3.1. Програмски језик Пајтон (Python)

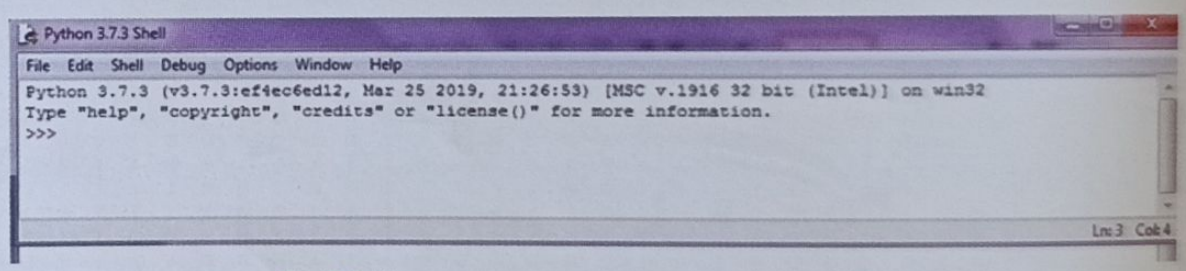
Кључне речи ▶ подаци, објекти, променљиве.

Подсети се:

- типова података;
- аритметичких операција;
- шта су то променљиве.

Типови података, аритметичке операције, променљиве

Пајтон (Python) јесте текстуални програмски језик. Садржи обимне и квалитетне програмске библиотеке. Радно окружење у Пајтону састоји се од линије менија и радне површине у којој се пише код програма (сл. 3.1.).



Сл. 3.1. Радна површина програмског језика Пајтон

САЗНАЈ ВИШЕ

Програмски језик Пајтон добио је име по серији „Летећи циркус Монтија Пајтона" (енг. *Monty Python's Flying Circus*), комедији BBC-а која је била актуелна седамдесетих година прошлог века. Програмски језик Пајтон настао је почетком 90-их година прошлог века у Холандији. Аутор је Гвидо ван Росум. Ауторска права за овај програмски језик држи непрофитна организација Python Software Foundation (PSF). Свако може да користи програмски језик Python за личне и за комерцијалне потребе.

Рачунарски програм чини низ наредби које се извршавају над неком врстом **података**, као што су карактер, број, ниска или стринг (низ карактера). Вредности података у Пајтону представљене су **објектима**. То могу бити слова (велика и мала), бројеви, знаци интерпункције, размаци и специјални симболи.

Приликом учења програмирања у Пајтону у 6. разреду, користио/користила си изразе за основне рачунарске операције. Погледај табелу у наставку и подсети се тих рачунарских операција.

Рачунарска операција	Математички запис	Запис у програмском језику Пајтон	Резултат
Сабирање	6 + 2	`6+2`	8
Одузимање	6 – 2	`6-2`	4
Множење	6 · 2	`6*2`	12
Дељење	6 : 2	`6/2`	3.0

У последњој ћелији претходне табеле резултат је записан у облику реалног броја. О овој особини учио/учила си када је било речи о реалном и целобројном дељењу које постоји код већине програмских језика. Наиме, код реалног дељења количник је увек реалан број, док при целобројном дељењу имаш две могућности: резултат – само целобројни део количника или остатак при дељењу.

Да се подсетимо.

Рачунарска операција	Запис у програмском језику Пајтон	Резултат	Објашњење
Реално дељење	`26/5`	5.2	количник је увек реалан број
Целобројно дељење	`26//5`	5	резултат целобројног дељења даје целобројну вредност количника, што је у овом случају 5
Дељење са остатком	`26%5`	1	остатак при дељењу броја 26 бројем 5 је 1, што је и резултат оваквог дељења

Улазни подаци могу се дефинисати **променљивама**. Променљиве су подаци који могу мењати своје вредности у току извршавања програма, не морају се декларисати, али имају правила којих се мораш придржавати. Име променљиве не може:

- почети бројем;
- садржати размаке, знаке интерпункције, цртице попут знака минус;
- садржати кључне речи као што су `true`, `false` итд.

