Записивање конкретних вредности – сваки тип вредности има своја правила за записивање

Вредност	Тип
`2`	цео број
`-3`	цео број
`23`	цео број
`3.14`	реалан број
`-3.0`	реалан број
`.18`	реалан број једнак 0,18
`"Vuk" или 'Vuk'`	ниска карактера
`"h"`	ниска карактера
`"!"`	ниска карактера
`"Добар дан!"`	ниска карактера
`"23"`	ниска карактера
`"3.14"`	ниска карактера
`"True"`	ниска карактера
`True`	истинитосна/логичка вредност Тачно
`False`	истинитосна/логичка вредност Нетачно

Примена операција – тип вредности израза зависи од типова података и операција које користимо

Израз	Вредност
`2 + 3`	5
`2.1 + 3.6`	5.7
`2 * 3.6`	7.2
`"2" + 3.0`	грешка, није могуће сабирати ниске и реалне бројеве
`"а" + "нанас"`	"ананас"
`"аб" * 3`	"абабаб"

Додела вредности променљивама – међурезултате израчунавања можемо памтити у променљивима, променљиве се могу наводити у изразима, а могућа је и додела вредности већем броју променљивих једном наредбом

Наредба	Резултат
`a = "а" + "нанас"`	a = "ананас"
`a = 5`	a = 5
`b = a + 5`	b = 10
`b = 31 - b`	b = 21
`a = (3 < 13)`	a = True
`a, b = 2, 3`	a = 2 b = 3
`(a, b) = (2, 3)`	a = 2 b = 3

Именовање променљивих – имена променљивих смеју да садрже једно или више слова, цифре (не сме бити први знак у имену) и подвлаке (знак _) - не смеју да садрже размаке, цртице ни остале интерпункцијске знаке

Исправно именовање	Неисправно именовање
`ime_i_prezime`	`ime prezime`
`d3`	`3d`
`_d`	`f!2`
`iMe`	`f[h`
`OBIM`	

Унос/учитавање података са стандардног улаза (унос преко тастатуре) – подаци се учитавају као ниске карактера, а када желимо да учитамо број, треба да применимо још и `int` или `float` функцију

Наредба	Резултат
`>>> x = input()` `44`	x = '44'
`>>> x = int(input())` `44`	x = 44
`>>> x = float(input())` `44`	x = 44.0
`>>> x = input("Unesi ime: ")` `Unesi ime: Una`	x = 'Una'

Испис вредности на стандардни излаз

Наредба	Исписује се
`>>> a = 5`	
`>>> print(a)`	`5`
`>>> print('Zdravo, svete!')`	`Zdravo, svete!`
`>>> print('Vrednost a je jednaka', a)`	`Vrednost a je jednaka 5`

ОСНОВНЕ НАРЕДБЕ

Коментари у програму – коментари не утичу на рад програма, они су само порука за оног ко чита програм. Користимо их и када неке наредбе не желимо да се извршавају, али не желимо ни да их обришемо јер нам могу поново затребати

Запис коментара	Примери	Објашњење

`# коментар`	`# испис поруке`
	`print("Zdravo, Petre!")`
	`# print("Zdravo, Aco!")`
		Коментар описује шта ради наредна наредба. Исписује се порука Zdravo, Petre! Наредба је стављена под коментар (закоментарисана), па се не извршава.

`"""`
`Вишелинијски`
`коментар`
`"""`
	`"""`
	`Рачунање обима`
	`и површине`
	`"""`
	`obim = (sirina + visina) * 2`
	`povrsina = sirina * visina`
	`# print("Obim:", obim)`
	`# print("Povrsina:", povrsina)`
		Када коментар не може да стане у један ред, можемо користити и овакав запис коментара у више редова.
		
