

Електротехничка школа „Стари град“, Београд
профил Електротехничар информационих технологија
предмет Програмирање, кроз четири године учења

1.разред, фонд часова 2 часа вежби седмично

материјал према лабораторијским вежбама са основама програмског језика Пајтон

материјал за ученике

верзија 1.0

јун 2021

обрађене теме:

- основе алгоритама
- графички приказ алгорита
- структура програма у Пајтону
- типови података
- улаз и излаз програма
- програмирање аритметичких израза
- условне структуре
- цикличне структуре
- увод у секвенце

наставник Ранковић Небојша

Алгоритамско решавање задатака.....	10
Алгоритам	10
Анализа и решавање задатака програмирањем.....	10
Питања и задаци за самосталан рад	11
Програмски језици	11
Особине и врсте програмских језика	11
Синтакса, семантика и прагматика	12
Метајезици.....	13
Питања и задаци за самосталан рад	13
Графички запис алгоритма	13
Стандардни дијаграм тока.....	13
Елементи дијаграма тока.....	13
Рад у апликацији FLOWGORITHM.....	16
001 Приказ унетог текста.....	17
002 Унос вредности у промењиве	19
Питања и задаци за самосталан рад	20
Алгоритам са простом линијском структуром	22
Проста линијска структура.....	22
003 Израчунати површину круга уносом полупречника круга	23
004 Исписивање имена и презимена ученика	23
005 Рачунање обима и површине квадрата	24
006 Број кругова у квадрату.....	25
Питања и задаци за самосталан рад	26
Алгоритам са разгранатом структуром	26
Разграната структура.....	26
007 Провера исправности унетог имена ученика.....	26
008 Корекција унетог броја на вредност 100 (верзија 1)	27
009 Корекција унетог броја на вредност 100 (верзија 2)	28
Математичке операције са једном промењивом	28
Провера парности броја	28
010 Претварање негативног у позитиван број.....	29
011 Ограничавање брзине кретања возила	30
012 Истраживање парности унетог броја	30
Питања и задаци за самосталан рад	31

Алгоритам са цикличном структуром	32
Циклична структура.....	32
013 Исписати на екрану све целе бројеве између 0 и 5	32
014 Исписати на екрану само парне бројеве између 0 и 10	33
015 Сабрати све бројеве између 0 и 5 и приказати коначан збир.....	33
016 Корисник уноси бројеве све до уноса 0.	34
017 Унос одговарајуће шифре	35
018 Приказ парних бројева помоћу разгранате структуре.....	36
019 У опсегу од 10 до 20 сабрати све бројеве који су дељиви са 3	36
Питања и задаци за самосталан рад	37
Пајтон интерпертер	37
Програми и програмирање	37
Компајлери и интерпретери.....	37
Пајтон окружење	38
Пајтон интерактивни мод	38
020 Куцање броја у интерактивном моду	39
021 Куцање неслужбене речи у интерактивном моду	39
022 Куцање службене речи у интерактивном моду	39
Питања и задаци за самосталан рад	39
Наредба излаза	40
Приказ излаза са функцијом print.....	40
023 Испис откуцаног текста у интерактивном моду	40
024 Испис откуцаних линија текста у скрипт моду	40
Основе рада са стринговима и стринг литералима	41
025 Приказ апострофа и наводника као део стринга	41
026 Приказ апострофа и наводника истовремено као део стринга	41
027 Приказ стринга у више линија на излазу	41
Питања и задаци за самосталан рад	42
Приказ података.....	43
Писање дугачких линија кода у више линија	43
Поништавање акције нове линије код функције print.....	44
Коришћење сепаратора ствари	44
Ескејп карактери.....	44
Приказ више стринг литерала помоћу оператора +	44
Питања и задаци за самосталан рад	45

Промењиве	46
Коментари	46
028 Употреба коментара у коду.....	46
Промењиве	47
Идентификатор и оператор доделе	47
Провера указивања на исту локацију	47
029 Указивање на исту меморијску локацију.....	47
Правила давања имена промењивима (избор идентификатора)	48
Константе	48
030 Записивање константи у коду	48
Питања и задаци за самосталан рад	48
Типови података	50
Нумерички типови података и литерали	50
Одређивање типа података	51
Давање промењивој вредност другачијег типа података.....	51
Конверзија типова података	51
Питања и задаци за самосталан рад	52
Наредба улаза	54
Читање улаза са тастатуре	54
031 Употреба наредбе улаза.....	54
032 Додела вредности промењивој наредбом улаза	54
Читање бројева са функцијом input	54
033 Конверзија унетог податка у целобројну вредност	55
034 Конверзија стринга у цео број нестовањем позива функције input	55
Питања и задаци за самосталан рад	55
Аритметички оператори	56
Математичке операције	56
Приоритет оператора.....	57
035 Приоритети аритметичких операција	57
Изрази са више типова података	57
036 Конверзија типова код израза са више типова	57
Питања и задаци за самосталан рад	58
Програмирање математичких израза	59

Манипулација цифрама.....	59
037 Цифре вишецифарског целог броја	59
038 Цифре вишецифарског реалног броја	59
Програмирање математичких израза	60
039 Програмирање математичких израза:.....	60
Питања и задаци за самосталан рад	60
Форматирање бројева	61
Приказ бројева	61
040 Приказ реалног броја без додатног форматирања	61
Предефинисана функција format.....	61
Форматирање у научној нотацији.....	61
Употреба зарез сепаратора	61
041 Зарез сепаратор	62
Минимална ширина поља.....	62
Форматирање целих бројева	62
Питања и задаци за самосталан рад	63
Релациони оператори	63
Булови изрази и релациони оператори	63
042 Рад са Буловим изразима	63
Оператори == и !=.....	63
043 Рад са операторима == и !=	64
Исказ if	64
Условне структуре	64
Исказ if	65
044 На екрану исписати поруку ако је унети број већи од 0.	65
045 На екрану исписати поруку ако унети број није једнак 10.	66
046 Повећање плате под условом продаје	66
047 Испис поруке под условом бодова	66
048 Испис вредности са једносмерном селекцијом.....	66
Питања и задаци за самосталан рад	67
if-else исказ.....	67
Увлачење линија кода код if-else исказа	68
049 Додела вредности преко испуњеног услова.....	68
050 Провера тренутне температуре.....	68
051 Рачунање зараде у односу на радне сате	69

052 Вредност у опсегу са две if-else петље	69
053 Испис вредности са двосмерном селекцијом.....	70
Питања и задаци за самосталан рад	70
Упоредивање стрингова	70
Тестирање вредности стрингова.....	70
054 Упоредивање идентичности стрингова.....	70
Друге врсте упоређивања стрингова.....	71
055 Упоредивање стрингова различитих дужина	72
Питања и задаци за самосталан рад	72
Угнеждене условне структуре	73
Тестирање више од једног услова	73
056 Испитивање услова за позајмицу	74
Питања и задаци за самосталан рад	75
Исказ if-elif-else.....	75
if-elif-else исказ	75
057 Конверзија бодова са теста у описну оцену	76
058 Провера вредности броја у датом опсегу вредности.....	76
059 Регулација тренутног нивоа воде.....	77
Питања и задаци за самосталан рад	77
Логички оператори	78
Логички оператори у Пајтону	78
Оператор and	78
Оператор or	78
Оператор not.....	78
Процена кратког споја (short-circuit evaluation)	78
Провера опсега бројева	79
060 Строжи услови за позајмицу новца	79
061 Блажи услови за позајмицу новца	79
Питања и задаци за самосталан рад	80
Искази услова	81
Булове промењиве.....	81
062 Употреба флегова	81
063 Конверзија нестоване петље у if-elif-else исказ	81
064 Магичан датум	82

065 Математички искази са условима	82
Питања и задаци за самосталан рад	82
while петља	84
Петље у програмирању.....	84
While петља.....	84
066 Рад while петље	85
067 Корисничка контрола петље	85
Питања и задаци за самосталан рад	86
for петља	86
Петља контролисана бројачем	86
068 Приказ бројева из уређене листе	87
069 Приказ бројева из неуређене листе	87
070 Петља for са стринговима у листи	87
Употреба функције range са for петљом	87
071 Употреба функције range	87
072 Замена функције range() листом	88
073 Исписивање стринга одређени број пута	88
074 Употреба два аргумента у функцији range	88
075 Употреба три аргумента у функцији range	88
Питања и задаци за самосталан рад	88
Бесконачне петље	89
Техника бесконачне петље.....	89
Коришћење циљне промењиве унутар петље	89
076 Приказ целих бројева и њихових квадрата	89
Корисничко управљање итерацијама	90
077 Корисничка контрола броја итерација	90
Постављање аргумената range функције	90
078 Рад са функцијом range	90
Питања и задаци за самосталан рад	91
Акумулатор у петљи	92
Рачунање тренутне суме.....	92
079 Сабирање бројева помоћу петље и акумулатора	92
Појачани оператори доделе.....	93
Стражари	93
080 Коришћење стражара	93

081 Пребројавање непарних бројева у листи.....	93
Питања и задаци за самосталан рад	94
Нестоване петље	94
Петље одобравања улаза	94
Нестоване петље	95
082 Могући резултати у фудбалу.....	95
083 Цртање карактерима правоугаоника димензија 6x8	96
084 Цртање правоуглог троугла карактерима	96
085 Цртање косе линије карактерима	96
086 Цртање обрнутог правоуглог троугла карактерима	96
Питања и задаци за самосталан рад	97
Излазак из петље	98
Безусловно искакање из петље	98
087 Проста употреба команде break.....	98
Безусловни скок на следећу итерацију петље.....	98
088 Проста употреба команде continue.....	98
089 Употреба команде break са for петљом	99
090 Употреба команде break са while петљом	99
091 Употреба команде continue са for петљом	99
092 Употреба команде continue са while петљом	99
Питања и задаци за самосталан рад	99
Дефиниција секвенци	100
Секвенце (низови)	100
093 Итерација преко торке	100
Рад са торкама	101
094 Оператор понављања.....	101
095 Оператор понављања код торки.....	101
096 Садржај постојеће торке доделити новој торци.....	101
097 Приказати поруку преко празне торке.....	101
098 Торка са различитим типовима података	102
099 Креирање торке преко функције range().	102
0100 Креирање торке функцијом range() са опсегом.....	102
Питања и задаци за самосталан рад	102
Торке.....	102
Итерација по торци са for петљом.....	103
0101 Итерација по торци са for петљом	103
Индексирање	103

0102 Штампање елемената торке помоћу петље	103
0103 Негативни индекси у торци	103
0104 Функције над торкама	104
0105 Креирање торке са 5 копија торке ("а", "с").....	104
0106 Штампање елемената торке помоћу петље while.	104
Питања и задаци за самосталан рад	104
Коришћена литература	105

Алгоритамско решавање задатака

Алгоритам

Алгоритам је процедура која се састоји од одређеног броја корака којима се решава дати проблем у коначном времену.

Програмирање омогућава да се рачунари (хардвер) користи да би се реализовао одређени задатак.

Анализа и решавање задатака програмирањем

Коришћење рачунара за решавање проблема и задатака значи креирање програма (програмирање) у неком од програмских језика.

Процес програмирања се састоји од више етапа :

Поставка проблема

У извршавању ове етапе заједно учествују корисник (за кога се пише програм) и програмер и то на природном језику.

Јасно се одређује циљ програма, анализирају се постојеће информације и описују сви подаци.

Анализа проблема

У овој етапи се дефинишу улазни и излазни подаци, ограничења њихових вредности и прецизно дефинишу везе између података.

Резултат ове фазе је формалан опис проблема, израда математичког модела који се може реализовати на рачунару.

Разрада алгоритма

Дефинише се начин на који се од улазних података долази до излазних.

Бира се оптималан начин решавања проблема (број потребних операција, време извршења алгоритма, коришћење меморијских ресурса).

Пројектовање опште структуре програма

Овде се врши избор програмског језика.

Задатак се дели у логичке целине, а свака целина се може и даље поделити.

Дефинише се начин чувања података тј. структура података.

Кодирање

Описивање претходно дефинисаних поступака решавања проблема и података у програмском коду.

Превођење, извршавање и тестирање програма

Коришћење компајлера / интерпретера; повезивача / билдера (креирање ехе фајла); провера правилног рада програма.

Израда документације

У документацији треба описати шта програм ради, дати упутства како се користи, детаљније описати алгоритам.

Документација се води током свих етапа.

Одржавање програма

Током коришћења програма се могу уочити неке грешке.

Некада треба извршити промене у програму услед нових околности.

Може доћи до проширивања програма додавањем нових функционалности.

Питања и задаци за самосталан рад

Питања

1. Шта је алгоритам?
2. Шта је програмирање?
3. Навести називе етапа у процесу програмирања.
4. У којој етапи у процесу програмирања учествује и корисник, за којег се прави програм?
5. Описати етапу анализе проблема.
6. У којој етапи у процесу програмирања се бира програмски језик са којим ће се кодовати програм?
7. Шта је кодирање?
8. У којој етапи се користи интерпретер?
9. Током које етапе се води документација о завршеним процесима програмирања?
10. На шта се односи одржавање програма?

Програмски језици

Особине и врсте програмских језика

У компјутеру се користи процесор као главни део рачунара и који се користи за обраду свих података и инструкција.

Програмирање омогућава да се рачунар (хардвер) користи да би се реализовао одређени задатак.

Програмер (softver developer) је особа која има вештине и знање неопходне за дизајнирање, креирање и тестирање компјутерских програма.

Алат којим се служи програмер за решавање задатака се назива програмски језик.

Програмски језици се сврставају према времену настанка и намени у генерације програмских језика:

1.генерација програмских језика

40-тих година 20.века се појављују први рачунари који обрађују програмерски код у машинском језику.

Машински језик је низ инструкција које одговарају појединим процесорским операцијама у виду низа битова (0,1).

Програмирање и исправљање грешака машинског језика је било изузетно компликовано и мали број људи се тада бавио програмирањем.

2.генерација програмских језика

50-тих година се појављују асемблерски програмски језици који су донекле прилагођени човеку. Инструкције и даље означавају поједине процесорске операције али се сада могу записати и речима које се лакше разумеју и памте.

Проблем је што и даље зависе од структуре и особина самог процесора на којем се примењују па је и даље смањена употребљивост на различитим типовима компјутера.

3.генерација програмских језика

60-тих година се појављују виши програмски језици који су независни од хардвера рачунара.

Њихова примењивост се повећава употребом помоћних програма (компајлери, интерпретери) који преводе код из програмског језика у машински код.

Сада је неопходно да се само инсталира на компјутер одговарајући преводац за одговарајући програмски језик.

Најстарији је FORTRAN (1957), намењен је нумеричком решавању математичких проблема.

Тада је био популаран COBOL (60-тих) намењен изградњи пословних апликација.

Програмски језик BASIC (70-тих) је један од првих језика опште намене који је долазио и преинсталиран на рачунарима.

Програмски језик C је најпопуларнији процедурални структурални програмски језик настао крајем 60-тих година.

Повезао је ефикасност (коју имају асемблери) са једноставности у писању кода програмских језика 3.генерације.

И данас се користи као језик за комуникацију и управљање великог броја комплексних уређаја и машина.

80-тих година настају објектно орјентисани програмски језици где се уводи појам објекта са међусобном интеракцијом између објеката.

Данас су најпопуларнији C++ (1983., проширење C-а, користи се где је битна брзина извршавања инструкција), Јава (1991., извршава се на виртуелној машини па зато није завистан од хардвера), C# (Microsoft-ов одговор на Јаву) и Пајтон.

Пајтон је програмски језик који ће се користити као алат и окружење за изучавање програмирања.

Постоје програмски језици четврте (за рад са базама података) и пете генерације (један облик програмирања путем вештачке интелигенције).

Синтакса, семантика и прагматика

Програмски језик има своју синтаксу и семантику.

Синтакса

Синтакса је скуп правила који се морају стриктно поштовати приликом писања програма.

Синтаксна правила указују како се морају користити службене речи, оператори и други симболи у програму.

Синтаксна правила су граматичка правила програмског језика: како је нешто написано.

Синтаксна правила су фиксна и непромењива током кодовања и зато програмер мора да их познаје.

Семантика

Семантика је наука која изучава значење исписаних линија кода програмског језика.

Дешава се да је код правилно синтаксно написан али није семантички па програм не даје добре резултате.

Семантичка правила су логичка правила програмског језика: зашто је нешто написано.

Семантика указује на логичност, сврсисходност и ефикасност делова кода у програму.

Семантичке грешке се често не пријављују као грешке при куцању кода већ се откривају тек после извршења програма.

Прагматика

Прагматика се бави конкретном употребом језика у комуникацији пошто за људске језике није довољно само познавати граматiku и знати логичност језика.

Прагматика омогућава да исти изрази у различитим ситуацијама имају различито значење.

Програмски језици не смеју имати елементе прагматике јер њихово изражавање мора бити недвосмислено и једнозначно.

Искази

Појединачне инструкције које се користе за писање програма се називају искази (statements). Исказ у програмирању се састоји од службених речи, оператора, симбола и других елемената који ако су правилно поређани реализују операцију и дају очекивани резултат. Сваки програмски језик има операторе који извршавају различите операције над подацима (операндима).

Метајезици

У програмирању се користе посебни метајезици који служе за описивање неког другог језика. Метајезици често користе неки језик да би се лакше описала примена и правила неког другог формалнијег језика. Пример је метајезик XML који се користи за опис XHTML језика који се користи за приказ веб страна на интернету. Једна од верзија метајезика су псеудојезици који користе људски језик за опис рада алгоритма.

Питања и задаци за самосталан рад

Питања

1. Ко је програмер и чиме се он служи у свом послу?
2. Зашто се програмски језици деле по генерацијама?
3. Шта је машински језик?
4. Описати асемблерске програмске језике.
5. Навести ране програмске језике 3.генерације.
6. Навести особине савремених програмских језика.
7. Шта је то синтакса програмског језика?
8. Шта је то семантика програмског језика?
9. Зашто програмски језици не смеју да имају елементе прагматике?
10. Шта су искази и од чега се састоје?
11. Шта су оператори и операнди?
12. Објаснити сврху метајезика у програмирању.

Графички запис алгоритма

Стандардни дијаграм тока

Графички запис алгоритма се представља у облику дијаграма тока. Постоји више начина представљања елемената дијаграма тока а овде ће се користити такозвани стандардни са трапезоидима. Стандардни дијаграм тока је орјентисана структура која се састоји од одређеног броја дефинисаних елемената дијаграма тока између елемента START (Main) и елемента KRAJ (End). Дијаграм тока се састоји од елемената: блокови различитих облика, гране (стрелице) и тачке. Редослед обраде радњи је одређен редоследом блокова дуж орјентисане путање кроз структуру.

Елементи дијаграма тока

START и KRAJ

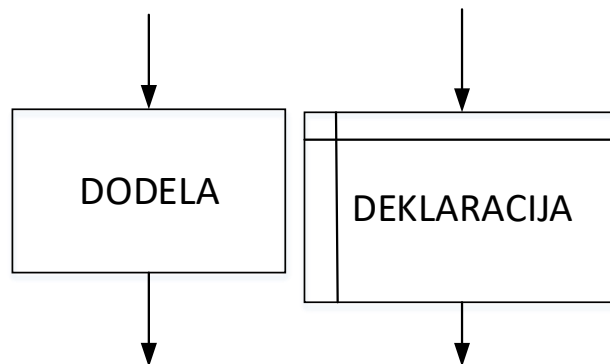
Елемент START (Main) се увек црта на почетку дијаграма тока. Има једну излазну грану која упућује на даљи ток алгоритма.

Елемент KRAJ (End) се црта увек на завршетку дијаграма тока.
Има једну улазну грану којом се остатак алгоритма упућује на овај елемент.



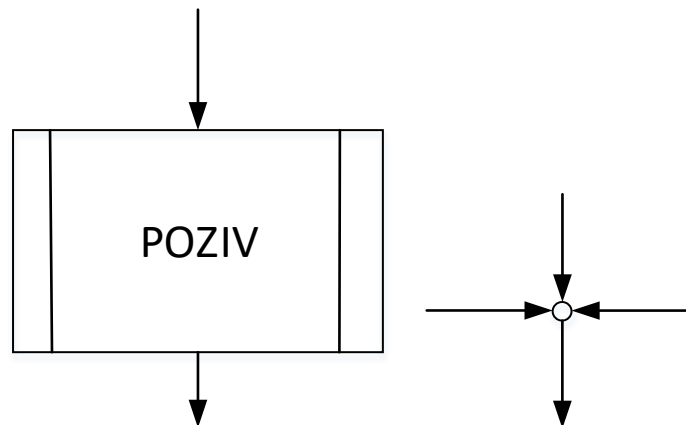
DODELA и DEKLARACIJA

Елемент DODELA (Assign) се користи за смештање резултата израчунавања у промењиву.
Елемент DEKLARACIJA (Declare) се користи за креирање промењивих.
Промењиве се користе за смештање података и вредности током извршавања програма.



POZIV и KOLEKTOR

Елемент POZIV (Call) пребацује контролу кода на функцију.
Информације које се преносе у функцију се називају аргументи.
Елемент KOLEKTOR (Collector) омогућава спајање путања у дијаграму тока без обраде или испитивања података.
Састоји се од више улазних грана и једне излазне гране.



ULAZ и IZLAZ

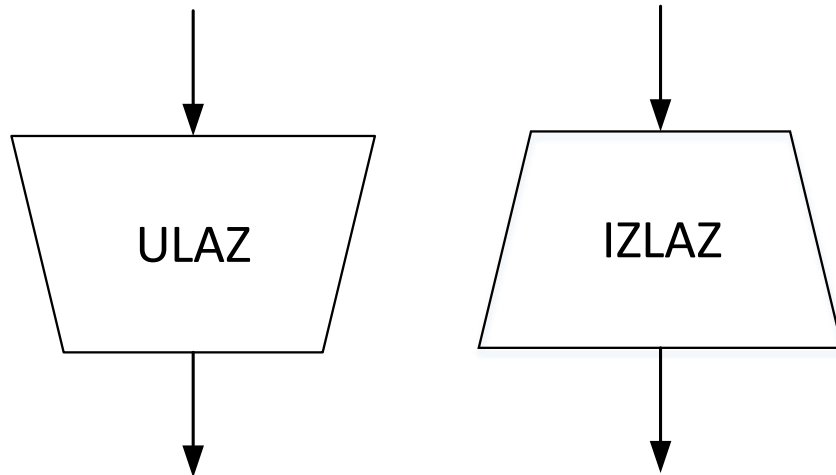
Елемент ULAZ (Input) прекида извршење кода и чека унос података са улазног уређаја рачунара (тастатура, миш, скенер...).

Састоји се од једне улазне и једне излазне гране.

Садржај елемента је низ промењивих чије се конкретне вредности очекују од корисника.

Елемент IZLAZ (Output) приказује вредности промењивих или текст преко неког од излазних уређаја рачунара (монитор, штампач...).

Састоји се од једне улазне и једне излазне гране.



