3.3. Функције

Кључне речи ▶ Функција, уграђене функције, минимум, максимум.

Подсети се:

- како се у програмском окружењу Пајтон извршава рачунање сложенијих израза;
- која је сврха коришћења наредбе `print()`.

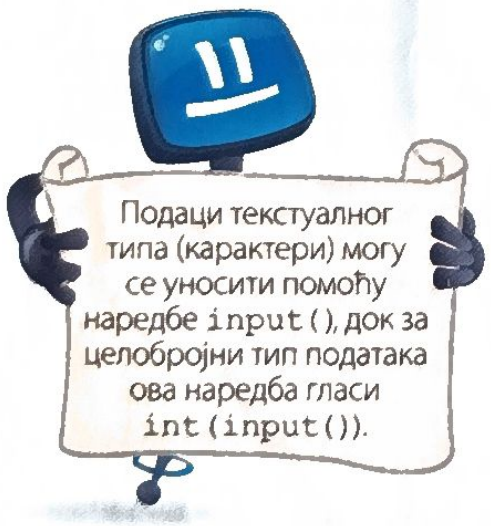
Функције у програмском окружењу Пајтон имају широку намену. Корисне су за дељење великих програма на мање делове, да би се учинили краћим, а због тога и читљивијим. Могу се извршавати више пута у зависности од задатка тј. проблема. Позивањем функција извршава се низ наредби.

У Пајтону се најпре дефинише функција која израчунава површину правоугаоника. Функције се дефинишу наредбом `def`, именом и улазним параметрима. Наредба се завршава двотачком, а код који је део функције пише се у блоку испод наредбе `def`.

```
>>> def print_povrsina ():
```

Пајтон има неке већ написане функције за команде које се често користе. То су, **уграђене функције** (енг. *Built-in*). Оне омогућавају много лакше и брже писање програма.

Функција `print` ти је већ позната, јер си је до сада користио/користила за исписивање бројева. Исту функцију можеш користити за исписивање текста, тј. низа карактера који се налазе између знакова навода.




```
>>> print ("Unesi vrednost prve stranice.")
Unesi vrednost prve stranice.
```

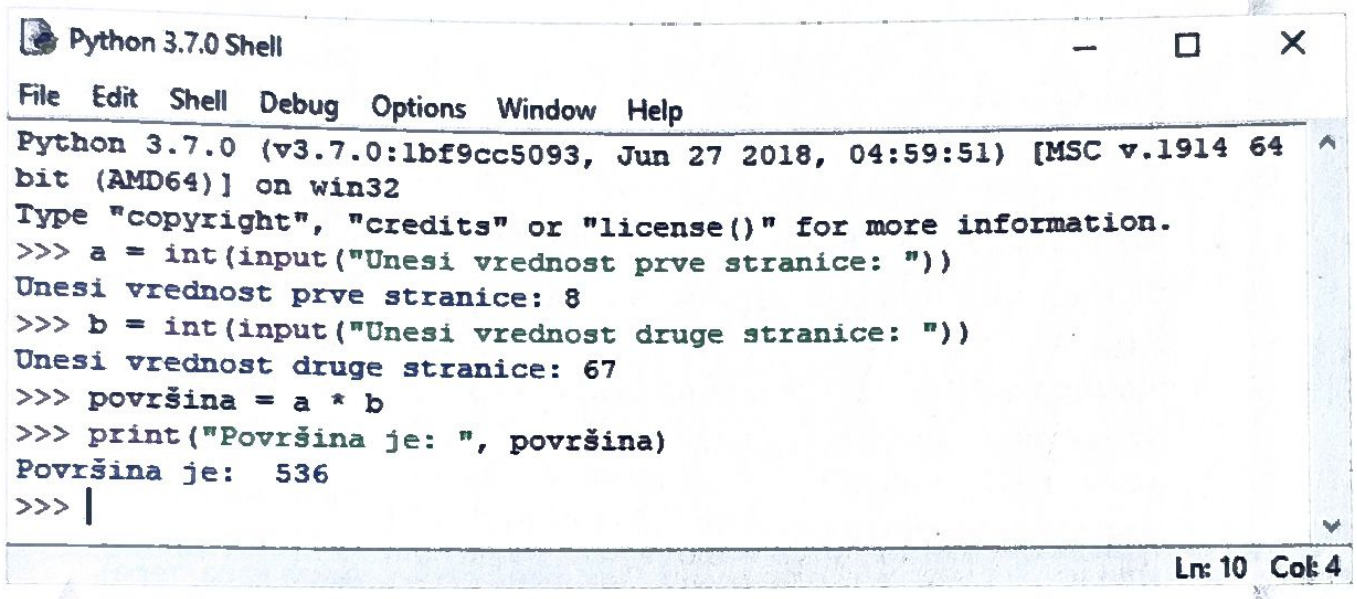
Осим ње, често се користи функција `input()`. Позивањем ове функције можеш да унесеш неку улазну вредност у програм. На пример:

```
>>> grad = input('Unesi ime grada: ')
Unesi ime grada: Beograd
>>> print('Glavni grad, ', grad)
Glavni grad, Beograd
```