		Када желимо да ставимо под коментар више линија кода, тада радије користимо овакав запис где сваку линију појединачно стављамо под коментар

Контрола тока

Команда	Примери	Резултат

if *услов* :
  наредба/е
[elif:
  наредба/е]
[else:
  наредба/е]
	`if x == 1 :`
	`  print 1`
		За x = 1 се исписује 1

	`if x == 1 :`
	`  print 1`
	`else:`
	`  print 2`
		За x = 1 се исписује 1, а иначе се исписује 2

	`if x == 1 :`
	`  print 1`
	`elif x < 1 :`
	`  print 2`
	`else:`
	`  print 3`
		За x = 1 се исписује 1, за било које x мање од 1 се исписује 2, а иначе се исписује 3

for el in s:
  наредба/е
[else:
  наредба/е]
 	
s може бити интервал (range), листа, ниска и сл.
	`for i in range(5):`
	`  print("Zdravo")`
		Пет пута се исписује реч Zdravo

	`for i in range(2, 5):` `  print(i)`	Један испод другог исписују се бројеви: 2, 3 и 4. Доња граница (број 2) улази у интервал, а горња граница (број 5) не улази у интервал.
	`for i in range(5):` `  print(i)`	Један испод другог исписују се бројеви: 0, 1, 2, 3 и 4. Овде је наведена само горња граница (број 5), а подразумева се да је доња граница једнака нули.
	`for x in [2, 5, "šest"]:` `  print(x)`	Исписују се један испод другог чланови листе: 2, 5 и šest. У овом случају су два члана листе број, а један члан листе је ниска карактера.

	`for i in [2, 5, "šest"]:`
	`  print(i)`
	`else:`
	`  print "kraj"`
		Исписују се један испод другог: 2, 5, šest и kraj. Са else означавамо шта се извршава при изласку из петље.

while *услов* :
  наредба/е
 	
[else:
  наредба/е]
	x = int(input())
	while x != 0:
	  print(x, "nije nula")
	  x = int(input())
		Учитавају се бројеви док се не унесе 0. За сваки број различит од 0 се исписује да тај број није нула.

	suma = 0
	while suma <= 1000:
	  x = int(input())
	  suma = suma + x
	print(suma)
		Учитава целе бројеве и сабира их док сума не буде већа од 1000.

	suma = 0
	while suma <= 1000:
	  x = int(input())
	  suma = suma + x
	else:
	  print(suma – 1000)
		Учитава целе бројеве и сабира их док сума не буде већа од 1000. При изласку из петље исписује за колико је сума већа од 1000.

Функције - дефинисање и позив

	Примери	Резултат

Дефинисање функције
def *називФункције* ([*листа параметара*]):
  наредба/е
	def povrsina(a, b):
	  return a * b
		Дефинисање функције која рачуна површину правоугаоника чије су дужине страница a и b.

Позив функције *називФункције* ([*листа аргумената*])	P = povrsina(3, 4)	Позива дефинисану функцију и резултат (број 12) додељује промењивој P
	def f(x): return x // 10, x % 10 (a, b) = f(75) a, b = f(75) r = f(75)	Дефинисање функције која враћа две вредности: цео део и остатак при целобројном дељењу x са 10 a = 7, b = 5 a = 7, b = 5 r = (7, 5)
	def obim_trougla(a, b, c): print (a + b + c) obim_trougla(3, 4, 5)	Функција не враћа никакву вредност већ само исписује обим троугла чије су дужине страница a, b и c Позив функције исписује 12, тј. обим троугла чије су дужине страница 3, 4 и 5.

РАД СА ПРОСТИМ ТИПОВИМА ПОДАТАКА

Аритметичке операције и уграђене функције за рад са бројевним подацима

Функција	Вредност	Пример употребе
abs(x)	Апсолутна вредност променљиве x	x = abs(-17.8) # x = 17

int(x)	Цео део реалног броја, ако је x реалан број
	Бројевна вредност која одговара запису у ниски карактера, ако је x ниска
		x = int(17.8) # x = 17
		x = int(17.43) # x = 17
		x = int("12") # x = 12

float(x)	x представљен као реалан број	x = float(3) # x = 3.0
		x = float("12.3") # x = 12.3

x + y	Збир вредности променљивих x и y	x = 12 + 3 # x = 15
		x = 12 + 3.0 # x = 15.0

x - y	Разлика вредности променљивих x и y	x = 12 - 3 # x = 9
		x = 12.0 - 3.0 # x = 9.0

x * y	Производ вредности променљивих x и y	x = 12 * 3 # x = 36
		x = 12 * 3.0 # x = 36.0

x / y	Резултат реалног дељења вредности променљивих x и y	x = 1/2 # x = 0.5

x // y	Резултат целобројног дељења вредности променљивих x и y	x = 1//2 # x = 0
		x = 11//4 # x = 2

x % y	Остатак при целобројном дељењу вредности променљивих x и y	`x = 1 % 2 # x = 1` `x = 11 % 4 # x = 3`
divmod(x, y)	Торка **(x // y, x % y)**	`x = divmod(11, 4) # x = (2,3)`
x ** y	Степен основе **x** са изложиоцем **y**	`x = 3 ** 2 # x = 9`