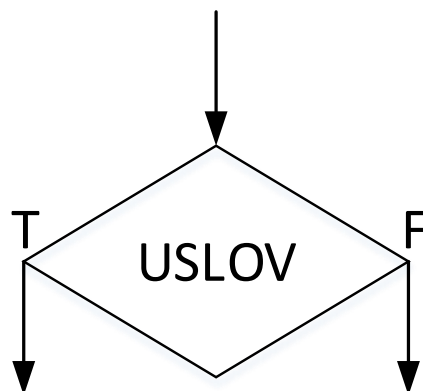
USLOV

Елемент USLOV (If) служи за гранање у дијаграму.

Садржај елемента је логички услов.

Ако је услов испуњен, даља обрада података се наставља дуж гране обележене као T (True).

Ако услов није испуњен, даља обрада података се наставља дуж гране обележена као F (False).



PETLJE

Елемент са леве стране је FOR PETLJA.

Петља инкрементира (повећава за један) вредност променљиве која се назива бројач, кроз дати OPSEG вредности.

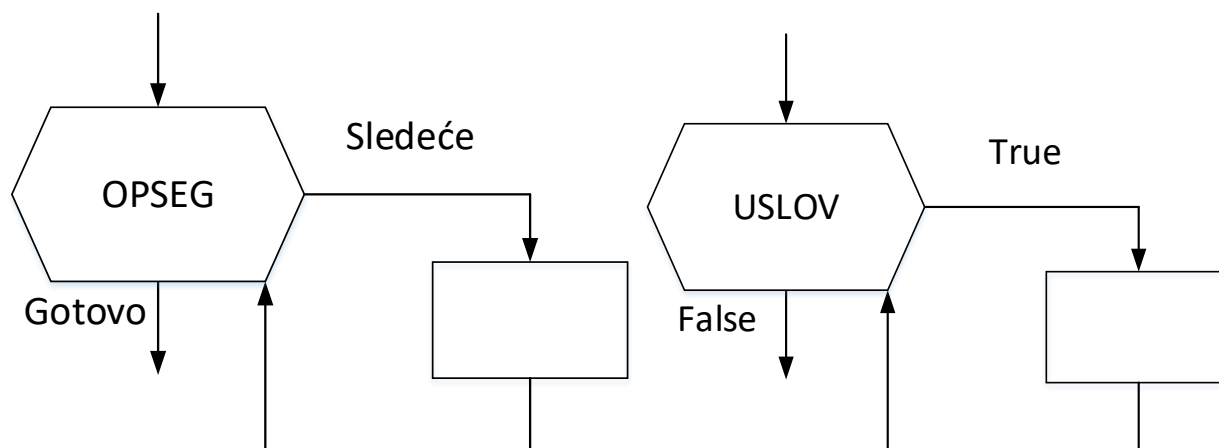
Све док се вредност бројача инкрементира, извршаваће се одређена обрада података у телу петље.

Елемент са десне стране је WHILE PETLJA.

Петља евалуира (процењује) израз који се назива USLOV.

Ако је услов тачан, извршиће се нека врста обраде а затим се поново евалуира исти услов.

Када услов више није тачан, излази се из опсега петље.

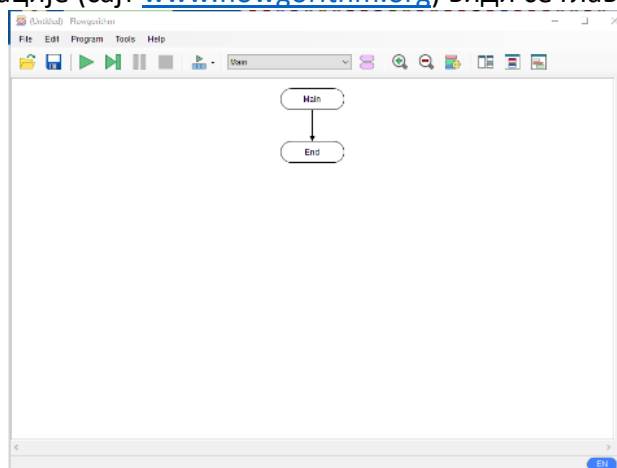


Рад у апликацији FLOWGORITHM

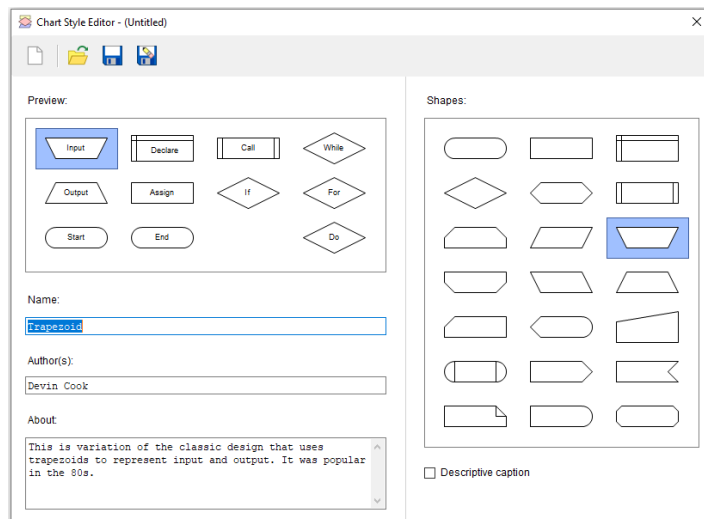
Апликација FLOWGORITHM је програмски језик за почетнике заснован на једноставним графичким елементима дијаграма тока алгоритма.

Креирање графичког приказа алгоритма омогућава да се програмер усредсреди на најбитније концепте и методе које чине креирање алгоритма.

После инсталације апликације (сајт www.flowgorithm.org) види се главни екран апликације.



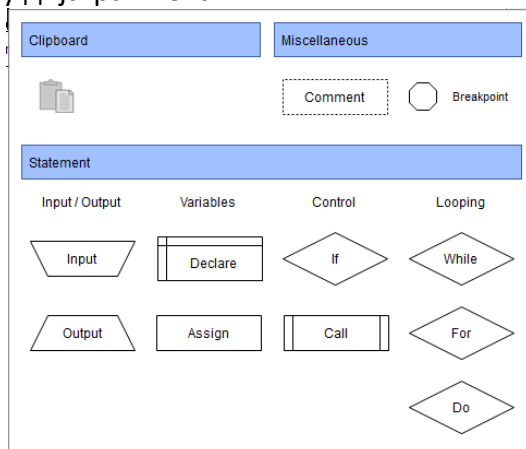
У опцији Edit->Chart Style Editor изабрати стандардни трапезоидни стил за облике приказа елемената дијаграма тока:



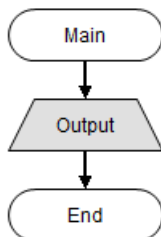
001 Приказ унетог текста

Избором опције File->Save As... бира се локација смештања и назив фајла на којем се тренутно ради.

Ако се кликне на стрелицу између Main и End елемената, појављује се нов прозор са избором елемента који се жели унети у дијаграм тока.



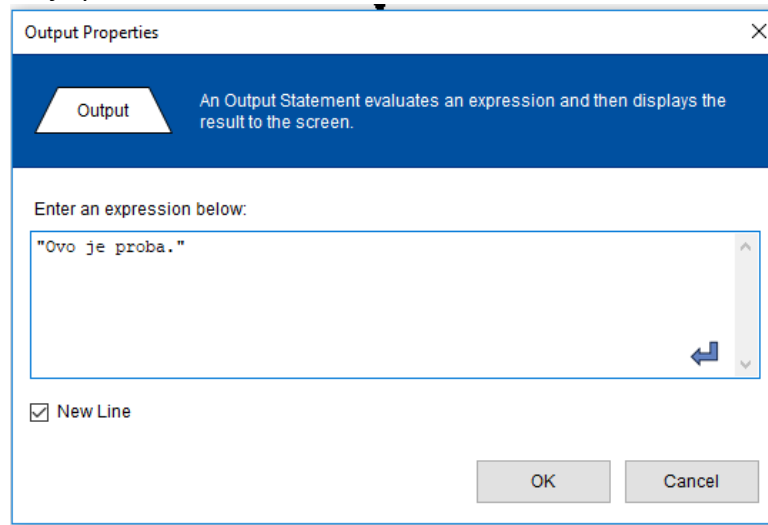
Клик на елемент Output.



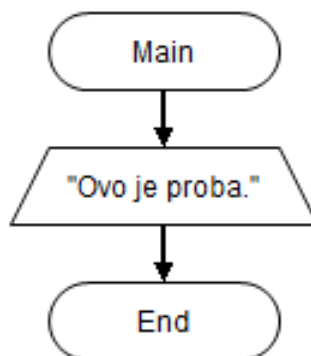
Двоструки клик на сам елемент Output и отвара се прозор са особеностима тог елемента.

Централни део прозора служи за унос текста или промењиве које се желе приказати на екрану.

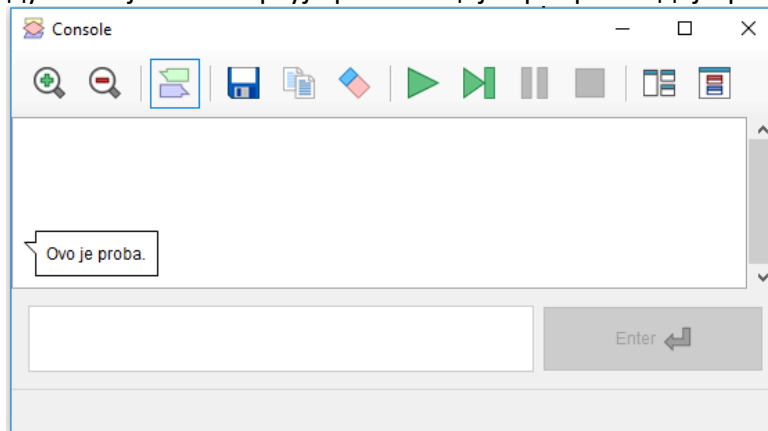
Унети само текст: "Ovo je proba."



После клика на ОК добија се текст као садржај елемента Output.



Клик на зелено Play дугме којим се стартује реализација креираног дијаграма тока.



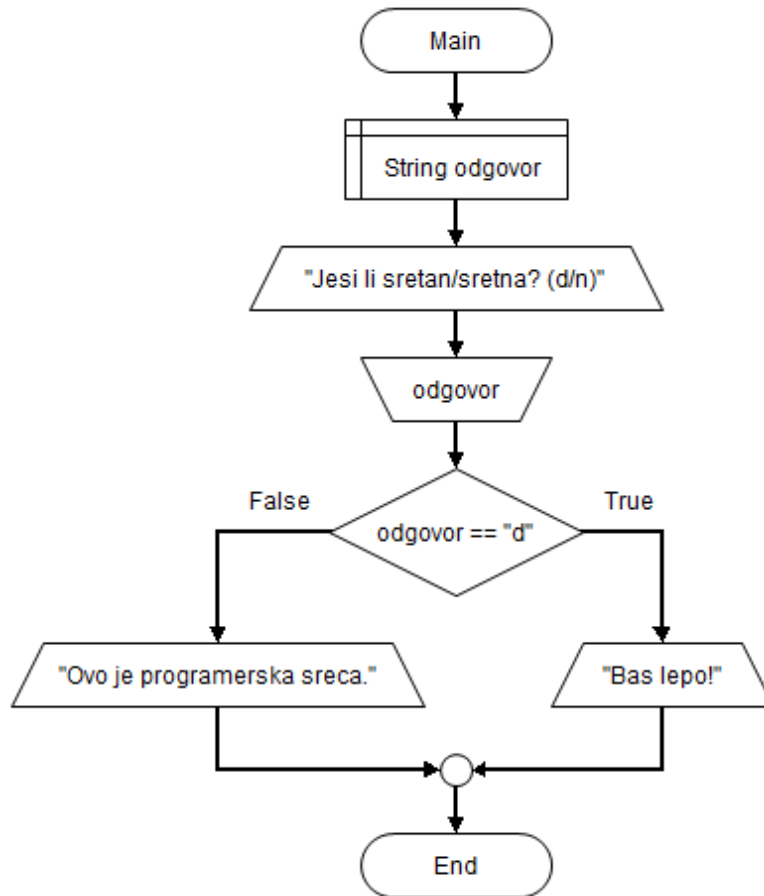
Појављује се нов прозор који се назива конзола.

Конзола се користи за приказ и унос текстуалних информација.

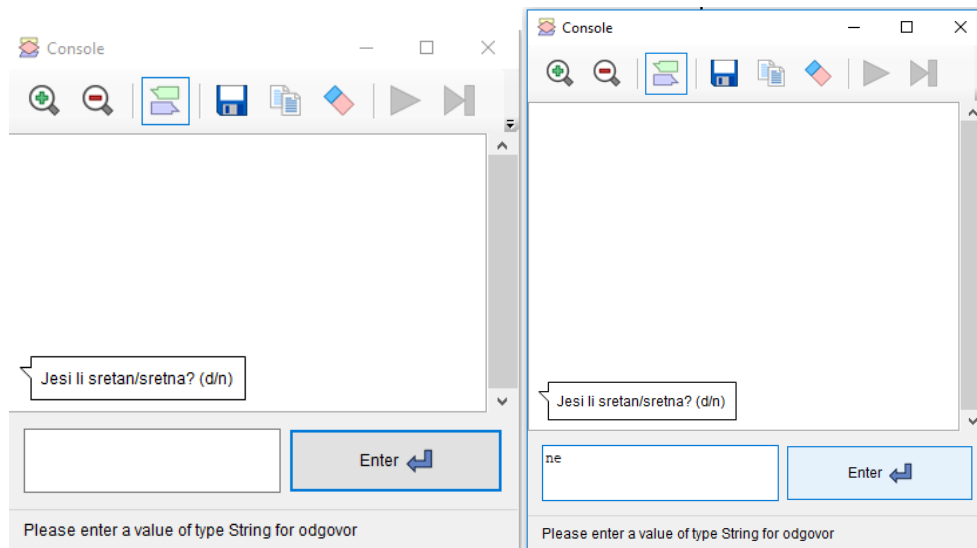
На конзоли се види као резултат дијаграма тока, унети текст у елементу Output.

002 Унос вредности у промењиве

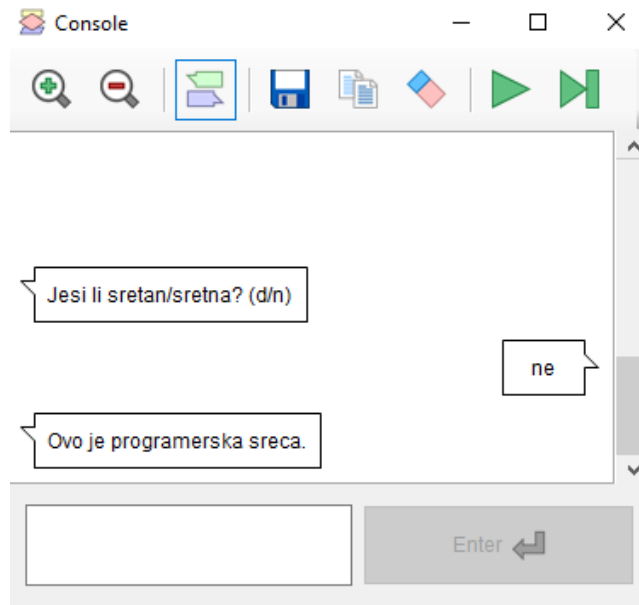
Креирати следећи дијаграм тока и назвати га Sreca:



После стартовања дијаграма тока, појављује се прозор конзоле за унос вредности као одговора на постављено питање.



Унети одговор се појављује на десној страни конзоле, а резултат унете вредности се поново појављује на левој страни екрана конзоле.



Питања и задаци за самосталан рад

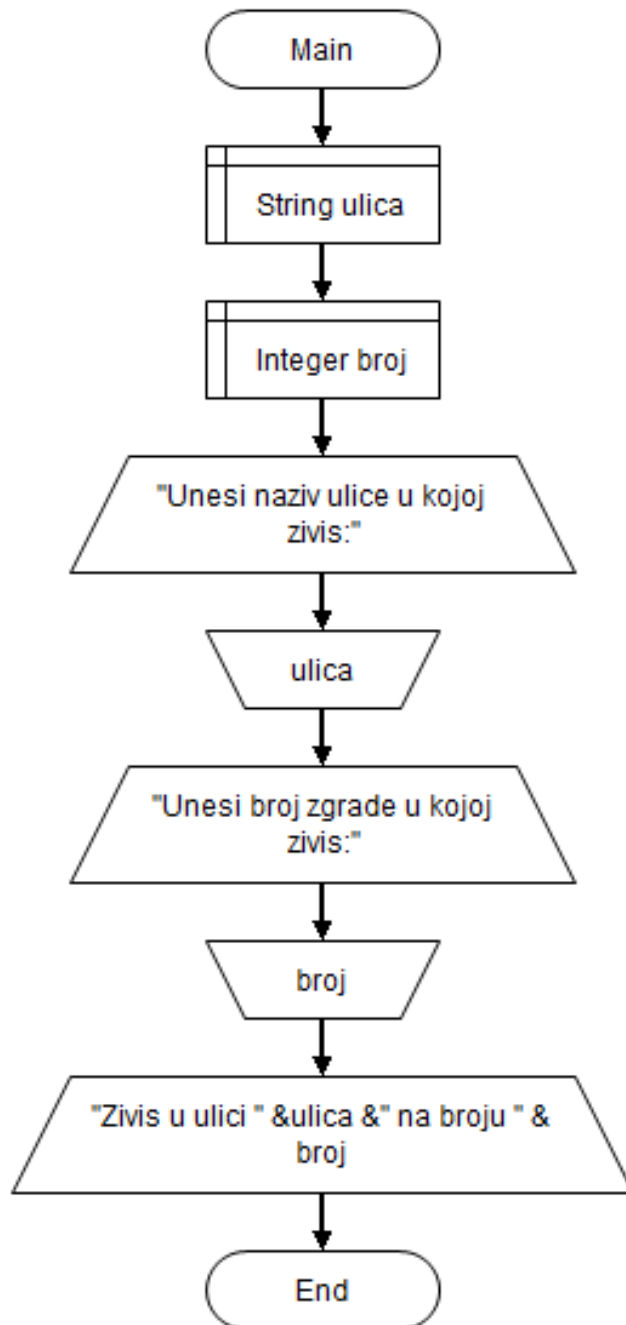
Питања

1. У којем облику се представља графички запис алгоритма?
2. Шта је стандардни дијаграм тока и од чега се састоји?
3. Које врсте елемената чине дијаграм тока?
4. Чему служе елементи START и KRAJ?
5. Описати улоге елемената DODELA и DEKLARACIJA.
6. У чему је разлика између елемената ULAZ и IZLAZ?
7. Шта је то грађање у дијаграму тока?
8. Које врсте петљи се користе у дијаграму тока?
9. За шта се користи апликација FLOWGORITHM?

Задаци

001 Унос дијаграма тока Adresa

Коришћењем апликације FLOWGORITHM унети дијаграм тока Adresa, стартовати дијаграм тока и током реализације програма уносити одговарајуће податке у програм.



Одговорити на следећа питања:

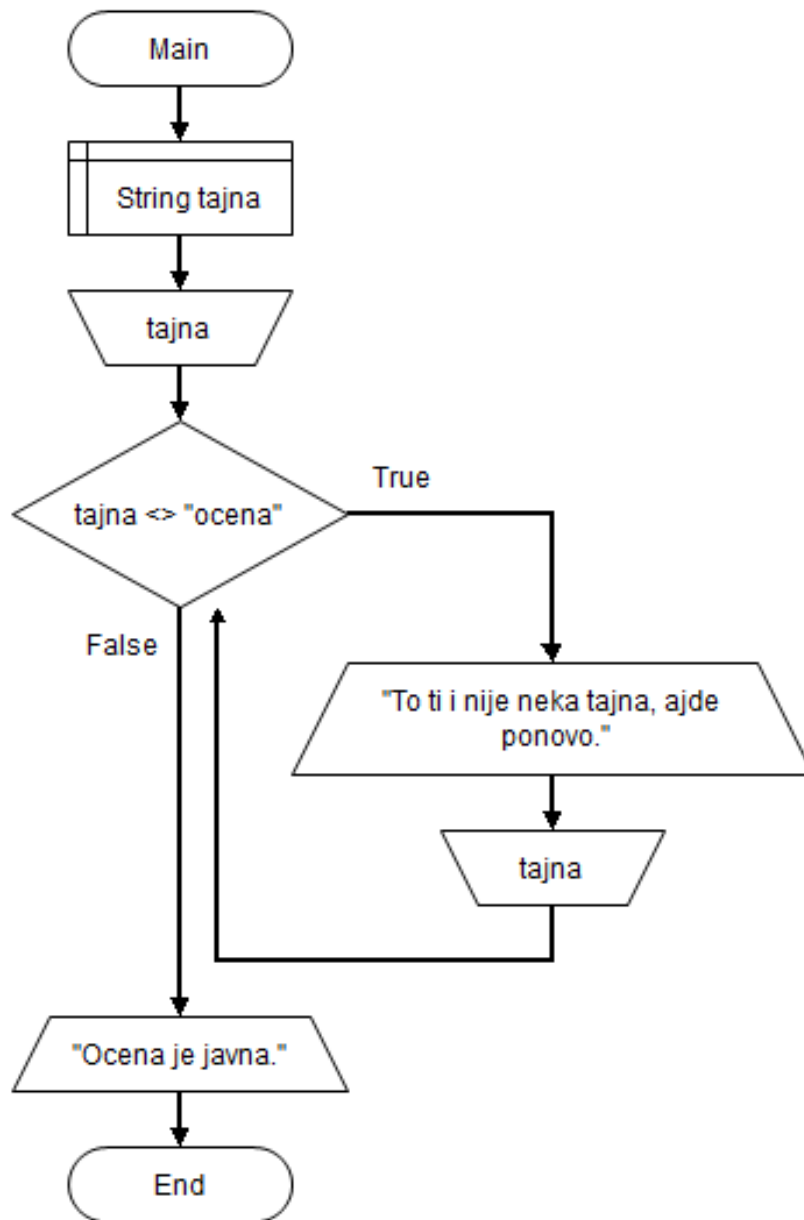
- 1) Који елементи се користе у датом дијаграму тока и колико пута се сваки од њих користи?
- 2) Шта су ulica и broj?
- 3) Шта би се десило када би се обрнуо редослед декларације за broj и ulicu?

002 Модификација дијаграма тока Sreca

Коришћењем апликације FLOWGORITHM променити одговоре на постављено питање у дијаграму тока Sreca.

003 Унос дијаграма тока Тајна

Коришћењем апликације FLOWGORITHM унети дијаграм тока Тајна.



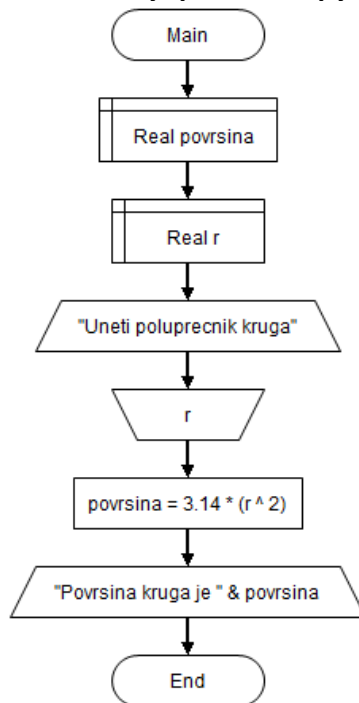
Алгоритам са простом линијском структуром

Проста линијска структура

Основни облик алгоритма је алгоритам са простом линијском структуром.