Да би унели целе бројеве користимо функцију `input()`, такође морамо дефинисисти да желимо да и наш програм разуме да је то цео број коришћењем `intiger`-а, који се пише `int`.

```
>>> a = int(input("Unesi vrednost prve stranice: ")).
```

Када све ово научиш, можеш написати програм за израчунавње површине правоугаоника (сл. 3.21).



3.21. Приказ уноса улазних променљивих

ЗАДАТАК

1. Задатак се ради индивидуално/у пару.
2. У наставку су дати делови програма за израчунавање просека оцена.
3. Правилно их допуни и израчунај у Пајтону просек својих оцена.

```
print ('Ocena iz srpskog jezika i književnosti: ')
s = input()
print ('Ocena iz matematike: ')
m = input()
print ('Ocena iz informatike i računarstva: ')
i = input()
print ('Ocena iz biologije: ')
b = input()
print ('Ocena iz geografije: ')
g = input()
zbir_ocena =
prosek =
print ('Prosek ocena je: ', prosek)
```

Уграђене функције које се често користе приказане су у табели у наставку.

Уграђене функције

Функције	Резултати	Значење
`abs(-5)`	`abs(-5) = 5`	одређивање апсолутне вредности броја
`min(12,5)`	`min(12,5) = 5`	одређивање минималног броја листе из задате листе бројева
`max(14,18,23,78,4)`	`max(14,18,23,78,4) = 78`	одређивање максималног броја листе из задате листе бројева
`round(3.4)`	`round(3.4) = 3`	заокруживање реалног броја на најближи цео број
`len("Aleksandra")`	`len("Aleksandra") = 10`	одређивање дужине стринга (низа карактера)

Да би неки проблем лакше решио, функције можеш дефинисати сам/сама. На пример, могуће је дефинисати функције за израчунавање унутрашњих углова многоугла, израчунавање површине различитих геометријских фигура, претварање јединица за мерење времена и др.

Дефинисање функције објаснићемо на примеру рачунања површине правоугаоника коме су познате дужине страница.

Дефинисање функције почињеш појмом `def`, а затим упишеш име функције. Након тога следи пар малих заграда, унутар којих се уписују улазни параметри, и на крају се ставља двотачка.

```
>>> def povrsina(a, b):
```

Назив функције је *povrsina*, док су *a* и *b* улазни подаци, који представљају дужине страница. Све после прве линије дефинисања функције, тј. двотачке је **тело функције**. Тело је увучени блок програмског кода. Када у телу функције наиђемо на ред `return a*b`, значи да функција престаје са радом и враћа неку вредност као излазни податак.

```
>>> print(povrsina(20, 10)
```

Команда `return` означава и излаз из функције, тако да се може писати и без пратећих података. Код најједноставнијих функција тело функције представља само наредба `return`, након које се налази израз који представља везу између улазних параметара и резултата функције.