Увоз и употреба модула `math`

Пример увоза	Пример употребе
`import math`	`print( math.sqrt(x) )`
`import math as m`	`print( m.sqrt(x) )`

Неке константе и уобичајене функције модула `math`

Функција	Вредност	Пример употребе
pi	3.1415926535897931	`x = math.pi`
ceil(x)	Функција враћа вредност најмањег целог броја који је већи или једнак реалном броју **x**.	`y = math.ceil(18.1) # y = 19.0` `y = math.ceil(18) # y = 19.0`
fabs(x)	Функција враћа апсолутну вредност броја **x** записану у облику реалног броја, без обзира на то да ли је **x** реалан или цео број.	`y = math.fabs(-4) # x = 4.0`
floor(x)	Функција враћа вредност највећег целог броја који је мањи или једнак реалном броју **x**.	`y = math.floor(18.9) # x = 18.0` `y = math.floor(18) # x = 18.0`
modf(x)	Функција враћа разломљени и цели део реалног броја.	`(r,c) = math.modf(16.1)` `# r = 0.10000000000000142 c = 16.0`
pow(x, y)	Функција враћа вредност степена у којем је основа **x** , а изложилац **y**. x^y	`x,y = 5, 2` `r = math.pow(x,y) # r = 25.0`
sqrt(x)	Функција враћа квадратни корен броја.	`y = math.sqrt(4) # y = 2.0`
trunc(x)	Функција враћа цели део реалног броја **x**, записаног у облику целог броја.	`y = math.trunc(4.91) # y = 4`

Релацијски и логички оператори

Функција	Значење	Пример употребе

<, <=	релација *је мање*, релације *је мање или једнако*	`2 <= 2 # True`
		`2 < 2 # False`
		`2 <= 5 # True`
		`2 < 5 # True`

>, >=	релација *је веће*, релације *је веће или једнако*	`2 >= 2 # True`
		`2 > 2 # False`
		`10 >= 5 # True`
		`10 > 5 # True`

==	релација *је једнако*	`x == x # True`
!= или <>	релација *је различито*	`x != x # False`
not x	логички оператор *не*	`not (3 == 5) # True`

x or y	логички оператор *или*	`(2 < 5) or (5 < 10) # True`
		`(2 > 5) or (5 < 10) # True`
		`(2 < 5) or (5 > 10) # True`
		`(2 > 5) or (5 > 10) # False`

x and y	логички оператор *и*	`(2 < 5) and (5 < 10) # True`
		`(2 > 5) and (5 < 10) # False`
		`(2 < 5) and (5 > 10) # False`
		`(2 > 5) and (5 > 10) # False`

КОЛЕКЦИЈЕ

Тип	Вредност	Примери	Примери израза чија је вредност наведеног типа

ниска	непроменљива уређена колекција карактера	`"петља"`
		`""`
		`"Zdravo, svete!"`
			`"Zdravo" + "," + " svete!"`

листа	уређена колекција елемената могуће различитог типа	`[2]`
		`[4, 5, 6]`
		`[]`
			`[2] + [4, 5, 6]`
			`x.append(3) # где је x листа`

речник	колекција кључ-вредност елемената	`{1:"Petar",`
		`2:"Milena",`
		`4:"Marko"}`
			`x.update({3:"Jovana"}) # где је x речник`

скуп	неуређена колекција елемената, могуће различитог типа, при чему се свака вредност у колекцији може јавити само једанпут	`{1, 4, 2, "12"}` `{4, 8, 3} - {8} # {4, 3}`
торка	уређена непроменљива колекција	`(2, 4, 7)` `(2, 3) + (4, 5, 6) # (2, 3, 4, 5, 6)`