Уколико постоји више операција које се комбинују, користе се заграде како би се одредио редослед израчунавања рачунских операција (заграде се примењују као и у математици).

Не може се писати `promenljiva-1`, али може `promenljiva_1`.

ЗАДАТАК

1. Користећи раније стечено знање, напиши програм којим ћеш:
   а) израчунати количник збира бројева 15 и 90 и разлике бројева 14 и 9;
   б) израчунати збир производа бројева 2 и 8 и количника бројева 64 и 8.
2. Напиши програм којим ћеш одредити вредности израза у Пајтону:
   а) `125/5;` б) `75//10;` в) `15%3.`

Функције

Кључне речи ▶ уграђене функције, тело функције, дефинисање функције.

Функције помажу да програм буде краћи и боље организован. Користе се, обично, када у програму треба више пута написати један исти кôд, па је много боље написати га једном у функцији и позивати по потреби. Функције имају широку намену.

Најчешће коришћене функције

Име функције	Значење
`print`	функција за исписивање вредности (бројева или текста који се налази између знакова навода)
`input`	функција помоћу које се уносе вредности у програм
`int`	функција која је скраћеница од `integer`, што представља целобројан тип података и уз помоћ функције `input` омогућава унос целобројне вредности

Користио/користила си такозване **уграђене функције**, које омогућавају много лакше и брже писање програма.

Најчешће коришћене уграђене функције

Име функције	Запис и резултат	Значење
`abs`	`abs(-3)=3`	одређивање апсолутне вредности броја
`min`	`min(9,2)=2`	одређивање најмањег броја од задате листе бројева
`max`	`max(14,75,4)=75`	одређивање највећег броја од задате листе бројева
`round`	`round(3.4)=3` `round(7.8)=8`	заокруживање реалног броја на најближи цео број

Често је неопходно да сам/сама дефинишеш функције. Функције се дефинишу наредбом `def`, именом и улазним параметрима, а завршавају се двотачком (:). Подсетићемо се примера површине правоугаоника у којем су познате дужине страница.

```
def povrsina(a, b):
    pov = a*b
    return pov
print(povrsina(30, 20))
```

Назив функције је `povrsina`, док су `a` и `b` улазни подаци. Увучени блок зове се **тело функције**. Наредба `return` враћа вредност као излазни податак, а уједно је и излаз из функције.

ЗАДАТАК

3. Напиши програм којим ћеш израчунати све унутрашње углове једнакокраког трапеза, ако је позната мера једног од њих, угао $\alpha = 75°$.
4. Напиши програм у Пајтону за претварање унете вредности у метрима у километре и метре.
5. Марко три дана заредом чита књигу. Напиши програм којим се рачуна колико је страна у просеку Марко прочитао сваког дана. Бројеви прочитаних страна су произвољни, цели бројеви, који се уносе са тастатуре.

Код једнакокраког трапеза углови алфа и бета су међусобно једнаке величине, а међусобно су једнаки и углови гама и делта.

Ниске (стрингови), структуре података и листе

Кључне речи: низ, текстуалне променљиве, структура података, листа.

Ниска или **стринг** (енг. *string*) представља низ карактера, а обележава се знацима навода или апострофа. Уколико је неопходно да се знаци навода испишу на екрану, користе се косе црте, које се не исписују приликом извршавања програма и штампања ниски.

Иста реченица може се написати комбинацијом знака навода и апострофа:

```
print('Uzviknuo je:"Čuvaj se psa!"').
```

Реченица:	Узвикнуо је: „Чувај се пса!”
Иста реченица у Пајтону:	`print("Uzviknuo je:\"Čuvaj se psa!\"")`

Изглед на екрану:

```
Uzviknuo je:"Čuvaj se psa!"
```

Текстуалне променљиве добијају се надовезивањем променљивих једне на другу. Над неким карактерима могу се вршити операције сабирања и множења.

`Len` је функција за пребројавање карактера. Приликом претраге редног броја карактера, односно позиције ниске, почиње се са нулом.