Њега карактерише дијаграм тока у којем нема рачвања, нема испитивања услова нити алтернативних делова линија кода, већ се цео програм одвија од прве до последње линије кода без прескакања линија.

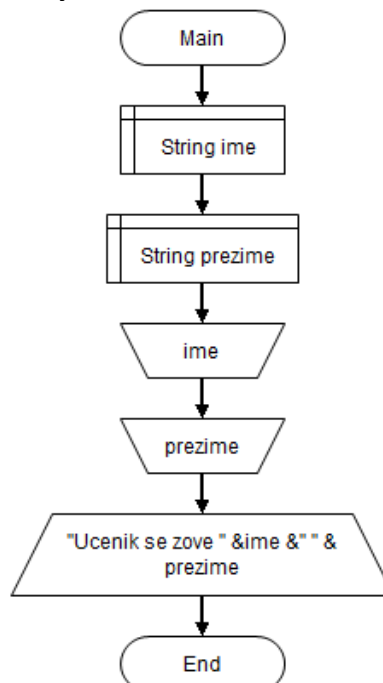
003 Израчунати површину круга уносом полупречника круга



За рад са промењивима у овој апликацији прво је неопходно декларисати сваку од промењивих, а то значи дефинисати који тип података ће бити смештен у ту промењиву, пре него се промењива по први пут у коду користи.

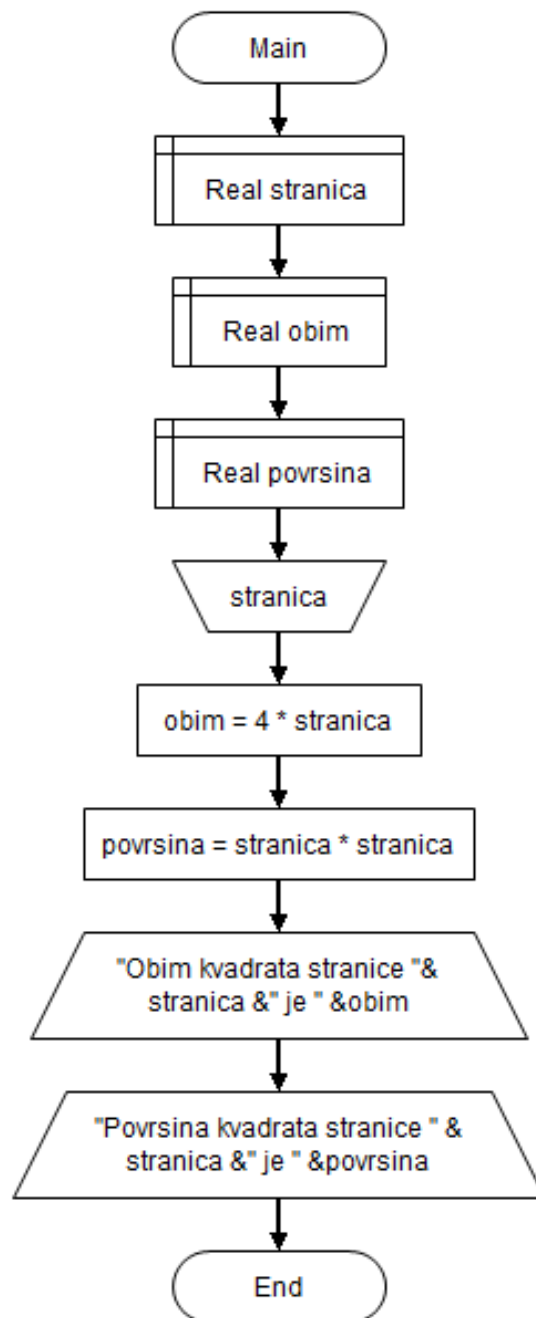
Ако је потребно израчунати неку вредност коришћењем формуле, неопходно је добити све податке за све промењиве које се налазе у формули пре коришћења саме формуле.

004 Исписивање имена и презимена ученика



005 Рачунање обима и површине квадрата

Коришћењем апликације FLOWGORITHM креирати алгоритам у облику дијаграма тока који за унуту страницу квадрата рачуна и приказује обим и површину квадрата.

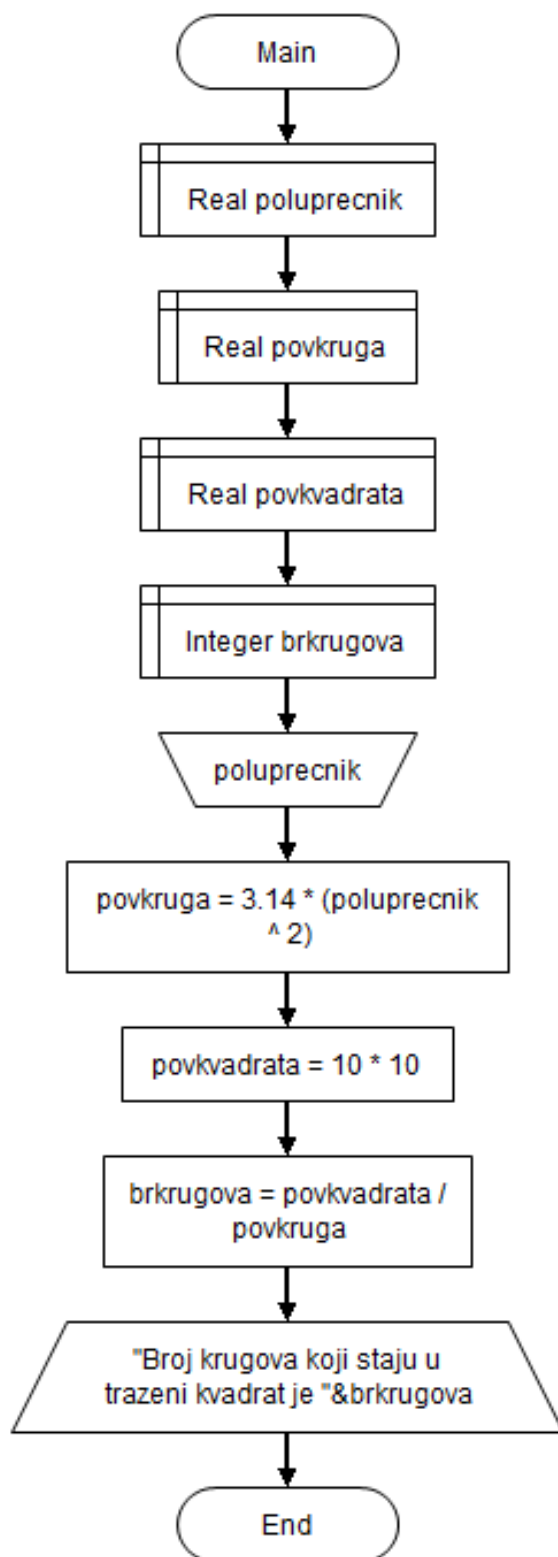


Одговорити на следећа питања:

- 1) Који елементи се користе у добијеном дијаграму тока и колико пута се сваки од њих користи?
- 2) Колико математичких формула се користи за реализацију алгоритма и помоћу којих елемената дијаграма тока се оне реализују?
- 3) Тестирати добијени алгоритам са страницама квадрата 5.0 и 0 и на основу добијених резултата извести закључке.

006 Број кругова у квадрату

Коришћењем апликације FLOWGORITHM креирати алгоритам у облику дијаграма тока који за унет полупречник круга испитује колико таквих кругова може стати у квадрат странице 10.



004 Креирати алгоритам који даје обим и површину правоугаоника за унете странице.

005 Креирати алгоритам који од корисника тражи два цела броја па на екрану исписује њихову аритметичку средину.

006 Креирати алгоритам који за дату дијагоналу квадрата рачуна обим и површину квадрата.

Алгоритам са разгранатом структуром

Разграната структура

Један од сложених облика алгоритма је алгоритам са разгранатом структуром.

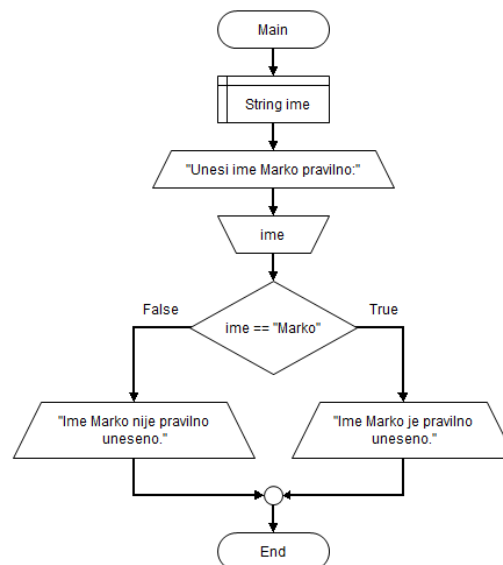
Њега карактерише дијаграм тока у којем постоји макар једно рачвање, а то значи да макар у једној линији кода правац реализације кода има алтернативу.

То значи да дијаграм тока алгоритма са разгранатом структуром мора имати макар један елемент USLOV.

Овај елемент обезбеђује да се, у зависности од испуњености постављеног услова, реализација једног дела кода уопште и не изврши.

Овакви алгоритми се користе у апликацијама где реализација кода зависи од уноса од стране корисника (видео игре, мултимедијалне апликације, бизнис апликације...)

007 Провера исправности унетог имена ученика



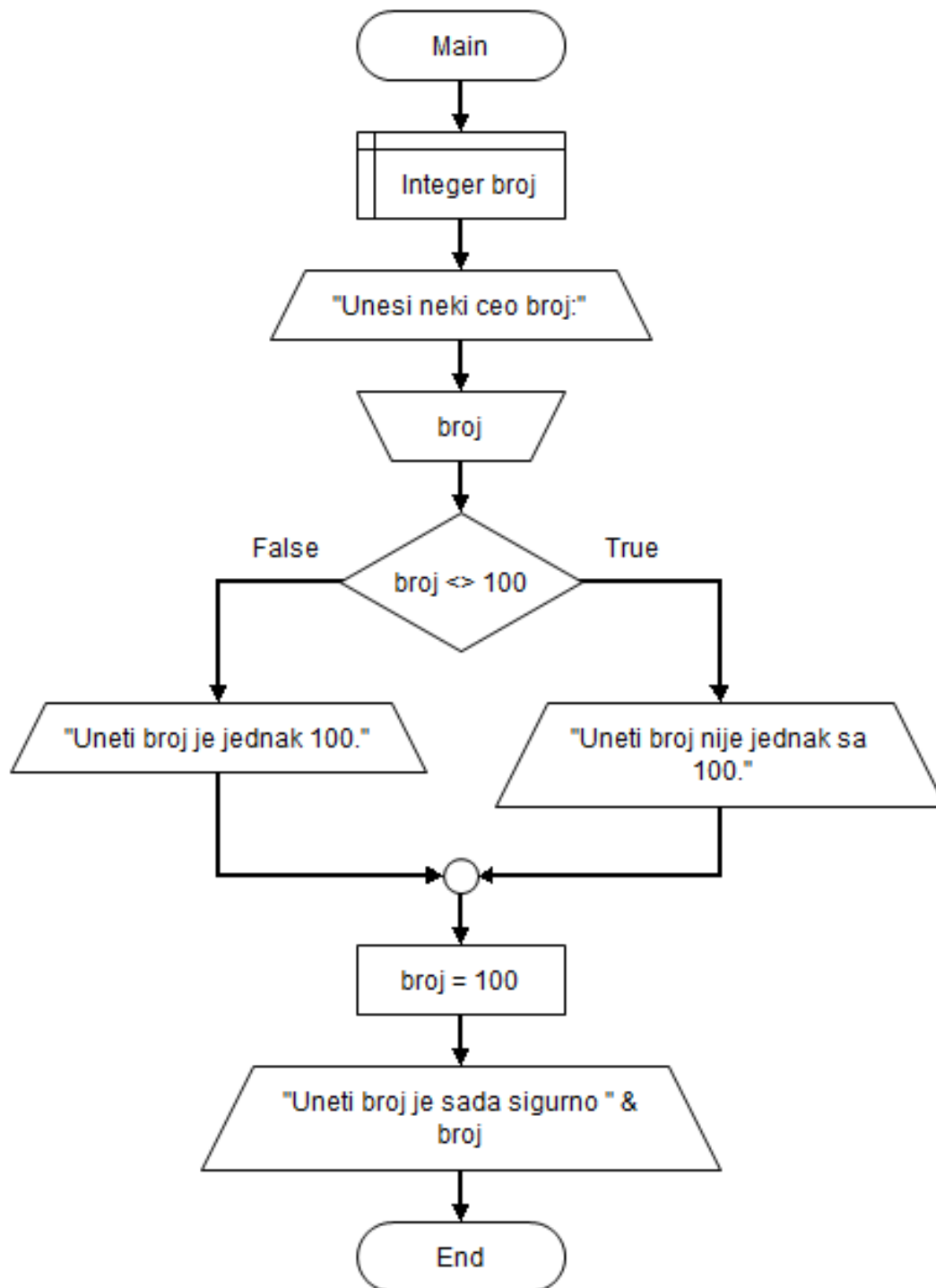
Приметити да је за рад са елементом USLOV најважније правилно постављање неједначине за испитивање тачности услова.

Услов се мора поставити тако да постоје само два могућа одговора, тачно или нетачно.

У примеру 007, један од два стринга на излазу се неће уопште појавити на екрану у зависности од тачности услова.

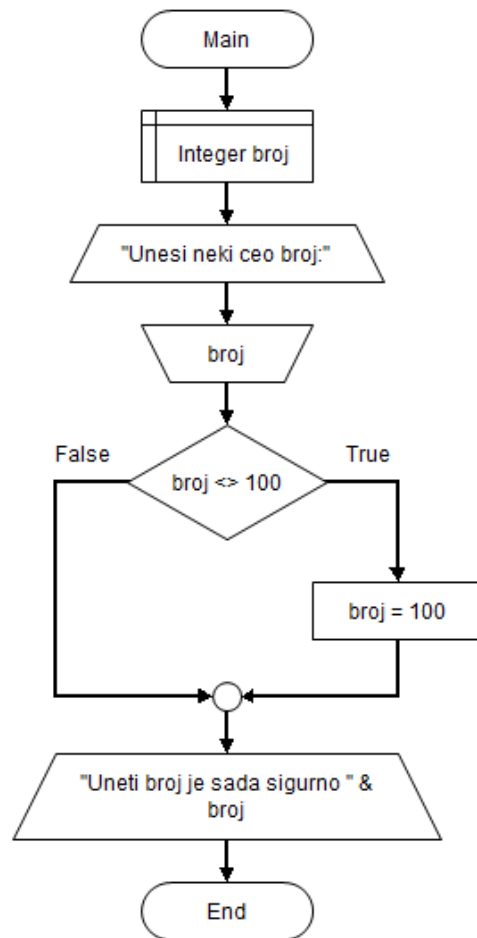
Остатак кода, од елемента KOLEKTOR, је заједнички за све алтернативе путање кроз код.

008 Корекција унетог броја на вредност 100 (верзија 1)



У овом примеру услов испитује неједнакост унетог броја са бројем 100. Резултат True се добија ако број није једнак са 100. Део кода после KOLEKTORA је заједнички и у њему се врши корекција унете вредности. Приметити да овде се корекција извршава чак и ако је вредност унетог броја једнака 100.

009 Корекција унетог броја на вредност 100 (верзија 2)



У примеру 009 корекција унетог броја на вредност 100 се извршава само ако је унети број различит од вредности 100.

Овде се корекција вредности извршава у грани која одговара случају тачности услова о неједнакости броја са 100.

Заједнички део алгоритма после KOLEKTORA само приказује резултат провере и евентуалне корекције вредности у промењивој broj.

Математичке операције са једном промењивом

Коришћење следећег математичког израза са једном промењивом: $a = a + 1$, значи да се тренутна вредност у промењивој а сабира са 1 и добијени резултат додељује истој промењивој а.

На исти начин се користе и следеће математичке операције: -, *, /.

Провера парности броја

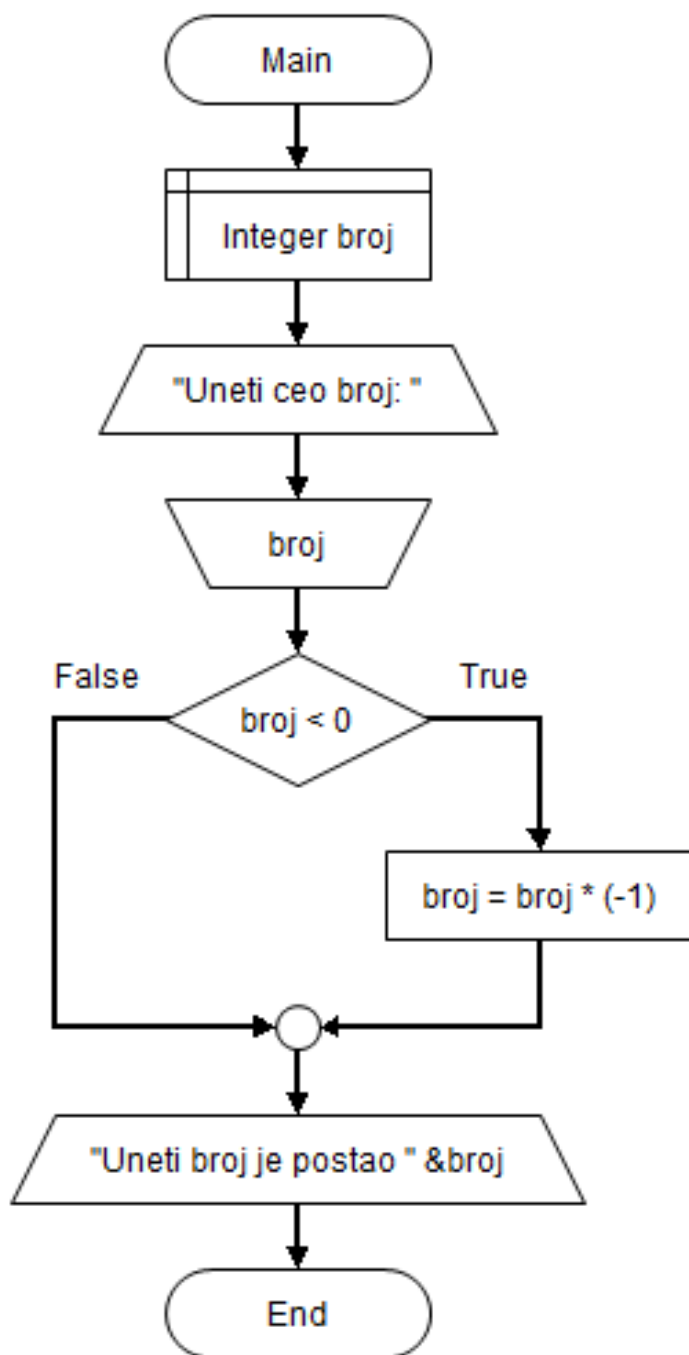
Због особине парних бројева да су дељиви са 2, тј да после дељења са 2 њихов остатак дељења је 0, парност бројева се може проверити употребом математичке операције модуо (%).

$4 \% 2 = 0$, пошто је 4 целобројно дељиво са 2, остатак дељења 4 са 2 је 0

$5 \% 2 = 1$, пошто 5 није целобројно дељиво са 2, остатак дељења 5 са 2 је 1

010 Претварање негативног у позитиван број

Коришћењем апликације FLOWGORITHM креирати алгоритам у облику дијаграма тока који, ако је унети број негативан, претвара га у позитиван са истом апсолутном вредности.

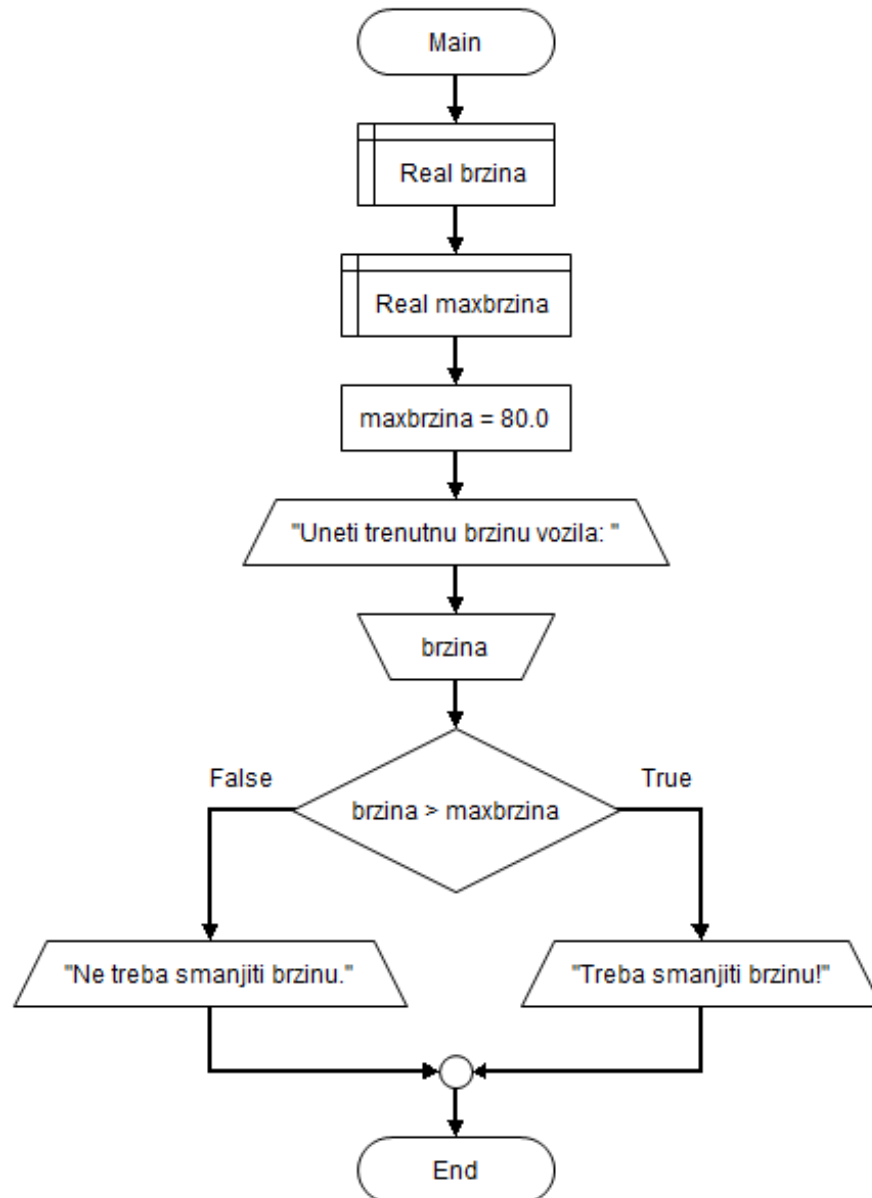


Одговорити на следећа питања:

- 1) Објаснити употребу елемента дијаграма тока USLOV и употребу елемената на две гране.
- 2) Како променити само елементе у једној грани елемента USLOV па да алгоритам реагује и на позитиван унети број тако што ће га претворити у негативан са истом апсолутном вредности?