ОПЕРАЦИЈЕ И УГРАЂЕНЕ ФУНКЦИЈЕ ЗА РАД СА КОЛЕКЦИЈАМА

Операције и уграђене функције за рад са свим уређеним колекцијама (листе, торке, ниске)

Операција / Функција	Вредност	Пример употребе
x in s	True ако је x елемент колекције s, у супротном је False	`7 in [2, 5, 7] # True` `"7" in [2, 5, 7] # False`
x not in s	True ако x није елемент колекције s, у супротном је False	`"7" not in [2, 5, 7] # True`
s1 + s2	Спајање колекција, где су **s1** и **s2** истог типа	`[ 3 ] + [3, 4] # [3, 3, 4]` `"ana"+"gram" # "anagram"`
s * n, n * s	Колекција настала спајањем **n** копија колекције **s**	`3 * "na" # "nanana"` `3 * [ 3 ] # [3, 3, 3]`

s[i]	Вредност елемента чији је индекс **i** у колекцији **s**
	Индекси почињу од нуле, а програмери често почетни елемент зову нултим, а не првим, слично као што у лифту иде приземље пре првог спрата
	Негативан индекс броји од краја колекције
		`# за s = [5, 6, 9]`
		`x = s [ 1 ] # x = 6`
		`# за s = "anagram"`
		`x = s [ 3 ] # x = 'g'`
		
		`x = s [ -2 ] # x = 'a'`

s[i : j]	Издвајање дела колекције **s** почевши од елемента са индексом **i** до елемента са индексом **j-1**	`# за s = "anagram"`
		`x = s [ 1 : 3 ] # x = "anagram"`
		`x = s [ : 2 ] # "ana"`
		`x = s [ 2 : ] # "agram"`
		`x = s [ -2 : ] # "am"`
		`x = s [ : -2 ] # "anagr"`

s[i : j : k]	Издвајање дела колекције **s** почевши од елемента са индексом **i** до елемента са индексом **j-1**, а у оквиру тога се узима сваки **k**-ти елемент	`# за s = "anagram"`
		`x = s [ 1 : 5 : 2 ] # x = "ng"`
		`x = s [ 1 : : 2 ] # x = "nga"`

s.index(x [,i [,j]])	**s.index(x)**
	Функција враћа индекс прве позиције на којој се појављује елемент **x** у колекцији **s**
	s.index(x, i)
	Функција враћа индекс прве позиције на којој се појављује елемент **x** у колекцији **s,** почевши позиције са индексом **i**
	s.index(x, i, j)
	Функција враћа индекс прве позиције на којој се појављује елемент **x** у колекцији **s,** почевши од позиције са индексом **i** до позиције са индексом `j-1`
	`Уколико није подниска ниске функција диже грешку.`
		`s = ['a','b','5','b','a','b']`
		`x = s.index('b') # x = 1`
		`x = s.index('b', 5) # x = 5`
		`x = s.index('b', 2, 4) # x = 2`

len(s)	Број елемената колекције **s**	`x = len([3,4,5,4,3]) # x = 5`
min(s)	Вредност најмањег елемента колекције **s**	`x = min([3,4,5,4,3]) # x = 3`
max(s)	Вредност највећег елемента колекције **s**	`x = max([3,4,5,4,3]) # x = 5`

Операције и уграђене функције за рад са уређеним променљивим колекцијама (листе)

Операција / Функција	Резултат	Пример употребе
s[i] = x	елементу са индексом **i** се додељује вредност променљиве **x**	`s = [3, 4, 5]` `s[1] = 6 # s = [3, 6, 5]`

s[i : j [: k]] = t	**s[i : j] = t**
	елементима на позицијама са индексима од **i** до индекса **j** су додељене редом вредности елемената колекције **t**
		`s = [ 3, 4, 5, 4, 3, 4 ]`
		