Позиција у ниски	0	1	2	3	4	5
Пример карактера	П	а	ј	т	о	н

Уколико желиш да издвојиш карактер „т", можеш га издвојити изразом `naziv[3]`. Део ниске „ајт" можеш издвојити изразом `naziv[1:4]`, где се приликом издвајања наводи распон позиција, где се даље издвајају карактери рачунајући прву позицију – до наведене друге.

Наредба за претрагу ниски је `find`, док се за пребројавање користи наредба `count`.

Ниске могу бити и цифре. Применом оператора **+** ниске се надовезују, а не сабирају, па за `"123"+"45"` добијамо `"12345"`.

Уколико се ради о децималним бројевима, користи се тип података `float`.

Структура података представља начин на који се подаци смештају, приказују, користе и обрађују у рачунарској меморији. **Листа** је структура података у којима се више вредности чува под једним именом у тачно одређеном редоследу. Сваки члан листе има свој индекс. Вредности листе су **елементи** и они су смештени унутар угластих заграда и одвајају се зарезом.

Најчешће коришћене функције за рад са листама

Наредбе листе	Значење наредбе листе

`max(a)`	приказује највећи елемент листе:
	`duzine=[3,5,7,6,4]`
	`max(duzine)=7`

`min(a)`	приказује најмањи елемент листе:
	`sirine=[3,5,7,6,4,2]`
	`min(sirine)=2`

`sum(a)`	сабира све елементе листе:
	`sabirci=[10,2,3,40]`
	`sum(sabirci)=55`

`a.index()`	приказује место траженог елемента у листи:
	`visine=[155,170,165,174,162]`
	`visine.index(3)=174`
	Дакле, трећи елемент листе је 174 (први елемент има вредност 0).

`sortied(a)`	сортира елементе у листи:
	`ucenici = ["Mirić Milenko", "Stojković Milica", "Adamović Aleksandar", "Jokić Đorđe", "Filipović Katarina", "Zlatković Jasna"]`
	сортира ученике по абецеди:
	`(sorted(ucenici))='Adamović Aleksandar', 'Filipović Katarina', 'Jokić Đorđe', 'Mirić Milenko', 'Stojković Milica', 'Zlatković Jasna'`

`dela`	брише елемент листе:
	`ocene=[2,2,5,4,5]`
	`del ocene[3]`
	брише елемент на позицији 3, па листа изгледа:
	`ocene=[2,2,5,5]`(избрисана је 4)

`a.append()`	додаје елемент на крају листе:
	`brojevi=[5,67,17,89]`
	`brojevi.append(33)`
	листа након додавања елемента изгледа:
	`brojevi=[5,67,17,89,33]`

Програмска структура

гранање програма (`if`, `elif`, `else`), петље (`for` и `while`).

Кључне речи

Програми могу имати различиту структуру. Постоје линијска структура програма, код које се кораци у програму изрвшавају један за другим од почетка до краја, и гранање у програму или петље.

Гранање програма

Често треба да донесеш одлуку о нечему, што је услов даљег тока догађаја. Тада одлучујеш да ли ћеш нешто урадити или не, а некад бираш и коју од две ствари треба да урадиш. Тако је и у програмирању. У програмирању је услов израз чија се истинитост проверава. Услови се могу испитати уз помоћ оператора поређења, који могу бити релацијски и логички.

Наредба гласи овако:

```
If uslov:
    naredba
```

Оператори	
Релацијски	**Логички**
веће (>)	И (`and`)
мање (<)	
једнако (=)	ИЛИ (`or`)
веће или једнако (>=)	
мање или једнако (<=)	НЕ (`not`)
различито (!=)	

Код једноставнијих примера проверава се истинитост израза и ако је услов тачан, извршиће се наредба, а затим и остатак програма. У ту сврху се користи наредба `if`.