Код сложенијих функција у телу се налазе компликованији изрази, али се и даље у телу функције (најчешће на самом крају) налази наредба `return`, иза које се наводи вредност функције.

```
>>> def povrsina(a, b):
        a = 20
        b = 10
        return a * b

>>> print(povrsina(20, 10)
```

ВЕЖБА

1. Вежба се ради индивидуално/у пару.
2. Покрени радно окружење Пајтона. Напиши програм за израчунавање максималне дужине скока удаљ за 3 такмичара, ако знаш да су дужине скокова такмичара биле 130 cm, 125 cm и 131 cm.
3. Програм можеш писати као што је започето у наставку.

```
>>> prvi_takmicar = int(input("Unesi dužinu skoka za prvog
takmičara: "))
>>> drugi_takmicar = int(input("Unesi dužinu skoka za drugog
takmičara: "))
>>> treci_takmicar = int(input("Unesi dužinu skoka za trećeg
takmicara: "))
>>> print("Najduži skok je: ", max(prvi_takmicar, drugi_
takmicar, treci_takmicar))
```

Напомена: Уочи да се под знацима навода пишу карактери који се појављују на екрану.

ЗАДАТАК

1. Задатак се ради индивидуално/у пару.
2. Израчунај унутрашње углове паралелограма, ако је позната мера једног од њих.

   $\alpha = 60$

Понекад функција треба да врати више вредности. На пример, желимо да претворимо метре у километре и метре. Резултат тада можемо вратити у облику два елемента или више њих.

```
>>> def kilometri_u_metre(m):
        return (m // 1000, m % 1000)

>>> (km, m) = kilometri_u_metre (2553)

>>> print(2553, "m = ", km, "km i", m, "m")

2553 m = 2 km i 553 m
```

ЗАДАТАК

1. Задатак се ради индивидуално/у пару.
2. Користећи претходни пример, напиши програм који за унети одређени број секунди израчунава колико је то сати минута и секунди.

Запамти!

Функције у програмском окружењу Пајтон користе се за дељење великих програма на мање делове, за креирање помоћних програма, израчунавање максимума, минимума или просека, понављање истог кода у различитим деловима програма и слично.

Пајтон има неке већ написане функције за команде које се често користе и зову се **уграђене функције**. То су: `print`, `input`, `int`, `abs`, `max`, `min` и др.

Функције може да креира корисник прогама, употребом појма `def`. Функција престаје са радом и враћа неку вредност помоћу команде `return`.

Провери знање

1. Шта у програмском језику Пајтон представљају функције?
2. Наведи неколико уграђених функција.
3. За које радње у програму се користе функције print и input?
4. Објасни поступак дефинисања функције.

Истражи

Пронађи у свом окружењу особу која је математичар и кроз разговор сазнај које су могућности Пајтона за израчунавање углова, дужине страница, површина и запремина различитих геометријских фигура и тела.

Корисни линкови

http://edusoft.matf.bg.ac.rs/ePython/

Постоји много функција које олакшавају рад са листама. Неке од најчешће коришћених су приказане у табели у наставку.

Наредбе листе	Значења 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.
`a.sort()`	Сортира елементе у листу.

`del a`	Брише елемент листе:
	`ocene = [2,2,5,4,5]`
	`del ocene[3]`
	Брише елемент на позицији 3, па листа изгледа овако:
	`ocene = [2,2,5,5]`

`a.append()`	Додаје елемент на крај листе:
	`brojevi = [5,67,17,89]`
	`brojevi.append(33)`
	Листа након додавања елемента изгледа овако:
	`brojevi = [5,67,17,89,33]`

Гранање програма

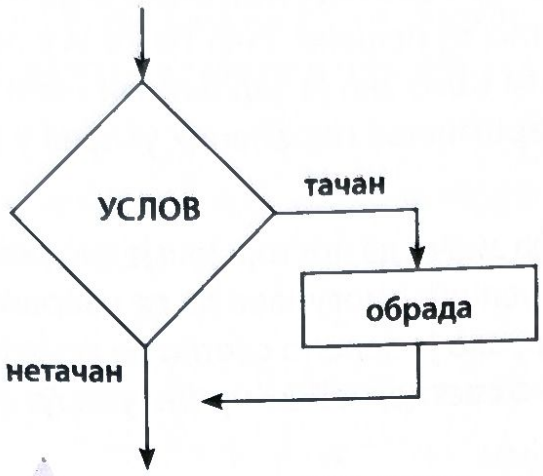
Често се сусрећеш са потребом доношења одлуке о нечему, што ће бити услов даљег тока догађаја. На пример код питања: Да ли си урадио/урадила домаћи задатак? Услов код овог питања је да одговор буде тачан (енг. *true*) да би могао/могла да се играш.

```
>>> uradio_sam_domaci = True
>>> if uradio_sam_domaci:
        print("Можеш da se igraš")
```

Важно!