011 Ограничавање брзине кретања возила

Коришћењем апликације FLOWGORITHM креирати алгоритам у облику дијаграма тока који проверава да ли је унета брзина кретања возила већа од 80 km/h и ако јесте, приказује поруку о потреби смањења брзине или како је не треба смањити.



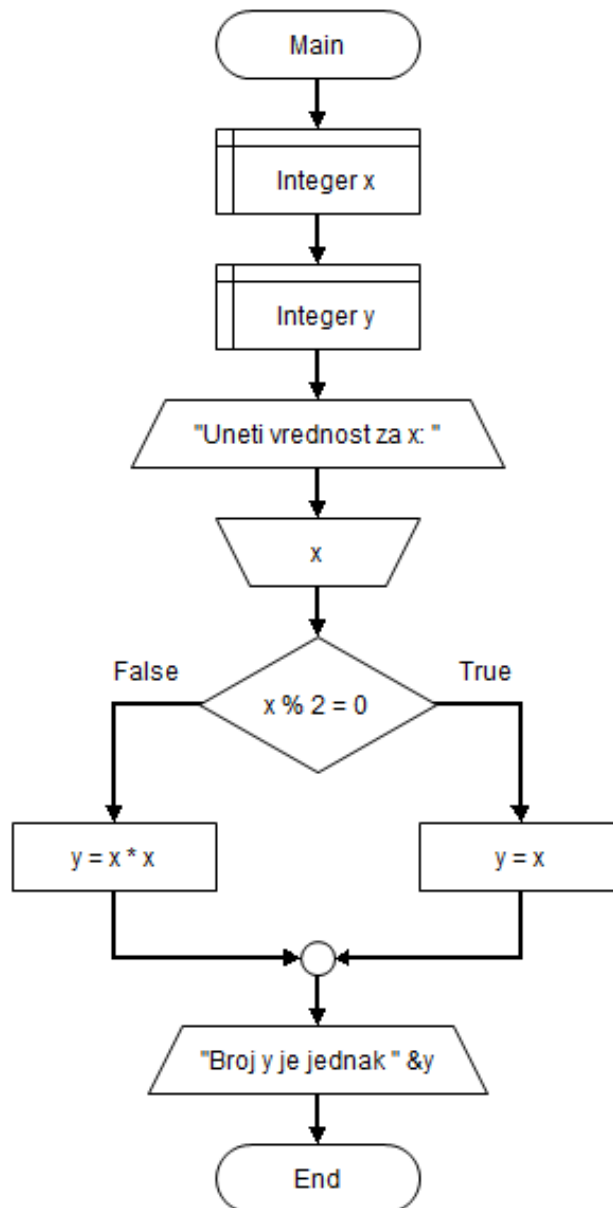
012 Истраживање парности унетог броја

Коришћењем апликације FLOWGORITHM креирати алгоритам у облику дијаграма тока који проверава унети цео број x .

Ако је x паран, броју y додељује вредност броја x .

Ако је x непаран, броју y додељује квадрат броја x .

$$y = \begin{cases} x, & x \% 2 = 0 \\ x * x, & x \% 2 \neq 0 \end{cases}$$



Питања и задаци за самосталан рад

Задаци

007 Креирати алгоритам који за два унета броја испитује који је већи и приказати одговарајућу поруку.

008 Креирати алгоритам цртањем дијаграма тока ако је:

$$z = \begin{cases} x + y, & x - y < 2 \\ x - y, & \end{cases}$$

009 Ако су дате све три странице троугла, открити да ли је троугао правоугли.

Алгоритам са цикличном структуром

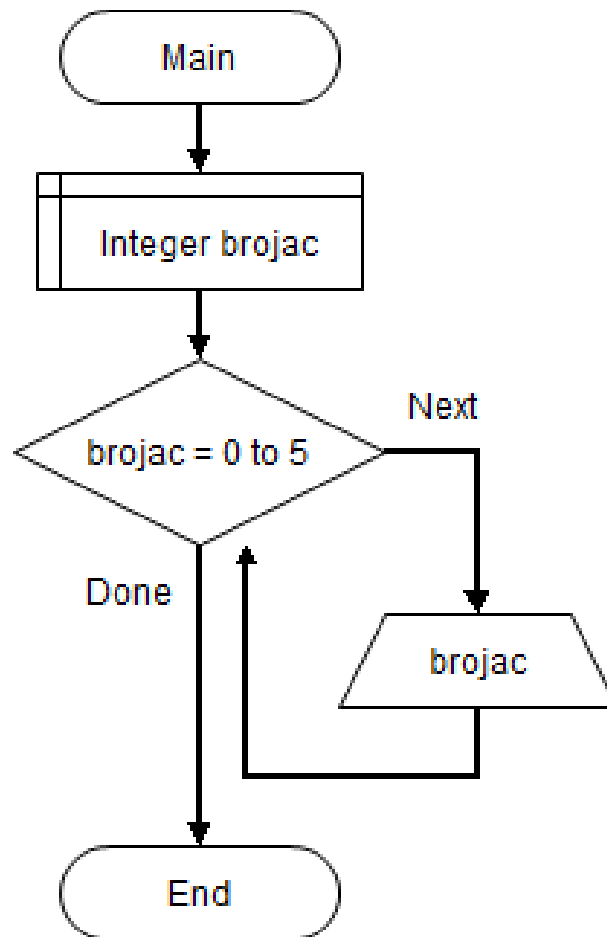
Циклична структура

Један од сложених облика алгоритма је алгоритам са цикличном структуром.

Овакав алгоритам се користи када је неопходно више пута реализовати исти код.

Колико пута ће се реализовати исти код зависи од врсте петље која се користи у алгоритму.

013 Исписати на екрану све целе бројеве између 0 и 5



У алгоритму се користи само једна промењива и то је промењива brojac.

У задатку је дефинисано да се посматра опсег вредности између 0 и 5.

Приликом дефинисања елемента FOR PETLJA дефинише се промењива, стартна вредност, коначна вредност и корак по којем ће се прелазити са једне вредности на другу унутар датог опсега.

Промењива brojac у сваком циклусу добија нову вредност која је дефинисана у опсегу.

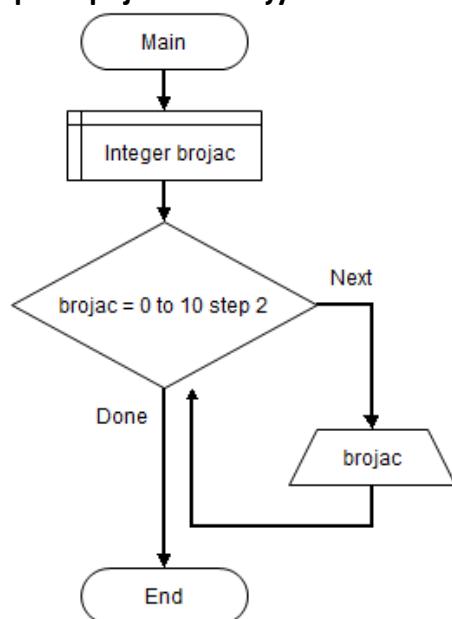
За сваки циклус (итерација) реализује се онај елемент који је дефинисан у грани Next.

У примеру то је елемент IZLAZ који само приказује тренутну вредност промењиве brojac.

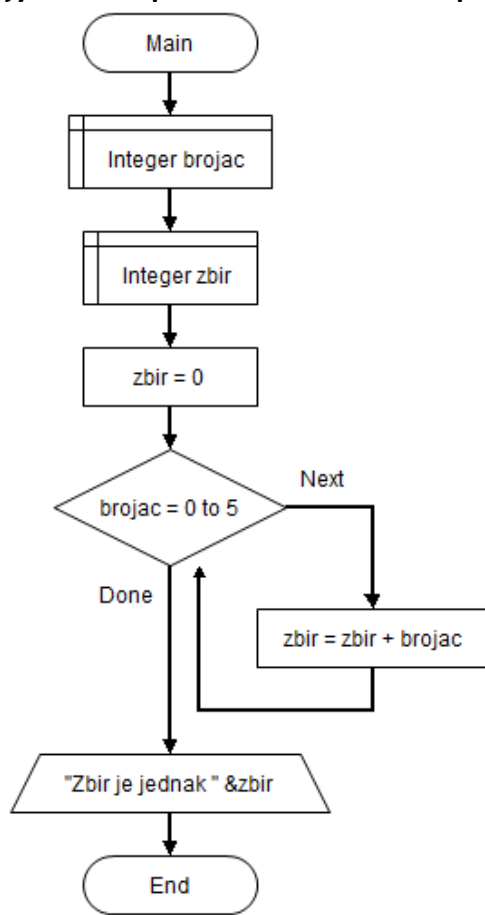
По дифолту, подразумева се да је вредност корака 1, па се то и не приказује као вредност унутар елемента FOR PETLJA.

Ако се промени вредност за корак на било коју другу вредност, онда ће се то и видети на елементу FOR PETLJA.

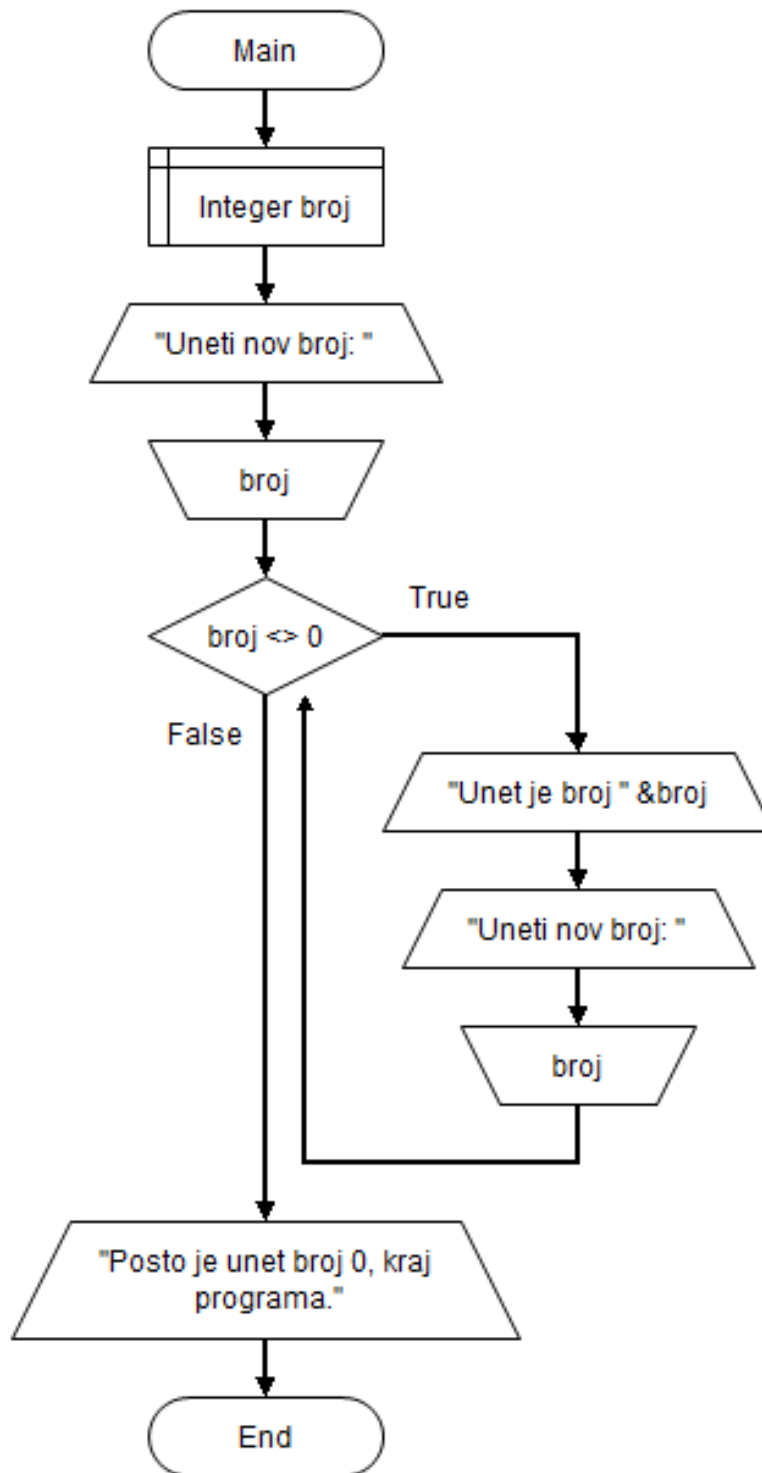
014 Исписати на екрану само парне бројеве између 0 и 10



015 Сабрати све бројеве између 0 и 5 и приказати коначан збир



016 Корисник уноси бројеве све до уноса 0.



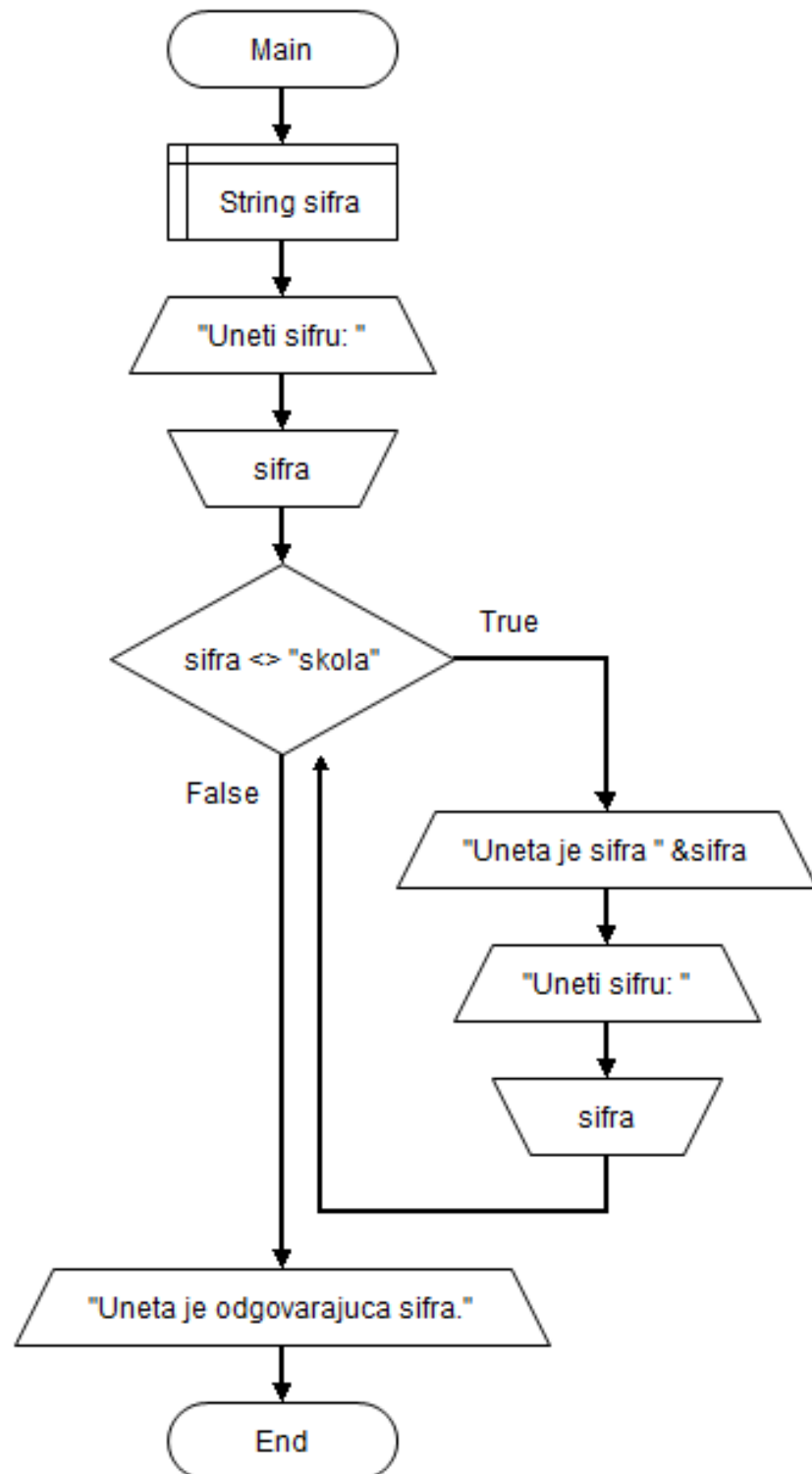
Из дијаграма тока се види да се у елементу WHILE PETLJA испитује услов broj <> 0.

Ако је услов испуњен, извршава се низ елемената који су у True грани.

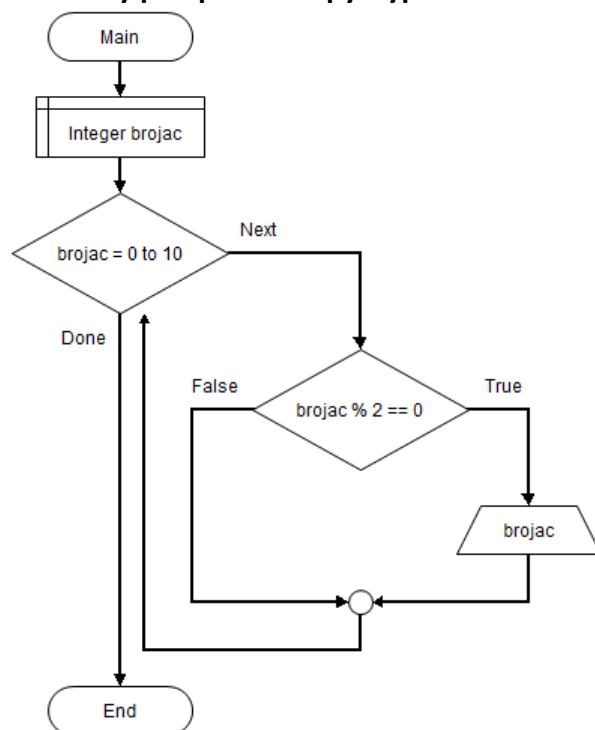
Тај низ елемената ће се извршавати све док корисник не унесе број 0 чиме више неће бити испуњен услов.

017 Унос одговарајуће шифре

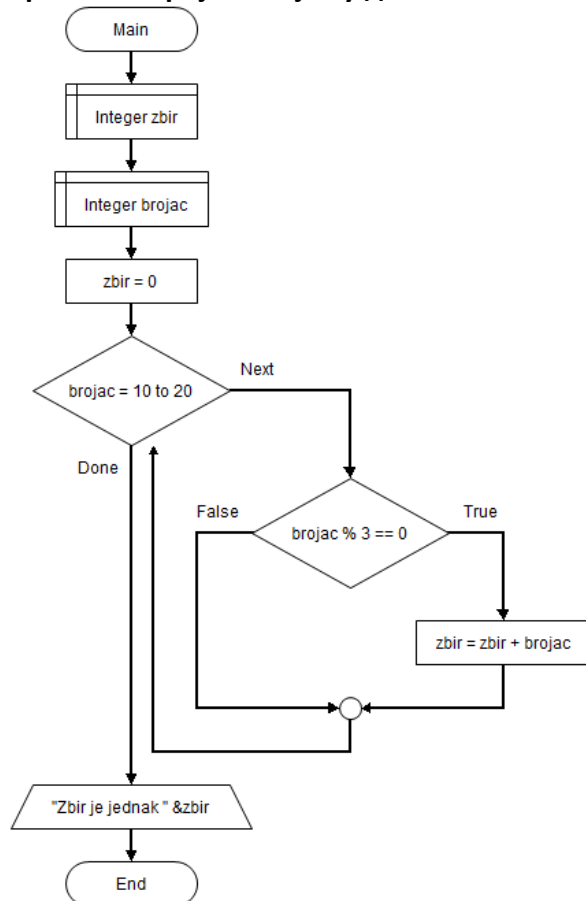
Коришћењем апликације FLOWGORITHM креирати алгоритам у облику дијаграма тока који кориснику не допушта излазак из петље све док не унесе одговарајућу текстуалну шифру.



018 Приказ парних бројева помоћу разгранате структуре



019 У опсегу од 10 до 20 сабрати све бројеве који су дељиви са 3



Питања и задаци за самосталан рад

Задаци

010 Сабрати све бројеве у опсегу између 100 и 200 који су дељиви са 5

011 Приказати на екрану само оне бројеве које уноси корисник а који су дељиви са 4 и ако број није дељив са 4 приказати поруку

012 Корисник уноси два цела броја па сабрати све бројеве између та два броја

Пајтон интерпертер

Програми и програмирање

У рачунару се подаци смештају као низ 0 и 1 које представљају битове информација.

Слова су такође низ битова пошто се свако слово претвара у облик броја помоћу ASCII (American Standard Code for Information Interchange) кода (нпр слово А има бројну вредност 65).

Низ битова процесор препознаје као податке пошто је такав начин представљања података у машинском језику.

Такође процесор добија на своје улазе програмске инструкције у виду низа битова, извршава их и на своје излазе шаље низ битова који престављају резултате унетих инструкција.

Овај процес се назива дохвати-декодуј-изврши (fetch-decode-execute) циклус.

Пошто је низ инструкција тешко писати у машинском језику осмишљени су разни програмски језици да олакшају процес писања програма.

Прво је осмишљен асемблерски језик а затим и низ програмских језика вишег нивоа.

Сваки програмски језик има посебне команде или службене речи које се могу користити само на тачно одређен начин.

Службене речи за програмски језик Пајтон:

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

Сваки програмски језик има и операторе који извршавају различите операције над подацима (операндима).

Компајлери и интерпретери

Пошто процесор разуме само машински језик (низ 0 и 1), програми написани у програмским језицима морају да се преводе на машински језик пре њиховог извршавања.

Компајлер је програм који преводи цео програм написан у програмском језику у посебан програм на машинском језику.

Пајтон користи интерпретер (преводаца) који преводи инструкције (делове програма) у машински језик и одмах их и извршава.

Компајлер реализује компајлирање (превођење) и извршавање кода у два одвојена корака.

Интерпретери комбинују превођење и извршавање програма инструкција по инструкција па зато они ни не праве посебан програм у машинском коду.

Компајлери су нешто спорији, док интерпретери не чувају међурезултате кодова у програму.

Изворни код (source code) или само код, су искази које програмер пише у програмском језику.

Ови искази се превode преко компајлера или интерпретера у машински код који процесор извршава.

Али ако постоји грешка у синтакси кода, јавља се синтаксна грешка које не дозвољава да се код до краја изврши.

Тада програмер мора да очистио код од синтаксних грешака да би се код могао извршити.

Пајтон окружење

Пајтон је објектно орјентисани, интерпретни програмски језик који се може користити на многим пословима у облику од кратких скрипти па све до кода за целокупне апликације.

Да би се састављали програми у програмском језику Пајтон неопходно је инсталирати апликације које то омогућавају.

Апликације који су потребене за куцање кода и креирање програма су Пајтон интерпретер и Пајтон едитор.

