sentence robiac **rezy od i do i + len(word)**.

Ak sa tento rez rovná premennej word, vráti True, inak po skončení for cyklu False.

Príklad **search(“auto”, “auto”)**:

vráti True

Príklad **search (“auto”, “atuo”)**:

vráti False

OPAKOVANIE

8.1 Príklad

Vypýtajte si od užívateľa dve slová pomocout **input()**
a následne ich vypíšte vedľa seba do riadku.

8.2 Príklad

Vypýtajte si od užívateľa dve čísla pomocout **input()**
a následne vypíšte pod seba ich: súčet, rozdiel, súčin.

8.2.1 Príklad

Vylepšite príklad 8.2 tak, že pre každú operáciu si vytvoríte samostatnú funkciu.

8.3 Príklad

Vypýtajte si od užívateľa číslo a následne toľko krát (pomocou for cyklu)
vypíšte ľubovoľné slovo.

8.4 Bonusový príklad

Spojte úlohy 8.3 a 8.2.1, teda:
Užívateľ zadá číslo, spustí sa for cyklus toľko krát ako zadané číslo,
v každom cykle si program vypýta dve čísla a vykoná s nimi matematické
operácie rovnako ako v úlohe 8.2.1

8.5 Príklad

Užívateľ zadá slovo. Ak je toto slovo **„leto“** program vypíše: **„oh, na to sa teším“**, inak vypíše **„jediné na čo sa teším je leto..“**

8.6 Príklad

Užívateľ zadá slovo. Pomocou for cyklu potom skontrolujte či sú všetky písmená v slove rovnaké a vypíšte či áno alebo nie.

8.7 Príklad

Užívateľ zadá slovo. Pomocou for cyklu potom skontrolujte či je prvé a posledné písmeno rovnaké a vypíšte či áno alebo nie.

8.8 Príklad

Vytvorte funkciu **sucetAleboSucin(a, b)**, ktorá dostane dve čísla, tie vynásobí a ak je súčin menší ako 1000, vráti ho, inak vráti namiesto neho ich súčet

8.9 Príklad

Vytvorte funkciu **parnePismena(slovo)**, ktorá dostane jedno slovo, a vypíše iba

písmená na párnych pozíciách

Príklad: **parnePismena(„Ahoj“)**

Výpis: **„Ao“** (pozície začínajú od nula)

8.10 Bonusový príklad

Užívateľ zadá číslo, následne, pomocou dvoch for cyklov, vypíšte každé číslo od 1 po zadané číslo do riadku toľko krát, aká je jeho hodnota

Príklad:

n = 2

1

2 2

n = 4

1

2 2

3 3 3

4 4 4 4

OPAKOVANIE - FUNKCIE NAD STRINGAMI

9.1 Príklad

Napíšte program, ktorý si od užívateľa vypýta string, a následne vypíše jeho dĺžku a prvé a posledné písmeno (ak je dĺžka stringu aspoň 1).

9.2 Príklad

Napíšte program, ktorý si od užívateľa vypýta string, a ak má string aspoň 1 znak, vypíše ho bez prvého znaku. **Teda, pre vstup „Slovo“, vypíše „lovo“.**

9.3 Príklad

Napíšte funkciu **vymaz(slovo, n)**, ktorej pošleme string a číslo a tá vráti/vypíše podreťazec od daného znaku.

9.4 Príklad

Napíšte funkciu **obsahuje(slovo, znak)**, ktorej pošlete string a string dĺžky jedna (písmeno) a ona vráti true alebo false podľa toho či sa znak nachádza v premennej slovo.

9.4.1 Príklad

Vylepšite funkciu **obsahuje(..)** tak aby nevracala true alebo false, ale naopak vrátila index, na ktorom sa písmeno nachádza, alebo -1 ak sa v stringu nenachádza

PRÍPRAVA NA HRU - OBESENEC

9.5 Príklad

Napíšte funkciu **vymen(slovo, znak, n)**, ktorej pošleme string, znak a číslo a tá vymení znak na n-tej pozícii zadaným znakom. Keďže sa toto nedá spraviť priamo pomocou slovo[n] = znak, musíme vytvoriť substring po n-tý znak pripojiť k nemu znak a k nim pripojiť substring od znaku na pozícii (n+1) po koniec stringu.

9.6 Príklad

Napíšte funkciu **skopiruj(slovo)**, ktorá dostane string a vráti string rovnakej dĺžky, ale všetky písmená budú **„_“**.