Након услова обавезно се мора навести двотачка (:). Наредбе које се условно извршавају морају се мало увући (можеш испред сваке наредбе откуцати три размака).

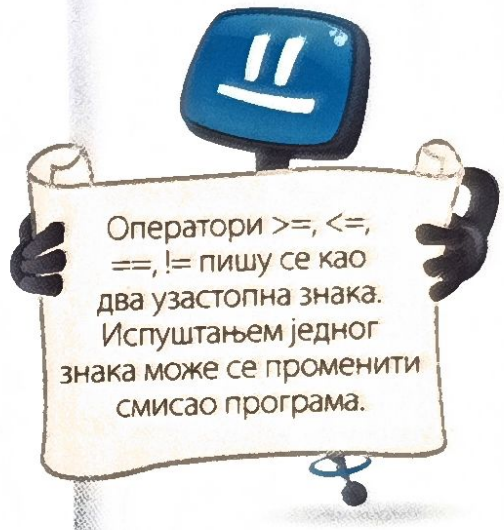
У програмирању услов је израз чија се истинитост проверава. Ако је услов тачан, извршиће се наведена наредба, а затим и остатак програма. Ако услов није тачан, одмах се прелази на остатак програма. Алгоритам услова приказан је на слици 3.23.



3.23. Алгоритам услова

Размисли

Дефиниши шта су логички оператори.



Услов се може испитивати уз помоћ оператора поређења, који могу бити релацијски и логички.

Релацијски оператори упоређују два операнта или израза. Резултат добија вредности True или False, што значи да може бити истинит или лажан. Сви релацијски оператори приказани су у схеми у наставку.

Логички оператори омогућавају да се усмерава ток извршавања програма. За стварање било ког сложеног суда, довољна су три логичка оператора. Логички оператор `and` (срп. и) захтева да оба израза буду тачна да би цео сложени израз био тачан. На пример: `(5 < 7) and (7 < 9)` је тачно `(True)`.

Код логичког оператора `or` довољно је да је само један од два релацијска операнта или израза тачан да би цео сложени израз био тачан. На пример: `(5 < 7) or (7 > 9)` је тачно `(True)` јер је задовољан услов да бар један израз, у овом случају `(5 < 7)`, буде тачан.

Логички оператор `not` мења стање истинитости у супротно. На пример `not(5 > 7)` је тачно `(True)`.

If-else наредба

Пожељно је при испитивању неког услова, одабрати једну од две могућности. На пример, када желиш да пређеш улицу на пешачком прелазу на којем постоји семафор, потребно је да буде задовољен услов да је зелено светло за пешаке. Тако ће се и у програмима поједине наредбе извршити само ако је задовољен неки услов. Наредбом `if` омогућава се извршавање одређених услова у програмима написаним у Пајтону.

`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, ако је зелено светло на семафору, прећи ћеш улицу, ако није зелено светло, нећеш прећи улицу.

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

Понекад је потребно комбиновати више услова и потпуно их проверавати. То нам омогућује конструкција `elif`. На пример, желимо да испитамо да ли је неки број позитиван, нула или негативан. Написаћемо програм у коме је први услов да се провери да ли је број већи од нуле. Ако јесте, исписаће се да је број позитиван. Следећи услов ће бити да се провери да ли је број једнак нули и ако јесте исписаће се да је број 0. Последњи услов је да се провери да ли је број мањи од нуле и ако јесте, исписаће се да је број негативан.

```
>>> broj = int(input("Unesi broj: "))
Unesi broj: 54
>>> if broj >0:
     print("Broj je pozitivan")
>>> elif broj==0:
     print("Broj je nula")
>>> else:
     print("Broj je negativan")

Broj je pozitivan
```

ВЕЖБА

1. Вежба се ради индивидуално/у пару.
2. Напиши програм који ће у зависности од тога колико је сати исписати једну од порука приказаних у табели. Води рачуна да се поруке приказују онако како су у табели написане.

Сати	Поруке
0–8	Добро јутро
9–18	Добар дан
19–23	Добро вече

ЗАДАТАК

1. Поделите се у три групе. Прочитајте текст задатка и урадите га.
2. На једном од наредних часова размените знања и искуства везана за резултате рада.