		`s[ 1 : 3 ] = [ 10, 12, 13 ]`
		`# [3, 10, 12, 13, 4, 3, 4]`

	s[i : j : k] = t сваком **k**-том елементу, почевши од елемента са индексом **i** до елемента са индексом **j**, додељене су редом вредности елемената колекције **t**	`s[ 1 : : 2 ] = [ 10, 12, 13 ]` `# s = [10, 12, 13]`

del s[i : j [:k]]	**del s[i : j]**
	из колекције **s** су избрисани елементи са индексом **i, i+1, …, j-1**
	del s[i : j : k]
	из колекције **s** је избачен сваки **k**-ти елемент почевши од позиције са индексом **i** до позиције са индексом **j-1**
		`s = [ 3, 4, 5, 4, 3, 4 ]`
		
		`del s[1:3] # s = [3, 4, 3, 4]`
		`del s[1:5:2] # s = [3, 5, 3, 4]`

s.append(x)	колекцији **s** додат елемент **x**	`s = [ 3, 4, 5 ]` `s.append(1) # s = [ 3, 4, 5, 1 ]`

s.extend(x)	колекцији **s** додати елементи колекције **x**	`s = [ 3, 4, 5, 4, 3, 4 ]`
		`s.extend([1,2])`
		`# s = [ 3, 4, 5, 1, 2 ]`

s.count(x)	функција враћа број појављивања елемента **x** у колекцији **s**	`s = [ 3, 4, 5, 4, 3, 4 ]` `x = s.count(4) # x = 3`

s.insert(i, x)	колекцији **s** је на позицију са индексом **i** додат елемент чија је вредност једнака **x**	`s = [ 3, 4, 5 ]`
		`s.insert(1,0) # s = [3, 0, 4, 5]`
		`s.insert(1,0) # s = [3, 4, 0, 5]`

s.remove(x)	из колекције је уклоњено прво појављивање елемента са вредношћу x	`s = [ 3, 4, 5, 3, 6 ]` `s.remove(3) # s = [4, 5, 3, 6]`
s.reverse()	елементи колекције су распоређени у обрнутом редоследу	`s = [ 3, 4, 5, 3, 6 ]` `s.reverse() # s = [6, 3, 5, 4, 3]`
s.sort()	елементи колекције су распоређени у растућем редоследу	`s = [ 3, 4, 5, 3, 6 ]` `s.sort() # s = [3, 3, 4, 5, 6]`

Операције и уграђене функције за рад са речником

Операција / Функција	Резултат	Пример употребе
len(d)	Функција враћа број парова кључ-вредност који се налазе у колекцији	`x = len( [{2:3}, {3:4}] ) #x = 2`

dict( )
dict(kwargs)**
dict(kolekcija)
	dict()
	функција враћа креиран празан речник
	dict(kwargs)**
	функција прима било које именоване аргументе, враћа речник у коме је име аргумента кључ са датом вредношћу
	dict(kolekcija)
	функција враћа речник ком је додала парове кључ-вредност генерисане на основу парова у из уређене колекције добијене у аргументу
		`x = dict()`
		`# x = {}`
		`x = dict(a=3, b=5)`
		`# x = {'a': 3, 'b': 5}`
		`y = dict({'a':4, 'c':7}, **x)`
		`# x = {'a': 3, 'c': 7, 'b': 5}`
		`x = dict( [(3,3), (4,5)] )`
		`# x = {3: 3, 4: 5}`

dict.fromkeys(iter, val = None)	Функција враћа речник у који је уписала кључеве из колекције `iter` и придружила им вредност `val`	`d = dict.fromkeys([2, 3, 5], 10)`
		`# d = {2 : 10, 3 : 10, 5 : 10}`
		`d = dict.fromkeys([2, 3, 5], None)`
		`# d = {2 : 10, 3 : 10, 5 : 10}`

d[k]	Вредност која је у речнику d придружена кључу k	`d = { 3 : 13, 4 : 5 }` `y = d[ 3 ] # y = 13`

d[k] = x	Додавање новог пара k:x, уколико кључ k не постоји у речнику.
	Придруживање вредности кључу k, уколико кључ у колекцији d постоји.
		`d = dict( { 3 : 3 , 4 : 5 } )`
		`d[ 10 ] = 9`
		`# d = { 3 : 3, 4 : 5, 10 : 9 }`
		`d[ 10 ] = 18`
		`# d = { 3 : 3, 4 : 5, 10 : 18 }`