За потребе наставе користиће се Microsoft Visual Studio 2017 Community edition или 2019, са инсталираним Пајтон модулима или IDLE едитор кода.

Ова апликација омогућава истовремено рад са Пајтоном у интерактивном моду и у скрипт моду.

Пајтон интерактивни мод

Интерпретер омогућава да се у интерактивном моду унесе команде у једној линији у програмском језику Пајтон.

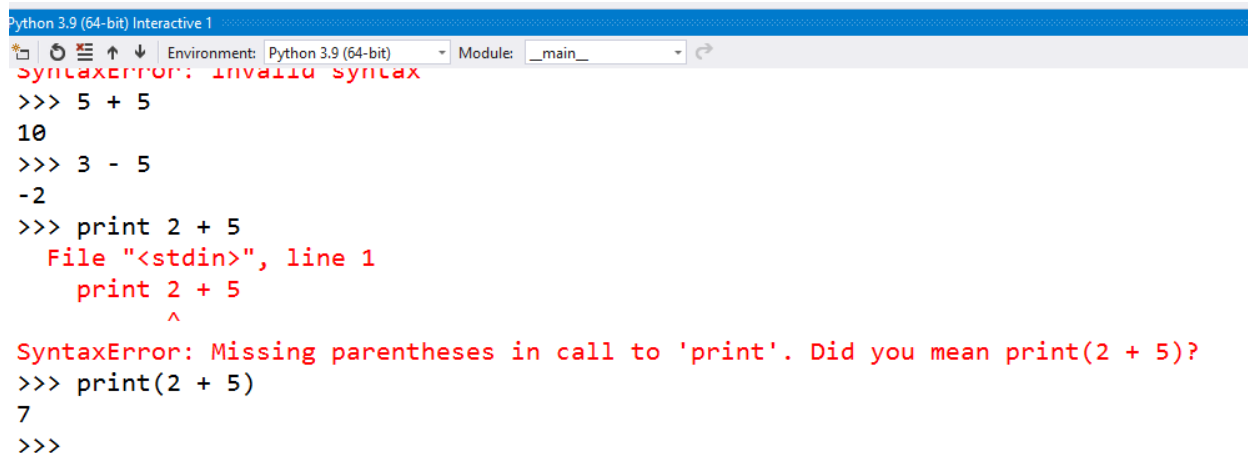
Прва слободна линија почиње са >>>.

Овај низ симбола се назива **промпт** (command prompt).

То је знак да је интерпретер спреман да прихвати укуцавање команде у новој линији.

По извршењу задате наредбе, поново се појављује >>>.

За овакав начин рада се каже да је секвенцијалан – појави се >>>, укуца се линија са командама, притисне се ЕНТЕР, добије се резултат, појави се >>> итд.



```
Python 3.9 (64-bit) Interactive 1
>>> 5 + 5
10
>>> 3 - 5
-2
>>> print 2 + 5
SyntaxError: invalid syntax
File "<stdin>", line 1
  print 2 + 5
    ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(2 + 5)?
>>> print(2 + 5)
7
>>>
```

020 Куцање броја у интерактивном моду

Откуцати следећи ред:

```
>>> 12345
```

Добија се резултат:

```
12345
>>> |
```

Из примера се види да куцање бројева и притисак на ЕНТЕР даје као резултат испис у новом реду укуцаног броја, прелазак у још један нови ред и постављање треперећег курсора на крају.

021 Куцање неслужбене речи у интерактивном моду

```
>>> hvala
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'hvala' is not defined
>>>
```

Из примера се види да куцањем непознатог низа знакова, који нису под наводницима или апострофом, изазива реакцију у виду поруке о грешци; Пајтон не препознаје укуцани низ знакова.

022 Куцање службене речи у интерактивном моду

```
>>> class
```

```
...
  File "<stdin>", line 1
    class
    ^
SyntaxError: invalid syntax
```

Сада је откуцана службена реч, коју интерпретер препознаје и која има посебно значење у Пајтон програмском језику.

Проблем је што она није коришћена на начин који синтакса Пајтон програмског језика захтева па је добијена порука о погрешној синтакси (граматичкој нетачности).

Питања и задаци за самосталан рад

Питања

1. Шта је то ASCII?
2. Објаснити дохвати-декодуј-изврши процес.
3. Шта су службене речи у програмским језицима?
4. Навести неколико службених речи из Пајтона.
5. У чему је разлика између компајлера и интерпретера?
6. Шта је Пајтон?
7. Која окружења за рад са Пајтоном постоје?
8. У којим модовима се може радити у Пајтону?
9. Описати интерактивни мод за рад у Пајтону.
10. Шта је то промпт?
11. Како реагује Пајтон ако познаје унуту вредност а како ако не познаје унуту вредност?

Задаци

013 Укуцати своје име и презиме у интерактивном моду и притиснути ЕНТЕР.

Објаснити шта се појавило на екрану.

014 а) Укуцати празно место па притиснути ЕНТЕР.

Поновити поступак два пута.

Шта се дешава на екрану интерактивног мода ?

б) Укуцати print и притиснути ЕНТЕР.

Укуцати input и притиснути ЕНТЕР.

Укуцати crazy и притиснути ЕНТЕР.

Објаснити шта се десило после сваког уноса и зашто су коментари различити.

015 Укуцати "tekst" и притиснути ЕНТЕР.

Укуцати 'tekst' и притиснути ЕНТЕР.

Укуцати tekst и притиснути ЕНТЕР.

Објаснити добијене излазе у интерактивном моду.

016 Укуцати 'tekst''tekst' и притиснути ЕНТЕР.

Укуцати "tekst""tekst" и притиснути ЕНТЕР.

Да ли постоје разлике у резултатима и зашто?

017 Укуцати 'tekst' , 10 празних места, 'tekst' и притиснути ЕНТЕР.

Како објашњавате добијени резултат?

Да ли интерпретер види празна места (белине) ?

Наредба излаза

Приказ излаза са функцијом print

Функција print се користи за приказ излаза Пajтон програма на екрану рачунара.

Функција је део претходно дефинисаног и написаног кода који изводи одређену операцију.

Пajтон има велики број предефинисаних функција које изводе различите операције.

Када се функција стартује, каже се да се функција позива.

Део кода функције print унутар заграде се назива аргумент функције.

А то је заправо податак који треба да се прикаже на екрану као резултат позива функције print.

Апострофи у примеру дефинишу почетак и крај текста који треба да се прикаже на екрану.

023 Испис откуцаног текста у интерактивном моду

```
>>> print('Zdravo, svete!')
```

```
Zdravo, svete!
```

```
>>>
```

024 Испис откуцаних линија текста у скрипт моду

```
print('Ovo je veoma')
```

```
print('veoma')
```

```
print('veoma')
```

```
print('kratak tekst.')
```


После куцања претходног кода, клик на опцију Debug па на Start Without debugging (Run->Run Module).

Тиме се стартује превођење целог откуцаног кода (програма) у машински језик и предаје се процесору на извршење.

Искази (statements) овог програма су се извршавали по редоследу по којем су се у куцали у коду, од прве до последње линије кода.

Основе рада са стринговима и стринг литералима

Подаци: 'Ovo je veoma', 'veoma', 'kratak tekst.' су низови карактера (знакова) и у програмирању низови знакова се називају стрингови.

Када се стринг појави као део кода, он се назива стринг литерал.

У Пајтону, стринг литерали се могу појавити између апострофа или између наводника.

Ако је потребно да стринг литерал садржи апостроф као део стринга, онда се стринг отвара и затвара са наводницима.

Ако је потребно да стринг литерал садржи наводнике као део стринга, онда се стринг отвара и затвара са апострофима.

Пајтон такође допушта да се стринг литерал садржи унутар троструких наводника или апострофа.

На овај начин могу се користити и наводници и апострофи у стрингу истовремено.

Такође, троструки наводници или апострофи се користе када је потребно приказати стринг литерал у више линија.

025 Приказ апострофа и наводника као део стринга

```
print('Rekao sam mu: "Nestle" je dobra cokolada.')  
print("A on meni: Znas, 'decko', u pravu si.")
```

Излаз:

Rekao sam mu: "Nestle" je dobra cokolada.

A on meni: Znas, 'decko', u pravu si.

У Пајтону се користе троструки знакови наводника или апострофа као посебан облик документовања кода (docstring) или за ограничавање стринга који може садржати и наводнике и апострофе истовремено.

026 Приказ апострофа и наводника истовремено као део стринга

```
print('Rekao sam mu: 'Decko', "Nesle" je dobra cokolada.')
```

Излаз:

Rekao sam mu: 'Decko', "Nesle" je dobra cokolada.

027 Приказ стринга у више линија на излазу

```
print("""Jedan  
Dva  
Tri""")
```

Излаз:

Jedan
Dva
Tri

Питања и задаци за самосталан рад

Питања

1. Чему служи функција?
2. Шта се дешава позивом функције?
3. Где се налази аргумент функције?
4. Чему служи аргумент функције print?
5. Шта је то стринг?
6. Шта је то стринг литерал?
7. Чему служи docstring?

Задаци

018 а) Отворити нови Пајтон пројекат и назвати га konan_inventar.

Како се зове фајл (сорс код)?

б) Додати следећу линију у скрипти: print("Imam sandale").

Шта је то print?

в) Кликнути на Debug-> Start Without Debugging.

Описати шта се десило.

г) У коду отворити нову линију кода пре већ попуњене и унети: print('Zovem se Konan.').

Шта ће бити сада написано у конзоли?

д) Има ли разлике између стрингова у коду? Зашто?

019 Следећа линија скрипте треба да изгледа овако:

Bravo Konan.

Написати код којим се то и приказује.

020 Који је најједноставнији начин да се испише празан ред у интерактивном моду?

Исписати једно празно место у интерактивном моду.

Да ли се може користити иста наредба као у претходном примеру?

Зашто?

021 Помоћу скрипт мода исписати своје име у једном реду а презиме у другом реду.

Помоћу скрипт мода исписати своје име и презиме у једном реду.

Извести закључак о потребном броју наредби излаза за оба случаја.

022 Да ли су по дефиницији стринга следећи примери стрингови: "а", " ", 'а', ' ', '' (наводници између апострофа)?

023 Шта ће се написати на екрану после реализације следећег кода у интерактивном моду:

```
print('Ja sam "dobar".')
```

Зашто се неће видети апострофи на екрану као резултат кода?

024 Шта се појављује после реализације следећег кода у скрипт моду:

```
print('Ja sam 'dobar'.')
```

Објаснити добијену поруку.

025 Написати код којим се твоје име појављује под апострофима у првом реду а твоје презиме под наводницима у другом реду на екрану.

026 Шта треба да се користи у коду да би се истом стрингу појавили и наводници и апострофи на екрану и зашто.

027 Коришћењем троструких наводника приказати следећи стринг:

```
nivo 1
nivo 0
nivo -1
```

028 Написати скрипт који даје на излазу следеће:

Bioskop	Film	Datum	Vreme
Dom sindikata	Ratovi zvezda	20.05.2018.	14:00
Sinapleks	Park iz doba jure	03.10.2018.	20:00

029 Коришћењем знака '@' нацртати квадрат, димензија 4x4.

030 У коду написати име, надимак и презиме најбољег друга и најбоље другарице у два реда. Надимке приказати под апострофима.

031 Нацртати једнакостранични троугао странице дужине 6, коришћењем знакова '.

Приказ података

Писање дугачких линија кода у више линија

Ако је исказ у једној линији дугачак, не може цео да се види у једном екрану корисника, па је корисно такав израз исписати у више линија кода, а да и даље задржи своју синтаксу и семантику. У Пајтону се то изводи коришћењем карактера за наставак линије (\) (line continuation character). Довољно је само откуцати \ и кликнути на Ентер и наставак исказа се може куцати у следећој линији.

```
print('Prodali smo', prodatih_jedinica, \
      'u sumi od', cena_prodatih)
result = var1 * 2 + var2 * 3 + \
        var3 * 4 + var4 * 5
```

Поништавање акције нове линије код функције print

Функција print приказује једну линију на излазу.

Исказ: `print("Jedan")` прво приказује стринг а затим активира карактер нове линије (newline character).

Овај карактер се не види али када се активира реализује прелазак у нову линију на излазу.

Ако није потребно да се после приказа стринга пређе у нову линију на излазу (поништавање акције нове линије) довољно је придодати посебан аргумент `end = ''`, функцији:

```
print('Jedan', end=' ')
```

```
print('Dva', end=' ')
```

```
print('Tri')
```

даје: Jedan Dva Tri и прелази се у нови ред.

Види се да после прва два стринга се исписује празно место.

Ако се не жели исписивање ни празног места између стрингова:

```
print('Jedan', end='')
```

```
print('Dva', end='')
```

```
print('Tri')
```

JedanDvaTri, и прелазак у следећи ред.

Аргументи функције print се називају keyword arguments.

Коришћење сепаратора ствари

Када се више аргумената придода функцији print, они су аутоматски одвојене празним местом када се прикажу на излазу:

```
print("Jedan", "Dva", "Tri")
```

даје: Jedan Dva Tri

Ако се не жели празно место између ствари, може се користити аргумент `sep = ''`:

```
print("Jedan", "Dva", "Tri", sep = '')
```

даје: JedanDvaTri

Може се између стрингова поставити и неки други карактер:

```
print("Jedan", "Dva", "Tri", sep = '*')
```

даје: Jedan*Dva*Tri

Ескејп карактери

Ескејп карактер је посебан карактер који се пише после \ и припада стринг литералу.

Они се сматрају посебним командама које су смештене унутар стринга.

<code>\n</code>	карактер нове линије	изазива да се на излазу пређе у нову линију
-----------------	----------------------	---

<code>\t</code>	табулар	изазива један таб прескок у истој линији
-----------------	---------	--

<code>\'</code>	апостроф	изазива штампање једног апострофа
-----------------	----------	-----------------------------------

<code>\"</code>	наводник	изазива штампање једног наводника
-----------------	----------	-----------------------------------

<code>\\</code>	бекслеш	изазива штампање бекслеш косе линије
-----------------	---------	--------------------------------------

Приказ више стринг литерала помоћу оператора +

Оператор + се иначе користи за сабирање два оператора.

Али, ако се оператор + напише између два стринга, активира се надовезивање стрингова (string concatenation).

```
print("Jedan " + "Dva " + "Tri.")
```

даје: Jedan Dva Tri.

Питања и задаци за самосталан рад

Задаци

032 Коришћењем карактера за наставак линије уредити следећу линију кода:

```
print("Ja sam pomislio da se ovde moze raditi i vikendom, ali sam pogresio. Moja kuca nikad nije  
bila veka.")
```

Линија кода треба да се уреди па да има максимално тридесет карактера у једној линији.

033 Поправити следећу линију кода тако да се донекле изједначи број карактера у свакој линији:

```
print("Ja sam\  
pomislio\  
da se ovde moze raditi i vikendom\  
ali sam pogresio. Moja kuca nikad nije bila\  
veca.")
```

034 У чему се разликују излази за линију: `print("Jedan")` и линију: `print("Jedan", end=' ')`

035 Шта ће бити излаз после:

```
print('Jedan', end='')  
print('Dva', end='')
```

036 У чему се разликују излази за линију: `print("Prvi", "Drugi", sep="")`

и линију: `print("Prvi", "Drugi", sep=' ')`

037 На које све начине је могуће кодовати овакав излаз из кода: `Ja***Ti***Mi`

038 Написати код који употребом једне функције `print` и `keyword arguments` на излазу изгледа овако:

`Plavo$Belo$Zuto$`

039 Објаснити шта се дешава на излазу линије:

```
print("Imam", "\tmalo vremena a", "\t\t\tmnogo prostora.")
```

040 а) Објаснити шта се дешава на излазу линије:

```
>>> print("Unesi kolicinu" + \  
        'kolaca koje si pojeo' +\  
        "\n")
```

'svakog dana.')

б) Исправи грешке које су се појавиле на излазу писањем другог скрипта са истим улазом

041 Коришћењем ескејп карактера у коду програмирати да излаз изгледа овако:

A B'C' "D" X

042 На излазу су све цифре а између цифара се налази **.

043 Коришћењем ескејп карактера написати скрипт који на излазу даје:

Ponedeljak	Utorak	Sreda
Cetvrtak	Petak	Subota
Nedelja		

044 Написати скрипт у три линије кода који на излазу изгледа:

ime@@ime ime@@ime @@ime@@

045 Креирати код који на излазу даје:

```
*****
**           **           **           **
**           **           **           **
*****
```

Променење

Коментари

Сваки коментар започиње симболом #, који се назива знаком бројева.

Било шта после тог знака до краја исте линије кода представља коментар.

Када напише компјутерски програм, своје идеје програмер преточи у велики број линија кода.

Програмерске идеје често нису очигледне било коме другоме сем самом аутору.

Дешава се да аутор кода после неког времена се врати написаном коду али ни њему самом више није лако да одреди смисао његових идеја.

Једини начин да се избегну овакви проблеми је писање коментара.

Коментари у коду помажу читању кода, олакшавају праћење развоја програмерских идеја и дају смисао и иначе тешко разумљивим линијама кода програмских језика.

028 Употреба коментара у коду

```
print('Naslov: Novi kod') #komentar zauzima deo linije koda
#komentar zauzima celu liniju bez koda
```

Излаз:

Naslov: Novi kod

Промењиве

Програми обично смештају податке у меморију рачунара и изводе операције над тим подацима. Програми користе промењиве за приступ и манипулацију подацима смештеним у меморији. Промењива је име које представља вредност у компјутерској меморији. Каже се да промењива упућује (reference) на вредност.

Идентификатор и оператор доделе

Исказ: `a = 23` значи да промењива **a** указује на меморијску локацију у којој се налази вредност **23**.

То је постигнуто помоћу оператора доделе (=, знак једнакости) (assignment operator).

Свака бројчана вредност која се користи у коду се назива нумерички литерал.

Промењиве се представљају помоћу идентификатора.

`a = 23, b = 11, ime = "MIKI"`

Промењива **a** указује на меморијску локацију у којој се налази вредност **23**, промењива **b** указује на локацију са вредности **11**, промењива **ime** указује на локацију са вредности **MIKI**.

Свака промењива може да промени вредност током рада програма.

Промењива садржи њој прво додељену вредност све док се промењивој не додели нека друга вредност.

Прво додељена вредност више није у употреби па Пајтон интерпретер аутоматски је брише из меморије преко процеса који се зове сакупљање ђубрета (*garbage collection*), чиме се ослобађа меморијска локација за даље коришћење.

Када се промењивој додељује вредност коју већ има нека друга промењива, интерпретер ће упутити да обе промењиве показују на исту меморијску локацију са жељеном вредности.

На овај начин се штеди употреба меморијских локација.

Провера указивања на исту локацију

Постоје два начина за проверу да ли промењиве указују на исту меморијску локацију:

Први је коришћењем инструкције `id(a)` унутар функције **print**, чиме се на екрану исписује адреса меморијске локације на коју упућује промењива **a**.

Други је коришћењем логичког оператора `is`.

Оператор упоређује адресе меморијских локација на које упућују промењиве **a** и **b** и ако су адресе идентичне, даје као резултат тачно (**True**) а у супротном нетачно (**False**).

029 Указивање на исту меморијску локацију

```
a = 5
```

```
print(id(a))    #140719639414560
```

```
b = 10
```

```
print(id(b))    #140719639414720
```

```
c = a
```

```
print(id(c))    #140719639414560
```

```
print(a is b)   # False
```

```
print(a is c)   # True
```

Правила давања имена промењивима (избор идентификатора)

- не може се користити службена реч
- у имену не може постојати празно место
- први карактер може бити било које велико или мало слово или доња линија (`_`) (*underscore*)
- следећи карактери у имену могу бити слова, цифре или доње линије
- постоји разлика између малих и великих слова (*ja* није иста промењива као *Ja* или *jA*)
- име промењиве би требало да има описно значење о сврси промењиве (*tezina_u_uncama*, *baza_podataka_ucenici*, *tabela_repertoar_bioskopa*)

Константе

Константе у програмирању су строго дефинисани идентификатори који се користе у кодовима као промењиве без могућности промене вредности за коју су везане.

У Пајтону не постоји строго дефинисани начин за употребу или синтаксу креирања константи. Постоји договор према којем сваки идентификатор који се пише великим словима представља неку константну вредност у датом коду.

Идеја је да програмер упозори на овај начин да не намерава да мења одређену вредност у коду.

030 Записивање константи у коду

ZNAK = "x"

REC = 'skola'

BROJ = 431

BROJ1 = 50.067

Питања и задаци за самосталан рад

Задаци

046 Нека се у скрипти налази следећи код:

```
print()
print("Moja omiljena knjiga je:")
print('')
print('"Gospodar prstenova"')
```

Убацити у сваку постојећу линију кода коментар

047 Следећи део кода се налази на самом почетку скрипте:

```
#+-----+
#| Naziv programa:   Interaktivni_potez      |
#| Autor:           Marko Markovic           |
#| Namena:           Igranje saha protiv kompjutera |
#| Vlasnik:          autor                     |
#| Datum izrade:     20.08.2020.              |
#+-----+
```

Која је намена датог коментара?

Зашто је стављен на почетак кода?

Коме је намењен?

Да ли аутор кода има корист од оваквог коментара и зашто?

048 Којим од следећих идентификатора су неправилно додељена имена и зашто:

units_per_day	print
dayOfWeek	Print
3dGraph	xxx
June1997	ja*ti
Mixture#3	danaz sutra

049 Без куцања кода одговорити шта ће се појавити на излазу:

```
a = 20
b = 40
c = 30
x, y, z = a, b, c
print(x, y, z)
```

050 Куцањем следећег кода проверити излаз:

```
a = 20
b = 40
c = 30
a = b = c
print(a, b, c)
```

објаснити резултат излаза.

051 Написати код којим се вредност 1 додељује константи BROJ, промењивој x додељује вредност -1 а затим се вредност на коју упућује константа додељује промењивој y.

На крају промењива x указује на вредност на коју упућује промењива y.

Приказати садржај свих употребљених промењивих и константи у задатку.

Да ли је правилно употребљена константа у овом примеру и зашто?

052 Без куцања кода одговорити шта ће се појавити на излазу:

```
a = 100
b = 200
b = a
a = 300
print(a, b)
```

Зашто промењива b на крају кода нема вредност 300?

Шта се десило са нумеричким литералом 200 на којег је указивала промењива b на почетку кода?

053 Шта се очекује на излазу после стартовања следећег кода:

```
a = 5
b = 5
c = 5
print(a is b)
print(a is c)
```

Објаснити добијени резултат на излазу.

054 Ако се после примене инструкције `id()` на две променљиве добија иста вредност, на који други начин се може објаснити садржај те две вредности?

055 Написати потпуни коментар на почетку кода којим се исписују лични подаци програмера.

056 Написати код којим се тренутни делови датума додељују трима променљивима.
Написати и одговарајуће коментаре на истим линијама кода.

057 Нека `x` и `y` добијају наизменично вредности од 1 до 6, тако што `x` добија само парне вредности а `y` само непарне вредности.
Написати скрипту која на излазу приказује како се `x` и `y` мењају.