Napríklad, pre vstup: **„ahoj“**

funkcia vráti: **„____“**

9.7 Bonusový príklad - hra obesenec

Najprv necháme užívateľa aby zadal slovo, ktoré sa bude snažiť „uhádnuť“

Následne si pomocou funkcie **skopiruj(slovo)**, vytvoríme kópiu s podtržítkami

a uložíme ju do premennej.

Ďalej napíšeme funkciu **hadaj(slovo, kopia, znak)**, ktorá dostane string, jeho kópiu a znak. Vytvoríme si premennú **najdene = False** a string slovo prechádzame for cyklom.

Ak narazíme na písmeno ktoré je rovnaké ako znak, zavoláme funkciu **kopia = vymen(..)** pre kopia, znak, aktuálny index a najdene nastavíme na True.

Nakoniec, ak je **najdene == True** vypíšeme “Uhádol si!”,

inak “Toto písmeno v slove nie je“ a vrátime novú kópiu slova.

9.7.1 Bonusový príklad - hra obesenec

Aby sa hra dala aj hrať, musíme ešte dokončiť kolá a vstup písmen od užívateľa.

Pre jednoduchú verziu, môžeme dať užívateľovi toľko pokusov, ako je v zadanom slove písmen.

To znamená že po zadaní slova na uhádnutie a vytvorení jeho kópie zistíme jeho dĺžku a použijeme for cyklus v ktorom bude užívateľ hádať. V každom cykle teda užívateľ zadá písmeno (ak zadá reťazec dlhší ako jedno písmeno, stratí jeden pokus) a následne zavoláme funkciu **hadaj(slovo, kopia, znak)** a uložíme jej výsledok znova do premennej kopia.

Ak sa kópia po skončení for cyklu rovná prvotne zadanému slovu, užívateľ vyhral, inak prehral.

WHILE CYKLUS

For cyklus sa hodí v prípade že dopredu vieme koľkokrát chceme kus kódu opakovať. No existujú aj prípady kedy túto informáciu nemáme a chceme cyklus opakovať kým (po anglicky while) platí nejaká podmienka. V takýchto prípadoch je ideálne použiť while cyklus.

While cyklus použijeme nasledovne - **while(podmienka):** kde podmienka je nejaký kus kódu ktorý nám vráti True alebo False.
Cyklus sa najprv pozrie na podmienku (rovnako ako if), ak je True, vykoná kód vo svojom bloku a potom sa vráti znova na začiatok, znova pozrie na svoju podmienku a tak ďalej až kým nezistí že podmienka je False, a až vtedy pokračuje ďalej.

Vyskúšajte

Vyskúšajte si nasledovný kus kódu a pohrajte sa s rôznymi podmienkami, nech vidíte ako asi while cyklus funguje.

```
x = 10
while(x > 5):
      print("x je stale viac ako 5")
      x -= 1
```

Dávajte si však pozor, pri while cykle hrozí že naša podmienka nikdy nebude False a teda by nastalo takzvané zacyklenie a kód by nikdy neskončil.

```
x = 10
while(x < 100):
      print("x je stale viac ako 5")
      x -= 1
```

Ak napríklad spustíme nasledujúci kód, program nám buď zamrzne, alebo bude donekonečna vypisovať to isté až kým ho „nasilu“ nezastavíme. (Pretože ak od 10 postupne odrátavame vždy jedna, číslo bude vždy menšie ako sto)

PRÍKLADY

10.1 Príklad

Napíšte program, ktorý pomocou while cyklu vypíše 10 krát ľubovoľné slovo.

Skúste vymyslieť aspoň dve rôzne podmienky, ako by ste to vedeli dosiahnuť.

10.2 Príklad

Napíšte program, ktorý si od užívateľa pýta dokola slovo, až kým zadané slovo nie je „stop“.

Hint: vypýtajte si slovo od užívateľa a uložte ho do premennej ešte pred začatím while cyklu, aby ste túto premennú vedeli použiť v podmienke cyklu

10.2.1 Príklad

Vylepšite predošlý príklad tak že si program bude od užívateľa pýtať dve slová,

až kým slová nebudú rovnaké.

10.3 Príklad

Napíšte program, ktorý si od užívateľa vypýta číslo, ktoré následne vo while cykle delíte 10, kým je číslo väčšie ako 0.

Čo týmto spôsobom počítame?

10.4 Príklad

Napíšte program, ktorý si od užívateľa pýta dve čísla, až kým prvé číslo nie je väčšie ako druhé.

10.4.1 Bonusový príklad

Vylepšite príklad 10.4 tak, že keď získame dve čísla kde prvé je väčšie, vypíšte pomocou while cyklu „Prvé číslo je stále väčšie!“ toľko krát o koľko je prvé číslo väčšie.