del d[k]	Брисање кључа k из речника d.	`d = dict( {3 : 3, 4 : 5} )` `del d[ 3 ] # d = {4 : 5}`
d.clear()	Брисање свих кључ-вредност парова из колекције d.	`d = dict( {3 : 3, 4 : 5} )` `d.clear( ) # d = { }`
k in d	True уколико кључ постоји у речнику, иначе False.	`d = dict( {3 : 3, 4 : 5} )` `ima = 3 in d # d = True`

d.items()	Функција враћа листу торки (кључ, вредност)	`d = dict( {3 : 3, 4 : 5} )`
		`y = d.items( )`
		`# y = [(3, 3), (4, 5)]`

d.keys()	Функција враћа листу кључева који постоје у речнику d.	`d = dict( {3 : 3, 4 : 5} )` `y = d.keys() # y = dict_keys([3, 4])`

d1.update(d2)	Садржај речника **d1** се ажурира кључ-вредност подацима из **d2**.	`d1 = dict( {3 : 3, 4 : 5} )`
		`d2 = dict( {3 : 7, 9 : 10} )`
		`d1.update(d2)`
		`# d1 = {3 : 7, 4 : 5, 9 : 10}`

d.values()	Функција враћа листу вредности које су придружене кључевима у речнику **d**.	`d = dict( {3 : 3, 4 : 5} )`
		`y = d.values()`
		`# y = dict_values([3, 5])`

d.get(k [, def])	**d.get(k)**
	Функција враћа пар кључ-вредност за кључ k.
	d.get(k , def)
	Функција враћа пар кључ-вредност за кључ k, уколико кључ постоји, иначе враћа def.
		`d = dict( {3 : 3, 4 : 5} )`
		`y = d.get( 4 ) # y = 5`
		`y = d.get( 10 ) # y = None`
		`y = d.get( 10, 11 ) # y = 11`

Уграђене функције за рад са скуповима

Операција / Функција	Резултат	Пример употребе

set(iter)	Функција враћа скуп формиран од елемената листе/ниске iter, коју је добио у аргументу	`s = set( "petlja" )`
		`# s = {'t', 'p', 'j', 'a', 'l', 'e'}`
		`s = set( [3, 5, 2, 7] )`
		`# s = { 2, 3, 5, 7 }`

len(s)	Функција враћа број елемената скупа	`s = { 2, 3, 5, 7 }` `y = len( s ) # y = 4`
el in s / not in s	True уколико вредност el припада скупу s, иначе False.	`s = { 2, 3, 5, 7 }` `y = 2 in s # y = True`

s1.issubset(s2)	Функција враћа True уколико је s2 подскуп скупа s1, иначе враћа False.	`s1 = {2, 3, 5, 7}`
		`s2 = {2, 3}`
		`y = s2.issubset(s1) # y = True`
		`y = s1.issubset(s2) # y = False`

s1.issuperset(s2)	Функција враћа True уколико је s1 надскуп скупа s2, иначе враћа False	`s1 = {2, 3, 5, 7}`
		`s2 = {2, 3}`
		`y = s1.issuperset(s2)`
		`# y = True`
		`y = s1.issuperset(s2)`
		`# y = False`

s.add(el)	Функција додаје елемент el скупу s	`s1 = {2, "3"}`
		`s1.add("park")`
		`# s = {2, 'park', '3'}`

s.discard(el)	Функција избацује елемент el из скупа s	`s1 = {2, "3"}` `s1.discard("3") # s = { 2 }`
s.clear()	Функција брише све елементе скупа s	`s1.clear() # s = { }`

s1.intersection(s2[, s3...]) или s1 & s2	Функција враћа пресек скупова s1 и s2
	
	Вредност израза је скуп једнак пресеку скупова s1 и s2
		`s1 = {2, 3, 5, 7}`
		`s2 = {2, "3"}`
		`s3 = {2, 7, 13}`
		`k = s1 & s3`
		`# k = {2, 7}`
		`r = s1.intersection(s2, s3)`
		`# r = {2}`

s1.difference(s2[, s3...]) или s1 - s2	Функција враћа разлику скупа s1 и s2
	
	Вредност израза је скуп једнак разлици скупова s1 и s2
		`s1 = {2, 3, 5, 7}`
		`s2 = {2, "3"}`
		`s3 = {2, 7, 13}`
		`k = s1 - s2 # k = {3, 5, 7}`
		`r = s1.difference(s2, s3)`
		`# r = {3, 5}`

s1.symmetric_difference(s2) или s1 ^ s2	Функција враћа симетричну разлику скупова s1 и s2
	