058 Нека су почетни подаци за променљиву `a` највећи двоцифрени паран број а за променљиву `b` стринг `"x"`.
Разменити вредности променљивих `a` и `b` коришћењем помоћне променљиве тако да променљиве `a` и `b` увек указују на почетне, али међусобно замењене, податке.

Типови података

Нумерички типови података и литерали

Компјутери користе другачије технике и методе за смештање у меморију бројева са децималним зарезом од смештања целих бројева.

Такође, сличне операције над целим и реалним бројевима се изводе на различите начине.

Из тог разлога у Пајтону постоје различити типови података којима се различито вреднују подаци смештени у меморији.

Када се цео број смести у меморију, он се класификује као **int**.

Када се реалан број смести у меморију, он се класификује као **float**.

Израз `soba = 503`, смешта вредност 503 у меморију и променљива `soba` упућује на њу.

Израз `dinari = 15.55` смешта вредност 15.55 у меморију и променљива `dinari` упућује на њу.

Када се број напише унутар програмског кода, он се назива **нумерички литерал**.

Када интерпретер прочита нумерички литерал у коду, он одређује његов тип податка према следећим правилима:

Ако је нумерички литерал написан као цео број без децималне тачке сматра се да је типа `int` (7, 124, -20); ако је нумерички литерал написан са децималном тачком, сматра се да је типа `float` (1.5, 5.0203, -0.34).

Одређивање типа података

Уграђена функција `type` се користи за одређивање типа података дате вредности:

```
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>> type("recenica")
<class 'str'>
```

У свим примерима вредност у загради се придодаје као аргумент функцији `type`.

Види се да је 1 број типа `int`, 1.0 број типа `float`, "recenica" је низ знакова типа `string (str)`.

Давање промењивој вредност другачијег типа података

Током писања програма, Пајтон интерпретер води рачуна о именима промењивих и вредностима на које они упућују.

Промењива у Пајтону може у сваком тренутку да упућује на вредност било који типа података.

Једном додељена вредност промењивој се у току извршења програма може у било ком моменту променити; чак може да се додели и вредност другог типа.

```
>>> x = 5
>>> print(x)
5
>>> x = "Ovo je lep kod"
>>> print(x)
Ovo je lep kod
```

Конверзија типова података

У Пајтону, иако промењива може једноставно да указује на податак било којег типа, понекад је неопходно конвертовати (променити) податак из једног у други тип података.

Постоји два начина на који се то ради у Пајтону:

а) Експлицитна конверзија, када се изричито тражи да се конвертује податак из једног у други тип:

```
a = 10          #promenjiva a ukazuje na podatak 10 koji je tipa int
b = float(a)    #promenjiva b ukazuje na podatak koji je konvertovan podatak promenjive a u realan
                #broj, b=10.0
c = str(a)      #promenjiva c ukazuje na podatak koji je konvertovan podatak promenjive a u string,
                #c="10"
d = int(c)      #promenjiva d ukazuje na podatak koji je konvertovan podatak promenjive c u ceo broj,
                #d=10
```

Ако се експлицитна конверзија не може извести, јавиће се грешка у програму.

б) Имплицитна конверзија, када се подразумева конверзија података која се извршава по аутоматизму

```
z = 5           #z je ceo broj
x = 10          #x je ceo broj
y = 50.5        #y je realan broj
z = x + y       #z ce biti automatski konvertovan u realan broj da bi rezultat bio tacan, z = 60.5
```

Питања и задаци за самосталан рад

Задаци

059 Којег је типа податак за сваку од вредности:

"jedan"	-0.5	0
1	-5	0.000
1.0	"minus pet"	"nula"

060 Зашто су 5 и 5.0 различити типови података иако представљају исту вредност.

061 Низ бројева распореди према вредности од највећег према најмањем и одредити којем типу података припада: 0.5, -10, -1.36, 2.4567, 12, 2.45, -1.35

062 Код изгледа овако:

```
a = "rec"
b = 5
print(a,b)
```

Како се тачно називају обе вредности које се користе у овом коду?

063 а) Зашто је неопходно тачно знати за сваку вредност којег је типа као податак?

б) Шта ако нисмо сигурни којег је типа податак, како да будемо сигурни?

064 Коришћењем функције type, одредити којег су типа следеће вредности:

5.03, 'godina', 2018, "moja skola", -29, -0.234

065 Без куцања у коду размислити и дати одговор написмено: шта ће интерпретер да прикаже као излаз за случај кода type("slovo") а затим проверити у скрипт моду

066 Зашто су излази следећих кодова исти у интерактивном моду а различити у скрипт моду:

print(type(1)) и type(1)

067 Објаснити излазе следећег кода у интерактивном моду:

```
>>> x = 2.05
>>> print(type(x))
<class 'float'>
>>> x = "slovo"
>>> print(type(x))
<class 'str'>
```

068 Да ли ће следећи код произвести грешку и зашто:

```
a = 5.02
```

```
b = a
```

```
a = "skola"
```

069 У задатку 068, додати нове линије кода којима се доказују различити типови података и различите меморијске локације на које упућују промењиве.

070 Шта је резултат следећег кода:

```
x = 1
```

```
y = float(x)
```

```
z = str(y)
```

```
print(x, y, z)
```

Објаснити резултат на екрану.

071 Написати код којим се делови датума твог рођења уносе у три промењиве па се вредност дана конвертује у стринг а вредност године конвертује у реалан број.

072 Написати код којим се доказује да није могуће извршити конверзију из стринг типа података у целобројни тип података.

073 Да ли је могуће баш увек извршити било какву конверзију било којег типа података у било који други тип података?
Зашто?

074 Која врста грешке се добија ако се покуша извршити недозвољена конверзија?

075 Објаснити резултат кода:

```
x = 1
```

```
c = float(str(x))
```

```
print(c)
```

Објаснити како следећи код даје идентичан резултат као претходни:

```
print(float(str(1)))
```

076 Написати скрипт којим се вредност 5 предаје промењивој x на три различита начина као три различита типа података и на излазу приказати који су типови у питању.

077 Промењиве x и y указују на вредности различитих типова (било којих).

Написати скрипту којом се размењују почетне вредности ових промењивих а на излазу се доказује да је дошло и до размене типова података

Наредба улаза

Читање улаза са тастатуре

Програми траже унос података да би се извеле операције над њима.

Када се податак унесе у програм са тастатуре, најчешће се смешта у променљиву да би могао касније да се користи у програму.

У Пајтону постоји функција **input** која чита улазне податке са тастатуре и враћа их у програм као стрингове.

Општи формат функције input:

```
promenjiva = input(string)
```

где је string текст који ће се приказати на екрану; смисао овог текста је да објасни кориснику шта се тражи од њега, какву вредност да унесе.

Док је promenjiva име променљиве која упућује на податак који је унет са тастатуре.

```
name = input('Kako se zoves?')
```

Када се ова линија кода изврши, дешава се следеће: приказује се стринг Kako se zoves? на екрану; програм се зауставља и чека да корисник унесе нешто преко тастатуре и затим притисне ЕНТЕР; када се притисне ЕНТЕР, податак који је укуцан се предаје као стринг и додељује променљивој name.

031 Употреба наредбе улаза

```
>>> a = input('Kako se zoves?')
```

```
Kako se zoves?
```

```
Hulk
```

```
>>> print(a)
```

```
Hulk
```

```
>>>
```

После исписивања Kako se zoves? програм чека да се унесе податак са тастатуре.

Оно што се укуца (Hulk) се додељује променљивој a.

032 Додела вредности променљивој наредбом улаза

```
ime = input("Unesi ime: ")
```

```
prezime = input("Unesi prezime: ")
```

```
print('Zdravo', ime, prezime)
```

```
Unesi ime: Hogar
```

```
Unesi prezime: Strasni
```

```
Zdravo Hogar Strasni
```

Унутар стринга који је аргумент функције input, намерно је као последњи знак постављено празно место.

Разлог је тај пошто функција input не приказује аутоматски празно место после текста.

Читање бројева са функцијом input

Функција input увек враћа унету вредност као стринг, чак и ако је унета вредност број.

Проблем је што се математичке операције могу вршити само са бројчаним вредностима.

Зато се користе две предефинисане функције које конвертују стринг у нумерички тип података: `int(vrednost)`, `float(vrednost)`, чиме се вредност претвара у тип `int` или у тип `float`.

033 Конверзија унетог податка у целобројну вредност

```
godina_string = input("Unesi tvoj broj godina: ")
godina_int = int(godina_string)
print("Imam", godina_int)
print(type(godina_int))
```

Unesi tvoj broj godina: 30

Imam 30

<class 'int'>

034 Конверзија стринга у цео број нестовањем позива функције `input`

```
godina_int = int(input("Unesi tvoj broj godina: "))
print("Imam", godina_int)
```

Unesi tvoj broj godina: 35

Imam 35

Прва линија у примеру користи нестовање позива функције.

Вредност која се враћа из функције `input` се додаје као аргумент функцији `int()`.

У случају да је унета вредност неправилног бројчаног типа података, јавиће се грешка позната као **изузетак** (exception).

Изузетак је грешка која се јавља током извршења програма, чиме се прекида извршење програма.

Питања и задаци за самосталан рад

Задаци

078 У чему је функционална разлика између следећих линија кода :

```
ime = input("Unesi svoje ime")
ime = input("Kako se zoves?")
ime = input("")
ime = input()
```

079 Која је функција текста као аргумента функције `input`?

080 Зашто није добро да корисник уноси податке у програм на следећи начин:

```
input("a sada najbolja stvar")
```

081 Шта ће бити излаз следећег кода:

```
>>> x = input("ime?")
ime?
```

Betmen

```
>>> type(x)
```

- 082 а) Коју вредност ће имати променљива број на излазу: `broj = input("Unesi lep broj?")`
б) Којег ће типа података бити та вредност?
в) Да ли добијена вредност има смисла према коду који је унешен?
г) Како се назива процес промене типа података за одређену вредност у програмирању?
- 083 Какви се типови података добијају извршавањем следећих линија кода:
а) `float(input())`
б) `str(input())`
в) `int(input())`
г) `input()`
- 084 Написати скрипт којим се уносе твоје име и презиме а на излазу се исписује поздрав теби.
- 085 Написати скрипт којим се уносе са тастатуре твоје име и презиме, број година и висина у метрима.
- 086 За двојицу запослених се уносе плате у облику реалних бројева.
Написати скрипту којом се преко конверзије улазних података плате приказују у целобројном облику.
- 087 Написати скрипт којим се уносе са тастатуре називи две боје а на излазу се добија у једној линији комбинација та два назива без празног места између а у другој линији се између назива додаје карактер -.

Аритметички оператори

Математичке операције

Да би се извеле математичке операције у Пајтону, користе се математички оператори:

оператор	операција	опис
+	сабирање	сабира два броја (2 + 3, даје 5)
-	одузимање	одузима два броја (2 – 3, даје -1)
*	множење	множи два броја (2 * 3, даје 6)
/	дељење	дели два броја и резултат је број типа float (2 / 3, даје 0.66666)
//	целобројно дељење	дели два броја и резултат је број типа int (5 // 3, даје 1)
%	остатак дељења	после дељења два броја враћа остатак дељења (5 % 3, даје 2)
**	степеновање	диже први број на степен другог броја (2 ** 3, даје 8)

Вредности које се налазе са леве и десне стране оператора се називају операнди.

Променљиве се могу користити у математичким изразима (expression):

broj_minuta_ukupno = sati * 60

Код целобројног дељења, ако је резултат позитиван, врши се одсецање тј. део после децималног зареза се одсеца.

Код целобројног дељења, ако је резултат негативан, врши се заокруживање тј. заокружује се на најближи цео број.

-5 // 2, даје -3

Приоритет оператора

Ако се математички изрази пишу са већим бројем оператора, њихово извршавање зависи од приоритета оператора.

Следећа правила се користе:

- ако постоје заграде, прво се рачуна резултат у најужој загради
- по приоритету оператори су:
**
* / // %
+ -
- ако су оператори истих приоритета, прво извршава се оператор са леве стране према десној

035 Приоритети аритметичких операција

izlaz = 12.0 + 6.0 / 3.0

print(izlaz)

Добија се 14.0

Прво се изврши 6.0 / 3.0 пошто / има виши приоритет у односу на +; даје 2.0; затим се рачуна 12.0 + 2.0.

Једини изузетак је у случају вишеструког степеновања, где се оператори извршавају са десна на лево:

3 ** 2 ** 2 се рачуна: 3 ** (2 ** 2)

Изрази са више типова података

Када се изводи математичка операција са два операнда, тип резултата операције ће зависити од типова операнда.

Следећа правила постоје при евалуацији математичких израза:

- када се изводи операција са два int операнда, резултат ће бити типа int
- када се изводи операција са две float операнда, резултат ће бити типа float
- када се изводи операндија са int операндом и float операндом, int вредност ће привремено бити конвертована у float и резултат ће бити типа float

036 Конверзија типова код израза са више типова

rezultat = 5 * 2.0

print (rezultat)

Даје 10.0; пошто се операнд (вредност) 5 прво конвертује у 5.0 па се изводи операција множења као да су оба операнда типа float.

Питања и задаци за самосталан рад

Задаци

088 Навести свих седам основних математичких оператора у Пајтону и које операције симболизују.

089 Сви математички оператори су бинарни а који функционишу и као унарни?

090 Следећи израз написати у коду и израчунати: $8.4 \cdot 5.05 \cdot \frac{11}{3.45}$

091 а) Шта се очекује на излазу после сабирања једног целог и једног реалног броја?

б) А после одузимања?

в) Да ли тип резултата утиче избор математичке операције између операнда различитих типова?

г) Како гласи правило аутоматске конверзије при раду са int и float операндима.

092 а) Написати програм који подржава формулу за рачунање површине круга:

$$povrs_kruga = \pi r^2$$

б) Да ли је вредност π промењива у овом задатку и како се она у структури програма назива и обележава?

в) Модификовати написани код тако да се обезбеђује унос искључиво реалних вредности за полупречник круга.

093 Написати програм који за унету страницу квадрата рачуна обим и површину квадрата.

094 а) Написати програм који рачуна средњу вредност од унета три цела броја.

б) Конвертовати добијени резултат у целобројну вредност и у стринг.

в) Модификовати програм тако да рачуна грешку због конверзије у целобројну вредност у процентима.

095 Израчунати колико кругова полупречника 5 центиметара може да стане у правоугаоник димензија 30 центиметара и 50 центиметара.

096 Написати скрипту која вредност унешену од корисника диже на квадрат и на куб и то приказује на излазу.

097 За две унешене реалне вредности израчунати збир, разлику, производ и количник.

098 Колико је 30% од унете вредности?

099 Приказати шта се дешава са међурезултатима при рачунању израза $25 + 3.05 - 15$

0100 Под условом да не постоје застоји у путу, пређени пут који ауто прелази се добија према формули: $distanca = brzina * vreme$. Нека се ауто креће брзином од 80 километара на час. Написати скрипт који приказује:

- Пређени пут аута после 6 сати
- Пређени пут аута после 10 сати
- Пређени пут аута после 15 сати

0101 Написати скрипт који рачуна колики ће бити бакшиш појединачног келнера у ресторану. У ресторану раде 4 келнера. Договор је да на једнаке делове поделе сав бакшиш после радног дана. Улазни подаци су укупни трошкови набавке продате хране (troskovi), укупна цена продате хране (сена), укупан износ новца који су муштерије платиле тога дана (novac). Водити рачуна да ресторан захтева своју зараду која не сме бити мања од разлике : $troskovi - сена$. Оно што остаје је зарада келнера (baksis).

Програмирање математичких израза

Манипулација цифрама

Сваки вишецифарски број се може разложити на појединачне цифре коришћењем математичких операција над тим бројем.

037 Цифре вишецифарског целог броја

```
broj = int(input("Uneti pozitivan cetvorocifreni ceo broj: "))
cifra_jedinice = broj % 10
cifra_desetice = (broj // 10) % 10
cifra_stotine = (broj // 100) % 10
cifra_hiljade = (broj // 1000) % 10
print("Pojedinačne cifre broja", broj, "su: ", \
      cifra_jedinice, cifra_desetice, cifra_stotine, cifra_hiljade)
```

Uneti pozitivan cetvorocifreni ceo broj: 4321

Pojedinačne cifre broja 4321 su: 1 2 3 4

038 Цифре вишецифарског реалног броја

```
broj = float(input("Uneti pozitivan realan broj sa dve cifre u celom delu i \
                   dve cifre u decimalnom delu: "))
cifra_jedinice = int(broj % 10)
cifra_desetice = (int(broj // 10)) % 10
cifra_prve_decimale = (int(broj * 10)) % 10
cifra_druge_decimale = (int(broj * 100)) % 10
print("Pojedinačne cifre broja", broj, "su: ", \
      cifra_jedinice, cifra_desetice, cifra_prve_decimale, \
      cifra_druge_decimale)
```

Uneti pozitivan realan broj sa dve cifre u celom delu i dve cifre u decimalnom delu: 43.78

Pojedinačne cifre broja 43.78 su: 3 4 7 8

Програмирање математичких израза

Да би се математички написан израз могао израчунати писањем кода у Пајтону, потребно је правилно користити приоритете операција и жељене типове података као резултате тих операција.

039 Програмирање математичких израза: $a + \frac{(b-a)^{4.5}}{2.5\sqrt{\frac{b}{a}}}$

```
a = float(input("Unesi vrednost za a: "))
b = float(input("Unesi vrednost za b: "))
izraz1 = (b - a) ** 4.5
izraz2 = (b / a) ** (1 / 2.5)
print(a + (izraz1 / izraz2))
```

Unesi vrednost za a: 3.5

Unesi vrednost za b: 2.1

(3.5000000000000003+5.575908782090215j)

Питања и задаци за самосталан рад

Задаци

0102 Написати програм који од унетих цифара x, y, z креира троцифрени број хуz.

0103 Написати програм који приказује збир цифара унетог двоцифреног броја.

0104 Написати програм који приказује цифру стотина у унетом четвороцифреном броју.

0105 Компанија плаћа порез на годишњем нивоу од 23% на годишњи приход. Ако се унесе годишњи приход, написати скрипт за рачунање годишње зараде компаније.

0106 Написати програм који доказује тачност формула:

- $(x - y)(x + y) = x^2 - y^2$
- $(ab)^n = a^n b^n$
- $(a + b)^2 = a^2 + 2ab + b^2$

0107 За дате вредности отпорности отпорника R_1 , R_2 и R_3 израчунати еквивалентну отпорност паралелне везе прва два отпорника и са њима редно везану трећу отпорност и укупну јачину струје кроз цео систем ако је напон на првом отпорнику $U_1 = 10V$.

0108 Израчунати $\frac{(x)^5}{\sqrt{\frac{1}{y}}} - \frac{(y + 3.5)^{1.5}}{2.4\sqrt{x}}$

Форматирање бројева

Приказ бројева

Понекад је потребно другачије форматирати приказ бројева на екрану, што се посебно односи на децималне бројеве.

Када се приказује број са зарезом преко функције `print`, он се може појавити коришћењем до 12 значајних цифара.

040 Приказ реалног броја без додатног форматирања

```
ukupno_zaduzenje = 5000.0
mesecna_rata = ukupno_zaduzenje / 12.0
print('Mesecna rata je', mesecna_rata)
даје: Mesecna rata je 416.666666667
```

Предефинисана функција `format`

Коришћењем предефинисане функције `format` могуће је форматирати изглед бројева на излазу. Функција `format` при позиву придодаје два аргумента функцији: бројну вредност и спецификатор формата.

Спецификатор формата је стринг који садржи специјалне знаке који указују како се жели форматирање бројне вредности.

```
format(12345.6789, '.2f')
```

У примеру, први аргумент је `float` број 12345.6789 који се жели форматирати.

Други аргумент је стринг `'.2f'` а то је спецификатор формата:

- `.2` значи прецизност, указује да се жели заокруживање броја на два децимална места (два места после тачке)
- Слово `f` указује је тип броја који се жели форматирати, `float`

Функција `format` враћа стринг са форматираним бројем.

```
print(format(12345.6789, '.2f'))
```

даје: 12345.68

Форматирање у научној нотацији

Други начин представљања реалних бројева је у научној (scientific) нотацији:

```
print(format(12345.6789, 'e'))
```

1.234568e+04

```
print(format(12345.6789, '.2e'))
```

1.23e+04

Слово `e` означава експонент (може бити и `E`)

Употреба зарез сепаратора

```
print(format(12345.6789, ',.2f'))
```

12,345.68

```
print(format(123456789.456, ',.2f'))
```

123,456,789.46

```
print(format(12345.6789, 'f'))  
12,345.678900
```

У последњем примеру се користи зарез сепаратор али се не наводи податак о прецизности форматирања.

041 Зарез сепаратор

```
mesecna_rata = 5000.0  
ukupno_zaduzenje = mesecna_rata * 12  
print('Tvoje ukupno zaduzenje je $', format(ukupno_zaduzenje, ',.2f'), sep='')  
Tvoje ukupno zaduzenje je $60,000.00
```

Минимална ширина поља

Минимална ширина поља је број места који се користе за приказ вредности.

```
print('Broj je', format(12345.6789, '12,.2f'))
```

У примеру, 12 значи да је минимална ширина поља за приказ бројева 12.

У примеру је 12,345.68 са мање места (има 9 места).

У оваквим случајевима, број се поставља скроз десно у пољу.

Ако је вредност превелика да би се сместила у одређену ширину поља, поље се аутоматски повећава да би се прилагодило.

```
print('Broj je', format(12345.6789, '12,.2f'))  
Broj je      12,345.68
```

Уместо коришћења f, користи се % за форматирање као проценат:

```
print(format(0.5, '%'))  
50.000000%  
print(format(0.5, '.0%'))  
50%
```

Форматирање целих бројева

При форматирању целих бројева користи се обележавач d и не може се специфицирати прецизност:

```
print(format(123456, 'd')) #bez formatiranja  
123456  
print(format(123456, ',d')) #sa zarez separatorom  
123,456  
print(format(123456, '10d')) #polje sirine 10  
123456  
print(format(123456, '10,d')) #sa zarez separatorom i u polju sirine 10  
123,456
```

Питања и задаци за самосталан рад

Задаци

0109 Објаснити који су аргументи функције `format` у примеру: `format(356.9001, '9,.3f')`

0110 Чему служи зарез сепаратор у форматирању бројева?

0111 Форматирати број 5000200 коришћењем функције `format` тако да на излазу изгледа: 5,000,200 и да буде у пољу ширине 12 уз десну ивицу поља.

0112 Написати скрипт који рачуна проценат дечака и девојчица у одељењу са 11 дечака и 16 девојчица.

0113 Нека је рецепт за прављење 48 колача: 1.5 шоља шећера, 1 шоља маслаца, 2.75 шоља брашна. Написати програм који тражи од корисника колико укупно колача треба да се направи. На излазу приказује колико шоља шећера, маслаца и брашна је потребно за прављење жељеног броја колача. Форматирати излазе на две децимале и поља ширине 7.

Релациони оператори

Булови изрази и релациони оператори

Изрази који се као постављени услови испитују да ли су тачни или не, називају се Булови изрази. Релациони оператори служе за формирање Булових изрази.