ЗАДАТАК

Једна група ученика треба да истражи колика је цена млечних напитака. Друга група треба да провери цену воћа и поврћа. Трећа група треба да провери цену месних прерађевина.

Напишите програм који ће показивати колико ће се променити цена ужине ако се промени цена намирница.

Петља

Понекад је потребно да се програм извршава више пута. Структура која омогућава понављање назива се **петља** (енг. *loop*). Петљом се делови програма понављају унапред задани број пута или док је одређени услов испуњен. Постоје две врсте петље: `for` и `while`.

For петља

For петља контролише број понављања.

```
>>> for promenljiva_vrednost in [vrednost1, vrednost2...]:
    blok
```

Записивање ове петље почиње кључним појмом `for`. Затим следи име променљиве (`promenljiva_vrednost`), реч `in` и позив функције `range`. На крају првог реда петље стоји двотачка, а у наставку, односно другом реду петље пише се наредба која треба да се понавља. Та наредба је мало увучена у односу на остале наредбе у коду програма.

За писање петљи користи се функција `range`, која одређује колико ће се пута поновити наредбе у петљи. Функција range прави тип објекта познат као **итерација** (енг. *iterable*).

```
>>> for num in range(5):
        print(num)
```

Овај пример показује како се уместо листе вредности позива се `range` функција са аргументом 5. Функција генерише итерабилну секвенцу целих бројева у опсегу од 0 до 5 (без укључивања броја 5). Резултат је исти као да смо откуцали `for num in [0, 1, 2, 3, 4]`.

ВЕЖБА

1. Вежба се ради индивидуално/у пару и састоји се из два задатка.

ЗАДАТАК А

Напиши програм који исписује првих десет непарних бројева.

ЗАДАТАК Б

Напиши програм који исписује бројеве из прве двадесетице дељиве са 3.

Програм за задатак Б можеш написати на следећи начин:

```
>>> print("prikaz brojeva deljivih sa 3 u prvoj dvadesetici")
>>> lista = [x for x in range(20)]
>>> lista2 = [x for x in lista if x % 3 == 0]
>>> print(lista2)
```

Функција `range` у `for` петљи на овом примеру укључује три аргумента:

- први аргумент је вредност 1, који представља почетну вредност која ће се доделити променљивој broj;
- други аргумент је вредност 10, који представља граничну вредност у листи. То значи да ће променљива broj моћи да има било коју вредност у опсегу од 1 до 10, не укључујући вредност 10;
- трећи аргумент је вредност 2, који представља корак. То значи да ће вредност 2 бити додавана свакој следећој вредности у секвенци листе.

Петља функционише тако што се после доделе почетне вредности променљивој (`broj = 1`) врши провера да ли је вредност променљиве broj у дозвољеном опсегу листе и, ако јесте, она се додељује променљивој (`broj = 3`).

После одрађивања циклуса (итерације) за `broj = 9`, врши се сабирање претходне вредности у листи (9) са кораком листе (2) и добија се следећа вредност. У овом случају добијена вредност је `11`, што је изван опсега листе. Програм напушта структуру петље и креће да обрађује следећу линију кода.

Често се променљива којој се додељују вредности у петљи користи за разна израчунавања унутар тела петље.

ЗАДАТАК

1. Задатак се ради индивидуално/у пару.
2. Употребом for петље израчунај квадрат за бројеве од 10 до 15 и прикажи резултате.
3. Да би написао/написала програм, погледај како је урађен следећи пример у Пајтону.

```
>>> print('Broj\Kvadrat')
Broj\Kvadrat
>>> print("------------------")
------------------
>>> for broj in range(1, 11):
        kvadrat = broj**2
        print(broj, '\t', kvadrat)
```

Понекад је неопходно допустити кориснику да контролише број понављања петље.

Корисник уноси горње границе опсега и на тај начин директно утиче на број итерација. Наравно, могуће је да корисник својим уносима утиче на све делове петље.

Сазнај више

После сваке итерације у петљи долази до промене вредности неке од променљивих које се користе у петљи. Ако су у питању бројчане вредности, често се рачуна сума добијена променом вредности током итерација. Овакав збир смешта се у променљиву која се зове акумулатор. Сваки акумулатор се пре стартовања петље мора поставити на 0.