	Вредност израза је скуп једнак симетричној разлици скупова s1 и s2
		`s1 = {2, 3, 5, 7}`
		`s2 = {2, "3"}`
		`s3 = {2, 7, 13}`
		`k = s1 ^ s2`
		`# k = {3, 5, 7, '3'}`
		`s1.symmetric_difference (s2)`
		`# r = {3, 5, 7, '3'}`

s.update(i1[, i2...])	Функција у скуп **s** додаје све елементе из колекција наведених у аргументима	`s = {5, 7}`
		`l1 = [2, "3"]`
		`l2 = [7, 13]`
		`s.update(l1,l2)`
		`# s = {2, '3', 5, 7, 13}`

Уграђене функције за рад са нискама

Функција	Вредност	Пример употребе

s.count(sub [, start[, end]])	**s.count(sub)**
	Функција враћа број непреклапајућих појављивања подниске sub у ниски s
	s.count(sub, start)
	Функција враћа број појављивања подниске sub у ниски s почевши од карактера са индексом start
	s.count(sub, start, end)
	Функција враћа број појављивања подниске sub у ниски s почевши од карактера са индексом start до карактера са индексом end-1
		s = "anananas"
		x = s.count("ana") # x = 2
		x = s.count("ana", 3) # x = 1
		x = s.count("ana", 2, 5) # x = 1
		
		# Непреклапајуће појаве постринга "ana" су на позицијама са индексом 0 и 4. За разлику од њих, преклапајуће појаве би биле на позицијама са индексом 0, 2 и 4.

s.find(sub [, start [, end]])	**s.find(sub)**
	Функција проналази прво појављивање ниске sub у ниски s и враћа индекс позиције карактера sub[0] у s.
	s.find(sub, start)
	Функција проналази прво појављивање ниске sub у ниски s почевши од карактера са индексом start и враћа индекс позиције карактера sub[0] у s.
	s.find(sub, start, end)
	Функција проналази прво појављивање ниске sub у ниски s почевши од карактера са индексом start до карактера са индексом end-1 и враћа индекс позиције карактера sub[0] у s.
	У случају да ниска sub није подниска ниске s функција враћа -1.
		s = "anananas"
		x = s.find("ana") # x = 0
		x = s.find("ana", 3) # x = 4
		x = s.find("ana", 2, 5) # x = 2
		
		x = s.find("ka") # x = -1

s.isalnum()	Функција враћа True уколико ниска садржи искључиво слова и цифре (алфанумеричке карактере), иначе враћа False	s = "anananas"
		x = s.isalnum() # x = True
		s = "15maj"
		x = s.isalnum() # x = True
		s = "15"
		x = s.isalnum() # x = True
		s = "15-maj"
		x = s.isalnum() # x = False

s.isalpha()	Функција враћа True уколико су сви карактери у ниски слова (алфабетски), иначе враћа False	s = "anananas"
		x = s.isalpha() # x = True
		s = "15maj"
		x = s.isalpha() # x = False
		s = "15"
		x = s.isalpha() # x = False
		s = "15-maj"
		x = s.isalpha() # x = False

s.isdigit()	Функција враћа уколико су сви карактери у ниски цифре, иначе враћа False	s = "anananas"
		x = s.isdigit() # x = False
		s = "15maj"
		x = s.isdigit() # x = False
		s = "15"
		x = s.isdigit() # x = True
		s = "15-maj"
		x = s.isdigit() # x = False

s.islower()	Функција враћа True уколико су сва слова која се појављују мала слова, иначе враћа False	s = "5lja"
		x = s.islower() # x = True
		s = "5ljA"
		x = s.islower() # x = False

s.isspace()	Функција враћа True уколико се ниска састоји само из бланко карактера, иначе враћа False	s = " "
		x = s.isspace() # x = True
		s = " . "
		x = s.isspace() # x = False