Релациони оператор одлучује да ли постоји некаква релација између две вредности (операнда). Релациони оператори су `>` (веће од), `<` (мање од), `>=` (веће или једнако), `<=` (мање или једнако), `==` (еквивалентан са), `!=` (није еквивалентан са).

042 Рад са Буловим изразима

```
>>> x = 1
>>> y = 0
>>> x > y
True
>>> y > x
False
```

Да би се могло спровести испитивање тачности Буловог изрази, сваки од операнда мора имати некаку вредност и мора постојати некакав релациони оператор између њих.

Оператори `==` и `!=`

Оператор `==` испитује да ли је операнд са леве стране идентичан са операндом са десне стране. Ако су вредности на које се односе оба операнда исте, израз је тачан.

Оператор `!=` испитује да ли је операнд са леве стране различит од операнда са десне стране. Ако су њихове вредности различите, израз је тачан.

043 Рад са операторима == и !=

```
>>> x = 1
>>> y = 0
>>> z = 1
>>> x == y
False
>>> x != y
True
>>> x == z
True
```

Питања и задаци за самосталан рад

Задаци

0114 Ако су $a = 1$, $b = 2$, $c = 2.5$, испитати следеће релације:

- a) **a** веће од **b**
- b) **b** мање или једнако **c**
- c) **c** идентично са 2.5
- d) **c** веће или једнако од **a** плус **b**
- e) **b** идентично са конвертовано **c** у целобројно

0115 Шта је резултат: $0 \neq \text{int}(0.1)$

0116 Објаснити како се реализује следећа линија кода:

```
a = 5 <= 5
```

0117 a) Шта је резултат следећег кода?

```
a = 5 < 5
b = 3.0 == 3
print(b != a)
```

б) Колико у првој линији кода постоји оператора а колико операнда?

в) Да ли се измењена прва линија кода у примеру $a < 5 = 5$ уопште може реализовати и зашто?

0118 Шта ће бити резултати следећих Булових израза:

```
True > True
False == True
True > False
False >= True
```

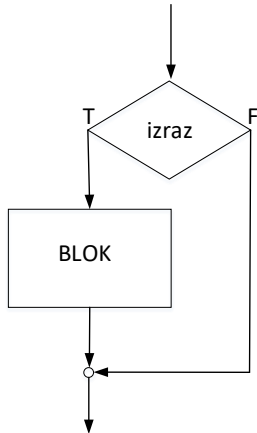
Исказ if

Условне структуре

Секвенциона структура је низ исказа који се извршавају према редоследу по којем су написани. Контролна структура је логички дизајн који контролише редослед према којем се искази извршавају.

Условне структуре или структуре одлучивања омогућавају извршавање низа исказа само под одређеним условима.

Исказ if



На слици је структура која се назива **једносмерна селекција** (single alternative decision structure). Ако је израз испуњен (тачан) почеће извршавање путање кода симболично означен као T (true). На тој путањи се налази елемент обраде података (блок) који ће се извршити само у случају испуњености услова.

Ако израз није испуњен (нетачан) почеће извршавање путање симболично назване F (false).

На тој путањи не постоји неједан елемент обраде података па се директно долази до скретнице у дијаграму тока.

Општи формат if исказа:

If uslov:

 blok

Сам израз представља испитивање испуњености некаквог услова.

Услов може бити испуњен или неиспуњен; ако је испуњен извршиће се блок наредби.

044 На екрану исписати поруку ако је унети број већи од 0.

```
broj = int(input("Unesi ceo broj: "))
if broj > 0:
    print("Broj", broj, "jeste veci od 0.")
```

Unesi ceo broj: 50

Broj 50 jeste veci od 0.

Unesi ceo broj: -50

Press any key to continue . . . █

Unesi ceo broj: 0

Press any key to continue . . . █

045 На екрану исписати поруку ако унети број није једнак 10.

```
broj = int(input("Unesi ceo broj: "))
if broj != 10:
    print("Broj", broj, "nije jednak 10.")
```

046 Повећање плате под условом продаје

Ако је продавац у продавници рачунара продао, за текући месец, робу у вредности од 2.000.000,00 динара (и више), његова плата се повећава за 2% и добија бонус од 5.000,00 динара. Колико ће зарадити тај продавац на крају месеца ако је његова месечна плата 50.000,00 динара?

```
plata = 50000.00
povecanje = 0.02          #to je 2% u realnim brojevima
bonus = 5000.00
novac = float(input("Uneti novac dobijen od prodaje robe: "))
if novac >= 2000000.00:
    plata += (plata * povecanje) + bonus

print("Prodavac je zaradio za ovaj mesec ", plata, "dinara.")
```

047 Испис поруке под условом бодова

Наставник током године даје три теста из предмета програмирање. Написати програм који сваки ученик може користити да израчуна средњи број бодова са тестова. Такође, ако је средњи број бодова на тестовима једнак или већи од 95, написати посебну поруку таквом ученику.

```
odlican_rezultat = 95
test1 = int(input('Uneti bodove sa 1.testa: '))
test2 = int(input('Uneti bodove sa 2.testa: '))
test3 = int(input('Uneti bodove sa 3.testa: '))

#racuna srednji broj bodova sa tri testa
srednje = (test1 + test2 + test3) / 3
print('Srednja vrednost bodova je', srednje)
if srednje >= odlican_rezultat:
    print('Cestitamo!')
    print('Ovo je odlicna srednja vrednost!')
```

048 Испис вредности са једносмерном селекцијом

Написати програм који за било које унете вредности променљиве x рачуна променљиву y као:

$$y = \begin{cases} x, & x < 0 \end{cases}$$

```
y = 0
print("Pocetna vrednost za y je: ", y)
x = int(input('Uneti vrednost za x: '))
if x < 0:
    y = x

print("y ima vrednost: ", y)
```

Питања и задаци за самосталан рад

Задаци

0119 Ако се зна да вредности a и b нису исте, написати код за откривање која је већа.

0120 Ако су a и b бројеви различитих типова, испитати њихову идентичност.

0121 Продавац има циљ да заради месечно 100.000,00 динара. Ако у прва три месеца продаје није зарадио потребну своту, добија отказ. Написати програм који води рачуна о овом проблему.

0122 Ако је y једнако 20, доделити променљивој x вредност 0.

0123 Корисник уноси боју по жељи. Написати код који испитује да ли је унета боја бела и на излазу даје одговарајуће поруке.

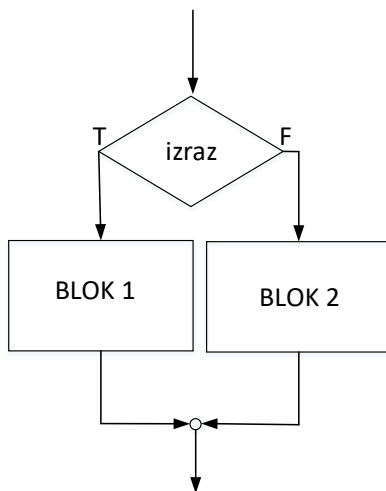
0124 Корисник уноси боју по жељи. Претпоставља се да може унети белу, црвену или плаву боју. За сваку од унетих боја на излазу дати одговарајуће поруке.

0125 Ученик уноси, у виду реалног броја са два децимална места, број који представља његов просек на крају школске године. Написати програм који на екрану исписује његов успех у текстуалном облику (одличан,...).

Исказ if-else

if-else исказ

Секвенциона структура је низ исказа који се извршавају према редоследу по којем су написани.



Ако је Булов израз тачан, извршавање кода ће кренути делом кода симболично обележеним са T и извршиће се блок наредби блок1.

Ако Булов израз није тачан, извршавање кода ће кренути делом кода симболично обележеним са F и извршиће се блок наредби блок2.

Општи формат if-else исказа:

If uslov:

 blok 1

else:

 blok2

Услов се испитује само једном и тада се одлучује који део кода ће се извршити.

Ако је услов испуњен блок1 се извршава.

Ако услов није испуњен извршиће се блок 2 који се налази у телу else исказа.

По извршењу блок 1 или блок 2 обраде података извршава се код испод if-else исказа.

Увлачење линија кода код if-else исказа

Код креирања if-else исказа битно је уредити линије кода које припадају телу if исказа и else исказа.

То уређивање се врши увлачењем (indentation) линија кода.

Пајтон не може функционисати ако одређене линије кода нису правилно увучене.

Зато је неопходно проверити да ли су if и else искази у истој линији а затим осигурати да блокови исказа који припадају if телу и else телу буду увучени у односу на if и else исказе.

if uslov:

 blok1

else:

 blok2

049 Додела вредности преко испуњеног услова

Промењива x је 10. Ако је x веће од унетог y, y постаје x иначе y постаје негативна вредност од x.

```
x = 10
```

```
y = int(input('Uneti vrednost za y: '))
```

```
if x > y:
```

```
    y = x
```

```
else:
```

```
    y = -x
```

```
print("y je sada", y)
```

050 Провера тренутне температуре

```
temperatura = int(input("Unesi trenutnu temperaturu vazduha: "))
```

```
if temperatura > 30:
```

```
    print("Bas je vruce.")
```

```
else:
```

```
    print("Nije toliko vruce.")
```

```
Unesi trenutnu temperaturu vazduha: 25
```

```
Nije toliko vruce.
```

```
Unesi trenutnu temperaturu vazduha: 35
```

```
Bas je vruce.
```

—

051 Рачунање зараде у односу на радне сате

Ако радник у ауто-сервису ради преко 40 сати недељно, биће плаћен 1.5 пута више од по сату за сваки сат преко 40 сати рада недељно. Написати програм који рачуна плату радника у ауто-сервису.

Псеудокод:

унети број сати рада

унети цену рада по сату

ако радник ради више од 40 сати:

израчунати и приказати зараду прековременог рада

у супротном:

израчунати и приказати уобичајену зараду

```
radni_sati_nedeljno = 40
faktor = 1.5
sati = float(input("Uneti sati rada ove nedelje: "))
cena_rada = float(input("Uneti cenu rada po satu: "))
if sati > radni_sati_nedeljno:
    sati_preko_norme = sati - radni_sati_nedeljno
    placanje_preko_norme = sati_preko_norme * cena_rada * faktor
    pare = radni_sati_nedeljno * cena_rada + placanje_preko_norme
else:
    pare = sati * cena_rada
print('Radnik je za ovu nedelju zaradio u $', format(pare, ',.2f'), sep='')
```

```
Uneti sati rada ove nedelje: 40
Uneti cenu rada po satu: 20
Radnik je za ovu nedelju zaradio u $800.00
Uneti sati rada ove nedelje: 50
Uneti cenu rada po satu: 20
Radnik je za ovu nedelju zaradio u $1,100.00
```

052 Вредност у опсегу са две if-else петље

Написати програм који утврђује да ли је унета вредност у опсегу од 0 до 10, коришћењем две if-else структуре.

```
x = int(input('Uneti vrednost za x: '))
if x >= 0:
    print("Uneti broj jeste veci ili jednak sa 0.")
else:
    print("Uneti broj je manji od 0.")
if x <= 10:
    print("Uneti broj jeste manji ili jednak sa 10.")
else:
    print("Uneti broj je veci od 10.")
```

053 Испис вредности са двосмерном селекцијом

Написати програм који за било које унету вредност промењиве x рачуна промењиву y као:

$$y = \begin{cases} x ** 3, & x > 5 \\ x ** 2, & x \leq 5 \end{cases}$$

```
x = int(input('Uneti vrednost za x: '))
if x > 5:
    y = x ** 3
else:
    y = x ** 2

print("y ima vrednost: ", y)
```

Питања и задаци за самосталан рад

Задаци

0126 Аутомобил шаље поруку возачу да је прекорачио дозвољену брзину од 80 km/h или да је није прекорачио.

0127 Написати програм који за унети број између 1 и 7 враћа назив дана у недељи.

0128 За унете димензије за два правоугаоника дати поруку који правоугаоник има већу површину.

0129 Корисник уноси реалан број. Ако је број са нулама као децималама, приказати га као цео број, иначе као реалан какав је и унешен.

0130 Написати код који допушта или не допушта приступ подацима у зависности од унешене лозинке.

0131 Коришћењем два if-else исказа, креирати код који испитује која нова гума је постављена на бициклу.

0132 Корисник уноси дан, месец и годину у облику позитивних целих бројева. Ако било који од бројева није могућ дати поруку о томе (предвидети да дан није већи од 30, месец није већи од 12 а унета година није већа од 3000).

Упоредивање стрингова

Тестирање вредности стрингова

Секвенциона структура је низ исказа који се извршавају према редоследу по којем су написани. Пајтон омогућава упоређивање стрингова.

Могуће је направити структуру упоређивања која тестира вредности стрингова.

054 Упоредивање идентичности стрингова

```

ime1 = 'Milan'
ime2 = 'Ana'
if ime1 == ime2:
    print('Imena su ista.')
else:
    print('Imena nisu ista.')
Imena nisu ista.

```

Оператор == упоређује ime1 и ime2 да би одредио да ли су идентични.

Пошто њихове вредности, стрингови 'Milan' и 'Ana', нису исте, приказаће се порука на екрану 'Imena nisu ista.'.

Приликом куцања словних вредности са тастатуре треба водити рачуна да Пајтон разликује велика и мала слова и да вредности стрингова зависе од тога.

Друге врсте упоређивања стрингова

Осим што је могуће упоредити стрингове према идентичности, такође се стрингови могу упоредити према величини.

Ово је корисно када је потребно сортирати стрингове према уређеном редоследу.

ASCII код (American Standard Code for Information Interchange) служи да представи слова у бројчаном облику у рачунарској меморији:

- Велика слова од А до Z су бројеви између 65 и 90
- Мала слова од а до z су бројеви између 97 и 122
- Када се цифре од 0 до 9 смештене као карактери у меморији, оне су представљене бројевима од 48 до 57 (нпр стринг 'abc123' ће бити смештен у меморији као кодови 97, 98, 99, 49, 50, 51)
- Празно место (белина) се представља бројем 32

Поред тога, ASCII код успоставља редослед карактера ('А' је пре 'В' које је пре 'С' итд)

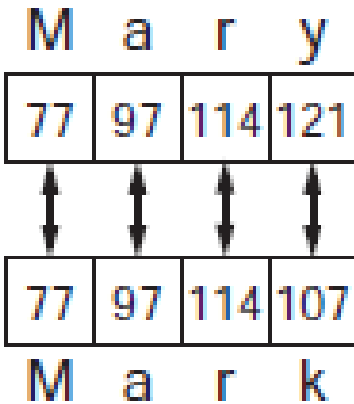
Када програм упоређује карактере, он заправо упоређује кодове тих карактера:

```
if 'a' < 'b':
```

```
    print('The letter a is less than the letter b.')
```

Овај пример упоређује ASCII кодове за карактер а и карактер b. Пошто је код за 'а' једнак 97, а код за 'b' једнак 98, израз if 'a' < 'b' ће бити True и извршиће се блок кода са исписивањем поруке.

Ако се упоређују стрингови са више карактера они се упоређују карактер са карактером на истој позицији у стринговима.



Ово је редослед упоређивања карактера у овом случају:

1. Упоређује се прво слово првог стринга са првим словом другог стринга: овде су оба слова иста 'М' (њихови ASCII кодови имају исту вредност), па се прелази на упоређивање следећих карактера
2. Упоређује се друго слово првог стринга са другим словом другог стринга: овде су оба слова иста 'а', па се прелази на упоређивање следећих карактера
3. Упоређује се треће слово првог стринга са трећим словом другог стринга: овде су оба слова иста 'р', па се прелази на упоређивање следећих карактера
4. Упоређује се четврто слово првог стринга са четвртим словом другог стринга: у првом стрингу то је слово 'у' са кодом 121, у другом стрингу то је слово 'к' са кодом 107
5. Ако се тражи који од ова два стринга је већи, пошто је код за 'у' > 'к', онда је први стринг већи од другог стринга

У случају да се упоређују стрингови различитог броја карактера, могу да се упоређују само карактери на одговарајућим позицијама.

Ако су одговарајући карактери идентични, онда се краћи стринг сматра мањим ('А' < 'АВ').

055 Упоређивање стрингова различитих дужина

```
ime1 = 'OK'
ime2 = 'Ana'
if ime1 <= ime2:
    print('String', ime1, "је manji ili jednak od stringa", ime2)
else:
    print('String', ime2, "је manji od stringa", ime1)
```

String Ana је manji od stringa OK

Питања и задаци за самосталан рад

Питања

1. Шта је то ASCII код и чему служи?
2. Ко је већи карактер х или w? Објаснити резултат.
3. Шта је мање "Dan" или "dan"?

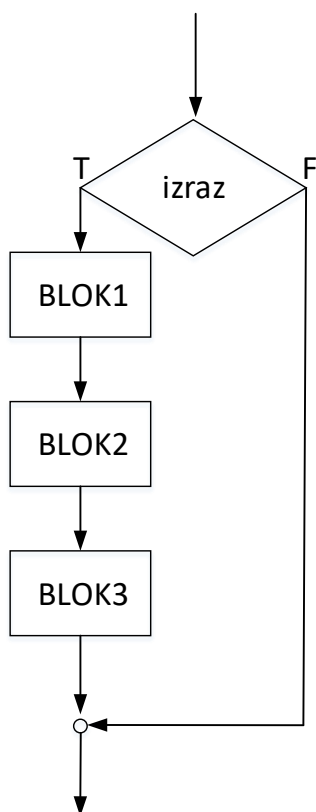
Задаци

0133 Написати код који допушта или не допушта приступ подацима у зависности од унешене лозинке.

0134 Написати код којим корисник уноси два стринга па на екрану исписује стрингове по алфаветском редоследу.

Угнеждане условне структуре

Тестирање више од једног услова



Да би се тестирало више од једног услова, условне структуре се могу угнездити унутар друге условне структуре.

Понекад се друге врсте структура угнезде унутар условне структуре.

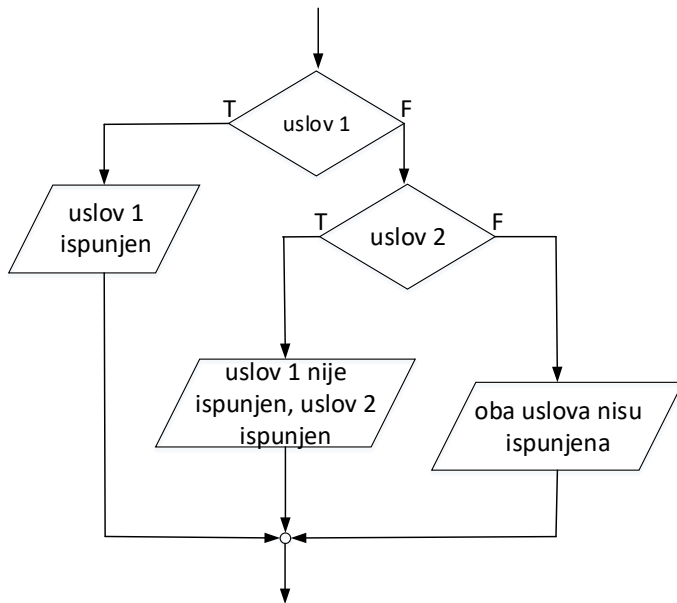
На слици је дијаграм тока који приказује условну структуру са низом секвенцијалних структура које су угнеждане (нестоване) унутар условне структуре.

Условна структура испитује услов изван ових секвенцијалних структура.

Ако је услов тачан, извршаваће се блокови секвенцијалних структура угнежедених унутар условне структуре.

У случају да услов није тачан, угнеждане структуре се неће извршавати.

Још чешће се дешава да су условне структуре унутар других условних структура:



Види се да постоје три путање извршавања кода.

Прва путања ће бити реализована ако је услов1 испуњен, у том случају није битно да ли је услов2 испуњен.

Друга путања ће бити реализована ако услов1 није испуњен а услов2 јесте.

Трећа путања ће бити реализована ако услов1 није испуњен и услов2 није испуњен.

056 Испитивање услова за позајмицу

#Program odredjuje da li musterija banke moze dobiti pozajmicu.

min_plata = 300000.0 #minimalna godisnja plata

min_godina = 2 #minimum godina zaposlen

#musterijina godisnja plata

plata = float(input('Uneti godisnju zaradu: '))

#godine zaposlenja na trenutnom poslu

godine_na_poslu = int(input('Uneti godine na trenutnom poslu: '))

#odredjivanje da li je musterija kvalifikovan za pozajmicu

if plata >= min_plata:

 if godine_na_poslu >= min_godina:

 print('Kvalifikovan si za pozajmicu.')

 else:

 print('Moras biti zaposlen najmanje', min_godina, 'godina.')

else:

 print('Moras zaradjivati najmanje ', format(min_plata, ',.2f'), \

 ' dinara godisnje da bi se kvalifikovao.', sep='')

Uneti godisnju zaradu: 350000.00

Uneti godine na trenutnom poslu: 1

Moras biti zaposlen najmanje 2 godina.

```

Uneti godisnju zaradu: 250000.00
Uneti godine na trenutnom poslu: 5
Moras zaradjivati najmanje 300,000.00 dinara godisnje da bi se kvalifikovao.
Uneti godisnju zaradu: 350000.00
Uneti godine na trenutnom poslu: 5
Kvalifikovan si za pozajmicu.

```

Питања и задаци за самосталан рад

Задаци

0135 Ако је унети број мањи од 50, испитати да ли је мањи и од 20. За сваки случај обезбедити одговарајућу поруку.

0136 Ако је време сунчано и ако је нема ветра и ако је температура већа од 30 степени, идемо на излет. У било којем другом случају, излет се неће одржати. Написати програм који шаље одговарајућу поруку која зависи од услова за одлазак на излет.

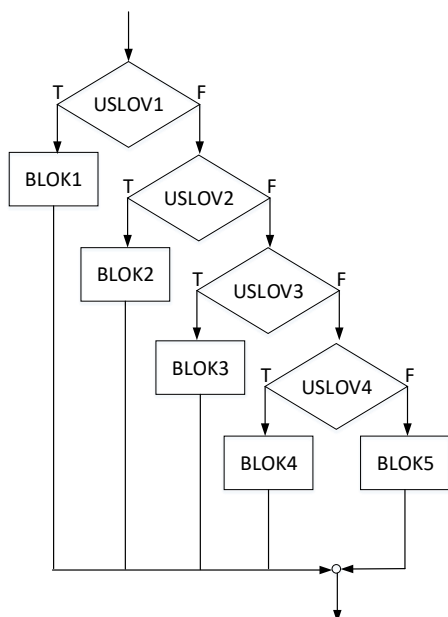
0137 За приступ бази података тражи се двостепена лозинка. Први степен лозинке је корисничко име (user name) и представља један стринг. Други степен је целобројна шифра (password number). Приступ бази података се неће дозволити и ако корисничко име није правилно унешено. За тестирање унети неколико сетова правилних лозинки.

Исказ if-elif-else

if-elif-else исказ

Логика која се примењује у угнежденим условним структурама може бити изузетно комплексна у реалним случајевима.

Из тог разлога, Пајтон омогућава специјалну верзију условне структуре која се назива if-elif-else исказ и која олакшава писање овакве структуре.



Општи формат if-elif-else исказа:

```
If uslov1:
    blok1
elif uslov2:
    blok2
elif uslov3:
    blok3
...
else:
    blok4
```

Прво се испитује услов1.