`If-else` наредба значи да постоји још једно решење и, у зависности од испуњености услова, омогућава да се изврши једна или друга наредба. На пример, када желиш да прећеш улицу на пешачком прелазу на којем постоји семафор, ако је зелено светло на семафору, прећи ћеш улицу, а ако није, нећеш прећи улицу.

```
if uslov:
    naredba_1
    ...
    naredba_m
else:
    naredba_1
    ...
    naredba_n
```

Понекад је потребно комбиновати више услова. Додавање додатног услова омогућава конструкција `if-elif-else`. На пример, желимо да се на екрану телефона или рачунара појави поздравна порука која је другачија за одређене делове дана. Написаћемо програм у коме ће се, на основу унете целобројне вредности сати, исписивати поздрав који се користи у том делу дана и то ћемо показати на примеру у наставку.

ПРИМЕР

На основу унете целобројне вредности за сате (часове), програм ће исписивати поздрав који се користи у том делу дана, на следећи начин:

1–8 часова, порука је „Добро јутро".

9–18 часова, порука је „Добар дан".

19–24 часа, порука је „Добро вече".

```
sati = int(input("Unesi koliko je sati:"))
    if sati <= 8:
        print("Dobro jutro")
    elif sati <= 18:
        print("Dobar dan")
    else:
        print("Dobro veče")
```

Петља

Када постоји потреба да се неки део програма понавља више пута, користе се петље (енг. *loop*). Користе се две врсте петље: `for` и `while`.

`For` петља контролише број понављања. Записивање петље почиње кључним појмом `for`, затим следи име променљиве (`promenljiva_vrednost`), па реч `in` и позив функције `range` која омогућује колико ће се пута поновити наредба у петљи. На крају првог реда петље стоји двотачка, а у другом делу петље пише се наредба која треба да се понавља.

Подсетићемо се уз помоћ примера у којем се одређују вредности бројева од 5 до 10 и њихових квадрата.

```
print("Kvadrati brojeva od 5 do 10")
for broj in range(5,11):
        kvadrat=broj*broj
        print(broj,"na kvadrat je",kvadrat)
```

Решење:

```
Kvadrati brojeva od 5 do 10
5 na kvadrat je 25
6 na kvadrat je 36
7 na kvadrat je 49
8 na kvadrat je 64
9 na kvadrat je 81
10 na kvadrat je 100
```

`While` петља користи се када није унапред познато колико пута треба нека наредба да се понови. Ова наредба је контролисана условом. Све док је услов испуњен, блок наредби унутар `while` петље ће се понављати.

Претходни пример може се урадити и уз помоћ `while` петље, на следећи начин.

```
print("Kvadrati brojeva od 5 do 10")
i=5
while i <= 10:
    print("Broj=", i, "kvadrat broja=", i*i)
    i=i+1
```

Решење:

```
Kvadrati brojeva od 5 do 10
Broj = 5 kvadrat broja = 25
Broj = 6 kvadrat broja = 36
Broj = 7 kvadrat broja = 49
Broj = 8 kvadrat broja = 64
Broj = 9 kvadrat broja = 81
Broj = 10 kvadrat broja = 100
```

Графика у Пајтону

корњача графика, библиотека. ◀ Кључне речи

За цртање у програмском језику Пајтон, углавном се користи библиотека за рад са корњачом, такозвана **корњача графика**. У Пајтону можеш бирати између неколико ликова. Подразумевано цртање врши троугао који се помера, али без обзира на то што је троугаоног облика, зову га корњачом. Укључивање библиотеке ради се помоћу наредбе `from turtle import*`, чиме се отвара прозор величине 600 x 600 пиксела, са оловком која је у центру прозора, усмерена ка десној страни. Дужина линије задаје се пикселима, а оријентација степенима. Одређују се још и боја, ширина, облик и положај оловке (да буде подигнута или спуштена). Кодови за различите облике могу се видети у табели у наставку.

`from turtle import*`
`shape("turtle")`
`forward(10)`
	`from turtle import*`
	`shape("circle")`
	`forward(10)`
		`from turtle import*`
		`shape("arrow")`
		`forward(10)`