s.isupper()	Функција враћа True уколико су сви алфабетски карактери велика слова, иначе враћа False	s = "5LJA"
		x = s.islower() # x = True
		s = "5ljA"
		x = s.islower() # x = False

separator.join(seq)	Функција враћа ниску која настаје спајањем свих ниски из листе seq, добијене у аргументу, при чему је између сваке две ниске које се спајају уметнут знак separator.	`s = ["Вук","Стефановић","Караџић"]`
		`separator = " "`
		`x = separator.join(s) # x = "Вук Стефановић Караџић"`

s.lower()	Функција враћа ниску s у којој су сва велика слова замењена малим.	`s = "1. Maj "` `x = s.lower() # x = "1. maj"`

s.replace(old, new[, maxCount =-1])	**s.replace(old, new)**
	Функција враћа ниску једнаку s у којој су све (непреклапајуће) појаве подниске old замењене ниском new.
	s.replace(old, new, maxCount)
	Функција враћа ниску једнаку s у којој је првих maxCount (непреклапајућих) појава подниске old замењене ниском new.
		`s = "anananas"`
		`y = s.replace("an","pet")`
		`# x = "petpetpetas"`
		`y = s.replace("an","pet",2)`
		`# x = "petpetanas"`

s.rfind(sub[, start[, end]])	Функција има улогу аналогну фуникцији find, с том разликом што претрагу врши од последњег ка првом карактеру стринга.	`s = "anananas"`
		`x = s.rfind( "ana" ) # x = 4`
		`x = s.rfind( "ana", 3) # x = 4`
		`x = s.rfind( "ana", 2, 5 ) # x = 2`

s.split([separator[, maxsplit]])	**s.split()**
	Функција враћа листу ниски добијених издвајањем подниски из ниске, при чему се као раздвојник користи *blanko* карактер.
	s.split(separator)
	Функција враћа листу ниски добијених издвајањем подниски из ниске s, при чему се као раздвојник користи separator.
	s.split(separator, maxsplit)
	Функција враћа листу ниски добијених издвајањем подниски из ниске, при чему се као раздвојник користи карактер separator, а раздвајање се врши на највише maxsplit позиција.
		`s = "Вук Стефановић Караџић"`
		`y = s.split()`
		`# y = ['Вук', 'Стефановић', 'Караџић']`
		
		`s = "auto-put Beograd-Niš"`
		`y = s.split("-")`
		`# y = ['auto', 'put Beograd', ' Niš ']`
		`y = s.split("-",1)`
		`# y = ['auto', ' put Beograd-Niš']`

s.upper()	Функција враћа ниску s у којој су сва мала слова замењена великим.	`s = "1. Maj "` `x = s.upper() # x = "1. MAJ"`

МОДУЛИ

Константе и уобичајене функције модула math - наставак

Операција / Функција	Вредност	Пример употребе
e	2.7182818284590451	`x = math.e`
cos(x)	Функција враћа косинус угла мере x изражену радијанима	`y = math.cos(0) # y = 1.0`
degrees(x)	Функција изражава у степенима x, величину угла задату у радијанима	`x = math.pi` `y = math.degrees(x) # x = 180.0`
exp(x)	Функција враћа вредност степена чија је основица константа e, а изложилац x.	`x = 1 / 2` `y = math.exp(x) # x = 1.6487212707001282`
fabs(x)	Функција враћа апсолутну вредност броја **x** записану у облику реалног броја, без обзира на то да ли је **x** реалан или цео број.	`y = math.fabs(-4) # x = 4.0`
factorial(n)	Функција враћа вредност n!	`y = math.factorial(5) # x = 120`

log(x[, base])	**log(x)**
	Функција враћа вредност логаритма за основу e.
	log(x, base)
	Функција враћа вредност логаритма за основу base.
		`x = math.e`
		`y = math.log(x) # y = 1.0`
		
		`x = 100`
		`y = math.log(x,10) # y = 2.0`

radians(x)	Функција изражава у радијанима величину угла **x** задату у степенима.	`x = 180`
		`y = math.radians(x)`
		`# y = 3.141592653589793`

sin(x)	Функција враћа синус броја.	`y = math.sin(0) # y = 0.0`
tan(x)	Функција враћа тангенс броја.	`y = math.tan(0) # y = 0.0`