Ако је тачан, прекида се даље испитивање осталих услова.

Ако услов1 није тачан, испитује се следећи а то је услов2.

Ако је услов2 тачан, прекида се даље испитивање осталих услова, итд.

Процес се наставља све док се не пронађе услов који је тачан или више не постоји elif исказа за проверу.

Ако ниједан услов није тачан извршава се блок који припада else исказу.

Сваки нестована условна структура се може заменити са одговарајућим if-elif-else исказом.

057 Конверзија бодова са теста у описну оцену

```
A_skor = 90
B_skor = 80
C_skor = 70
D_skor = 60
```

```
skor = int(input('Uneti tvoj skor na testu:'))
#odredjivanje opisne ocene
if skor >= A_skor:
    print('Tvoja ocena je A.')
elif skor >= B_skor:
    print('Tvoja ocena je B.')
elif skor >= C_skor:
    print('Tvoja ocena je C.')
elif skor >= D_skor:
    print('Tvoja ocena je D.')
else:
    print('Tvoja ocena je F.')
```

058 Провера вредности броја у датом опсегу вредности

```
donja_granica_opsega = int(input("Uneti donju granicu opsega: "))
gornja_granica_opsega = int(input("Uneti gornju granicu opsega: "))
broj = int(input("Uneti trazeni broj: "))
if broj < donja_granica_opsega:
    poruka = "Broj je ispod datog opsega."
elif broj > gornja_granica_opsega:
```

```

    poruka = "Broj je iznad datog opsega."
else:
    poruka = "Broj je unutar datog opsega."

```

```

print(poruka)

```

059 Регулација тренутног нивоа воде

```

nivo_granica = 10
razlika_u_nivoima = 0
nivo_trenutni = int(input("Uneti trenutni nivo tecnosti u sudu: "))
if nivo_trenutni < nivo_granica:
    razlika_u_nivoima = nivo_granica - nivo_trenutni
elif nivo_trenutni > nivo_granica:
    razlika_u_nivoima = nivo_trenutni - nivo_granica
else:
    razlika_u_nivoima = 0

print("Regulisem za", razlika_u_nivoima, "nivoa tecnosti.")

```

Питања и задаци за самосталан рад

Задаци

0138 Аутомобил будућности без људског возача је стао на семафору. Написати програм који задаје команду мотору аутомобила у зависности од тренутне боје на семафору.

0139 У посластичарници се може купити сладолед са једним од четири укуса: јагода, малина, чоколада и ванила. Написати програм који прима поруџбину од купца о сладоледу одређеног укуса и потврду да је сладолед тог укуса испоручен купцу.

0140 Човек се може наћи у једном од три расположења: добро, задовољавајуће или лоше. Написати код који утврђује тренутно расположење корисника.

0141 Написати код који претвара унети назив месеца у облику стринга у редни број месеца у години.

0142 Написати програм у облику квиза који тражи одговоре на три питања а за свако питање има понуђен неколико одговора. Бодовима се регулише тачност датих одговора. На излазу приказати добији број бодова и одговарајућу поруку за испитаника.

0143 Постоје три могућа стања у једном дану: добро (g), задовољавајуће (s) и лоше (b). Ако се посматра сет од три узастопна дана, у том сету нема узастопног понављања истог стања дан за даном (нпр, не може се наћи ggs или sbb). Ако се посматрају сетови који нису идентичног садржаја, колико различитих сетова се може узастопно појавити под датим правилима?

0144 Задата су два опсега целих бројева. Који су све могући случајеви преклапања вредности из ова два опсега?

Логички оператори

Логички оператори у Пајтону

Пајтон омогућава коришћење сета логичких оператора за креирање комплексних Булових израза. Логички оператори у програмском језику Пајтон су `and`, `or` и `not`.

Оператор `and`

Овај оператор узима два Булова израза као операнде и креира сложен Булов израз који је тачан само ако су оба Булова израза тачна: `if a < 10 and b > 5: print()`, што значи да само ако је `a < 10` и `b > 5` извршиће се тело `if` исказа.

Израз	вредност израза
T and F	F
F and T	F
F and F	F
T and T	T

Оператор `or`

Овај оператор узима два Булова израза као операнде и креира сложен Булов израз који је тачан када је макар један од Булових израза тачан: `if a < 10 or b > 5: print()`, што значи да ако је или `a < 10` или `b > 5` извршиће се тело `if` исказа.

Израз	вредност израза
T or F	T
F or T	T
F or F	F
T or T	T

Оператор `not`

Овај оператор је унарни оператор који узима Булов израз као операнд и обрће његову логичку вредност:

If `not(a > 10): print()`, што значи да ако `a` није веће од 10 извршиће се тело `if` исказа.

Израз	вредност израза
not T	F
not F	T

Процена кратког споја (short-circuit evaluation)

Оператори `and` и `or` извршавају процену кратког споја.

Код `and` оператора: ако израз са леве стране `and` оператора је нетачан, израз са десне стране неће бити ни провераван.

Пошто ће сложени израз бити нетачан ако је макар један од Булових израза нетачан, процесор неће губити време у проверавању десне стране израза.

Код од оператора: ако је израз са леве стране од оператора тачан, израз са десне стране неће бити ни провераван.

Пошто ће сложени израз бити тачан ако је макар један од Булових израза тачан, процесор неће губити време у проверавању десне стране израза.

Провера опсега бројева

Ако је потребно проверити да ли се задати број налази у задатом опсегу вредности најбоље је употребити and оператор:

if x >= 20 and x <=40:

print('Vrednost je u zeljenom opsegu')

Ако се жели проверити да ли је задати број изван датог опсега вредности, најбоље је користити or оператор:

if x < 20 or x > 40:

print('Vrednost je izvan zeljenog opsega')

060 Строжи услови за позајмицу новца

```
min_plata = 300000.0
```

```
min_godina = 2
```

```
plata = float(input('Uneti godisnju zaradu: '))
```

```
godine_na_poslu = int(input('Uneti godine na trenutnom poslu: '))
```

```
if plata >= min_plata and godine_na_poslu >= min_godina:
```

```
    print('Kvalifikovan si za pozajmicu.')
```

```
elif plata >= min_plata and godine_na_poslu < min_godina:
```

```
    print('Moras biti zaposlen najmanje', min_godina, 'godina.')
```

```
elif plata < min_plata and godine_na_poslu >= min_godina:
```

```
    print('Moras zaradjivati najmanje ', format(min_plata, ',.2f'), \
          ' dinara godisnje da bi se kvalifikovao.', sep='')
```

```
else:
```

```
    print("Uneti podaci nisu korektni.")
```

061 Блажи услови за позајмицу новца

```
min_plata = 300000.0
```

```
min_godina = 2
```

```
plata = float(input('Uneti godisnju zaradu: '))
```

```
godine_na_poslu = int(input('Uneti godine na trenutnom poslu: '))
```

```
if plata >= min_plata or godine_na_poslu >= min_godina:
```

```
    print('Kvalifikovan si za pozajmicu.')
```

```
elif plata < min_plata and godine_na_poslu < min_godina:
```

```
    print('Moras biti zaposlen najmanje ', min_godina, ' godine ili ' \
          'moras zaradjivati najmanje ', format(min_plata, ',.2f'), \
          ' dinara godisnje da bi se kvalifikovao.', sep='')
```

```
else:
```

```
    print("Uneti podaci nisu korektni.")
```

Питања и задаци за самосталан рад

Задаци

0145 Нека су вредности промењивих $a = 2$, $b = 4$, $c = 6$. Који су резултати следећих услова:

```
a == 4 or b > 2
6 <= c and a > 3
1 != b and c != 3
a >= -1 or a <= b
not (a > 2)
```

0146 Написати програм који тражи унос броја година особе. Програм приказује поруку у зависности од тога да ли су унете године за бебу, дете, тинејџера или одраслог према следећем упутству:

Ако особа са 1 годином или мање, особа је беба.

Ако је особа старија од 1 године али млађа од 13 година, особа је дете.

Ако особа има најмање 13 година, али и мање од 18 година, особа је тинејџер.

Ако је особа стара најмање 20 година, особа је одрасла.

0147 Објаснити шта је резултат кода и зашто:

```
a = 5
b = 10
if b == 0 and x == 5:
    pass
else:
    print("OK")
```

0148 Објаснити шта је резултат кода и зашто:

```
a = 5
b = 10
if a == 5 or x == 5:
    print("OK")
else:
    pass
```

0149 Ако је време сунчано и ако је нема ветра и ако је температура већа од 30 степени, идемо на излет. Или ако је температура већа од 30 степени или је време сунчано идемо на излет. У било којем другом случају, излет се неће одржати. Написати програм који шаље одговарајућу поруку која зависи од услова за одлазак на излет.

0150 Написати код који претвара унети цео позитиван број у опсегу од 1 до 10 и претвара га у одговарајући Римски број.

0151 За унешени број секунди на екрану исписати колико је то минута, сати и дана у целим

бројевима (нпр, 33568 секунди даје 0 dana 9 sati 19 minuta 28 sekundi).

0152 Написати код који за унети месец исписује које је годишње доба тога месеца.

Искази услова

Булове промењиве

Булове промењиве могу имати једно од две вредности: True или False.

Булове промењиве су промењиве типа bool.

Булове промењиве се најчешће користе као флегови.

Флегови су промењиве које сигнализирају када неки услови постоје у програму.

Када је флег промењива постављена на False, то указује да услов није испуњен.

Када је вредност флег промењиве постављена на True, то указује да је услов испуњен.

062 Употреба флегова

```
flag = False
broj = int(input())
if broj == 10:
    flag = True
else:
    flag = False
if flag == True:
    print("Unet je broj 10")
else:
    print("Unet je broj koji nije jednak 10")
```

Види се да промењива flag се користи као флег јер сигнализира да ли је испуњен услов broj == 10.

Део кода:

```
if flag == True:
    print("Unet je broj 10")
```

се може и другачије (ефикасније) написати:

```
if flag:
    print("Unet je broj 10")
```

063 Конверзија нестоване петље у if-elif-else исказ

```
broj = int(input("Uneti neki broj: "))
if broj == 1:
    print('Jedan')
else:
    if broj == 2:
        print('Dva')
    else:
        if broj == 3:
            print('Tri')
        else:
            print('Nepoznato')
```

Резултат:

```
broj = int(input("Uneti neki broj: "))
if broj == 1:
    print('Jedan')
elif broj == 2:
    print('Dva')
elif broj == 3:
    print('Tri')
else:
    print('Nepoznato')
```

064 Магичан датум

Када се помножи дан са месецом а добију се две последње цифре за годину (нпр, 6 октобар 1960 је 6.10.60.) каже се да је тај датум магичан. Ако корисник уноси дан, месец и годину као три позитивна броја, на екрану дати поруку да је датум магичан, иначе да није магичан.

```
dan = int(input("Uneti dan kao celi broj: "))
mesec = int(input("Uneti mesec kao ceo broj: "))
godina = int(input("Uneti dve poslednje cifre godine: "))
if (godina == dan * mesec):
    poruka = "Datum jeste magican."
else:
    poruka = "Datum nije magican."
```

```
print(poruka)
```

065 Математички искази са условима

Сабрати два двоцифрена унета броја ако је први број дељив са 3 и 7 али није дељив са 5 и други ако није дељив са 4 а јесте са 9.

```
broj1 = int(input("Uneti prvi docifreni broj: "))
broj2 = int(input("Uneti drugi dvocifreni broj: "))
zbir = 0
poruka1 = "Zbir nije dobijen!"
poruka2 = "Zbir je dobijen!"
if broj1 % 3 == 0 and broj1 % 7 == 0 and broj1 % 5 != 0:
    if broj2 % 4 != 4 and broj2 % 9 == 0:
        zbir = broj1 + broj2
        print(poruka2)
        print("Zbir je", zbir)
    else:
        print(poruka1)
else:
    print(poruka1)
```

Питања и задаци за самосталан рад

Задаци

0153 Преко једначине: $tezina = masa * 9.8$, добија се тежина објекта у њутнима ако је маса објекта дата у килограмима. Написати програм који тражи од корисника унос масе објекта а затим израчунава тежину тог објекта. Ако је тежина објекта већа од 500 њутна, дати поруку да је објекат претежак. Ако је објекат тежине мање од 100 њутна, дати поруку да је објекат прелеган. За преостале тежине објекат је добре тежине

0154 На точку рулета лежишта су нумерисана од 0 до 36. Боје су:

- 0 је зелена
- Од 1 до 10, непарни су црвени, парни су црни
- Од 11 до 18, непарни су црни, парни су црвени
- Од 19 до 28, непарни су црвени, парни су црни
- Од 29 до 36, непарни су црни, парни су црвени

Написати програм где корисник уноси број лежишта на точку рулета и приказује одговарајућу боју. Програм даје и поруку о грешци ако корисник унесе број који не постоји за лежиште рулета.

0155 Објаснити резултат следећег кода:

```
if True:
    print("Dobro")
else:
    print("Nije dobro")
```

0156 Објаснити резултат следећег кода:

```
if False:
    print("Nije dobro")
else:
    print("Dobro")
```

0157 Ученик на крају школске године може да буде позитиван по успеху и владању (оцене из свих предмета и владања су веће од 1), може да буде позитиван по успеху али не и по владању (оцене из предмета су позитивне али је недовољан из владања). У оба случаја може да му се израчуна средња оцена (оцене из свих предмета и оцена из владања). Такође може да буде недовољан (макар једна оцена 1 из неког од предмета) и може да буде неоцењен (макар из једног предмета нема оцену). У овим случајевима се не рачуна средња оцена. Написати програм који даје као излаз успех ученика заснован на оценама (одличан, врло добар, добар, довољан, недовољан или неоцењен) као и средњу оцену ако је ученик позитиван по успеху.

0158 Датум се уноси употребом два целобројна позитивна броја као месец и година. Када се на тај начин унесу два датума израчунати колико је месеци разлике између њих. Месеци се обележавају од 1 до 12.

0159 Путовање са континента на континент се може извести аутомобилом (ако је веза између континента копнена) или бродом (ако је веза између континента водена). Путовање аутомобилом је кошта 500 а путовање бродом кошта 1000. Први унети континет је стартна позиција а други унети континент је циљ пута. Написати програм путовања са било којег континента на било који континент а да се потроши најмања могућа свота новца.

0160 Написати програм који даје лекарску терапију лечења пацијента засновану на три описа стања пацијента: температура, бол, исцрпљеност.

Температура преко 39 – антибиотик.

Исцрпљеност, температура 36, нема бол – мировање.

Бол, температура преко 38, нема исцрпљеност – облоге.

Бол или исцрпљеност, температура преко 36 – специјалистички преглед.

while петља

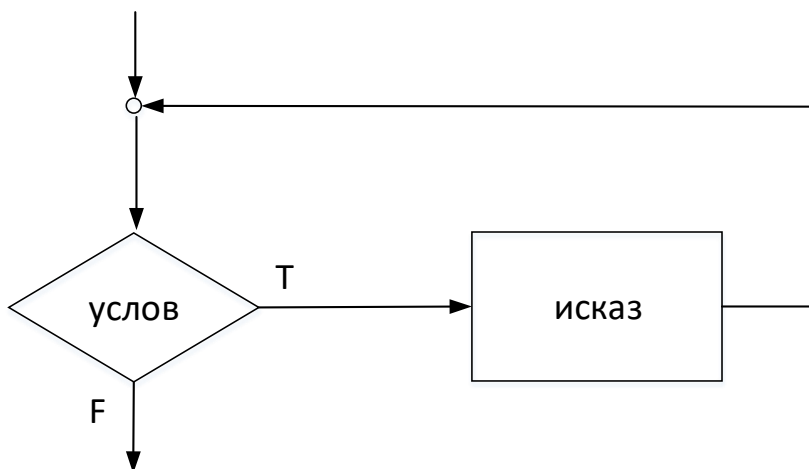
Петље у програмирању

Уместо да се пише исти код потребан број пута, логичније је написати део кода једном па програмирати да се понавља потребан број пута.

Ово се назива програмерска петља или структура понављања.

У Пајтону постоје две врсте петље: while петља (петља контролисана условом) и for петља (петља контролисана бројачем).

While петља



Петље контролисане условом се понављају док је услов испуњен.

Назив while петља потиче од објашњења на енглеском језику који приказује како ради петља: while a condition is true, do some task (док је услов испуњен, ради исти задатак).

While петља има два дела: услов који се тестира и низ исказа (тело петље) који се понављају све док је услов тачан.

Из дијаграма се види да се после извршења исказа ток кода враћа на поновно испитивање услова петље.

Општи формат while петље:

```
while uslov:  
    iskaz
```

066 Рад while петље

```
lep_dan = 'da'  
while lep_dan == 'da':  
    lep_dan = input("Da li je danas lep dan? ")  
print("Vise nije.")  
Da li je danas lep dan? da  
Da li je danas lep dan? da  
Da li je danas lep dan? da  
Da li je danas lep dan? ne  
Vise nije.
```

Из кода се види да линија `while lep_dan == 'da':` представља заглавље (хедер) петље while, где је само `lep_dan == 'da'` услов петље који се проверава.

Ако је услов испуњен, извршиће се `lep_dan = input("Da li je danas lep dan? ")` које је тело петље (исказ) па се поново услов проверава.

Када услов више није испуњен, извршава се `print("Vise nije.")` исказ који није део while петље.

Види се да промењива `lep_dan` има иницијалну вредност "да".

То је зато што ако не би имала баш ту вредност као иницијалну, програм никад не би испунио услов за улазак у петљу.

Једно извршење тела петље се назива **циклус** или **итерација**.

While петља је врста петље која се назива и петља са излазом на врху, а то значи да се увек прво провери тачност услова па се тек онда извршава тело петље.

Такође, пошто се не зна унапред у ком тренутку ће корисник унети другачију вредност за промењиву `lep_dan`, ова врста петље се назива и кориснички контролисана петља.

067 Корисничка контрола петље

```
print("Potrebno je voditi racuna o temperaturi u peci.")  
print("Temperatura ne sme pasti ispod 50 celzijusovih stepeni.")  
print("Ako se to desi odmah se mora povecati temperatura u peci.")  
temperatura = float(input("Kolika je temperatura u peci? "))  
while temperatura >= 50:  
    print("Temperatura je pogodna za odrzavanje procesa.")  
    temperatura = float(input("Kolika je temperatura u peci? "))  
print("Povecavam temperaturu!")
```

Potrebno je voditi racuna o temperaturi u peci.
Temperatura ne sme pasti ispod 50 celzijusovih stepeni.
Ako se to desi odmah se mora povecati temperatura u peci.
Kolika je temperatura u peci? 100
Temperatura je u pogodna za odrzavanje procesa.
Kolika je temperatura u peci? 120
Temperatura je u pogodna za odrzavanje procesa.
Kolika je temperatura u peci? 40
Povecavam temperaturu!

Питања и задаци за самосталан рад

Задаци

- 0161 Написати програм који исписује на екрану твоје име све док као одговор на питање „Ко си ти?“ се укуцава „JA“.
- 0162 Помоћу while петље написати код који сабира вредност у промењивој број са 1 све док корисник не унесе као одговор на питање “ne”.
- 0163 Додавати слово по слово у промењиву гес све док корисник не унесе слово “x”.
- 0164 Сабирати све унете бројеве од стране корисника коришћењем петље while, све док је укупан збир тих бројева мањи од 10.
- 0165 Корисник треба да уноси људска имена једно по једно све док не унесе име Јован, када се while петља прекида.

for петља

Петља контролисана бројачем

Као и while петља, петља for је петља са излазом на врху, али основна разлика је у томе што је ово петља контролисана бројачем.

For петља се извршава унапред задатим бројем пута.

Општи формат for петље:

```
for promenjiva in [vrednost1, vrednost2,...]:  
    iskaz
```

овде је promenjiva име промењиве а у правоуглој загради је низ вредности.

У Пајтону низ вредности у загради одвојени зарезом се назива листа.

Исказ чини тело петље.

Прво се промењивој додељује прва вредност из листе а затим се извршава исказ у телу петље.

Затим се промењивој додељује друга вредност из листе и поново се изврши исказ из тела петље итд.

Петља се завршава када је извршен исказ из тела петље са последњом вредности у листи.

068 Приказ бројева из уређене листе

```
for broj in [1, 2, 3]:  
    print(broj)
```

Приказаће се:

```
1  
2  
3
```

069 Приказ бројева из неуређене листе

```
for broj in [10, 5, -3]:  
    print(broj)
```

Приказаће се:

```
10  
5  
-3
```

У првој итерацији, променљива `broj` има вредност 1 и извршава се функција `print` која штампа ту вредност.

Затим се променљивој `broj` додели следећа вредност из листе (2) коју функција `print` штампа, као и за 3.

Каже се да је петља итерирала 3 пута.

У Пајтону се променљива у петљи `for` назива и циљна променљива (*target variable*) пошто је она циљ доделе на почетку сваке итерације.

070 Петља `for` са стринговима у листи

```
for ime in ['Ana', 'Jovana', 'Milanka']:  
    print(ime)
```

```
Ana  
Jovana  
Milanka
```

Употреба функције `range` са `for` петљом

Пајтон има уграђену функцију `range` која поједностављује процес писања `for` петље.

Функција `range` креира тип објекта познат као итерабла (*iterable*).

Итерабла је објекат који садржи низ вредности које итерирају преко нечега налик на петљу.

071 Употреба функције `range`

```
for broj in range(3):  
    print(broj)
```

Уместо листе вредности, преко функције `range` се даје број 3 као аргумент.

То значи да ће се генерисати итерабилна секвенца целих бројева у опсегу од 0 (укључујући и 0) до 3 (не укључујући и 3).

072 Замена функције range() листом

```
for broj in [0, 1, 2]:  
    print(broj)
```

Пошто се листа састоји од три вредности, петља ће итерирати три пута.

073 Исписивање стринга одређени број пута

```
for x in range(3):  
    print('Zdravo svete')
```

Zdravo svete

Zdravo svete

Zdravo svete

Када постоји један аргумент унутар range функције, он се користи за лимитирање низа бројева. Ако постоје два аргумента у range функцији, први аргумент се користи као почетна вредност секвенце а други аргумент као лимитер секвенце вредности.

074 Употреба два аргумента у функцији range

```
for broj in range(1, 5):  
    print(broj)
```

1

2

3

4

075 Употреба три аргумента у функцији range

```
for broj in range(1, 8, 2):  
    print(broj)
```

1

3

5

7

Када се користи и трећи аргумент (он се назива вредност корака (step value)), сваки следећи број у секвенци ће се повећати за вредност корака.

Питања и задаци за самосталан рад

Задаци

0166 Помоћу for петље написати код који штампа листу [-100, -200, 0, 100, 200] 5 пута.

0167 Помоћу for петље написати код који штампа појединачне елементе листе [-100, -200, 0, 100, 200].

0168 Следећи код изменити тако да се уместо листе користи функција range:

```
for x in [0, 1, 2, 3]:  
    print("Pajton!!")
```

0169 Колико циклуса се извршава у следећем коду:

```
for x in range[-2, 5, 2]:  
    pass
```

0170 Шта ће се приказати на екрану после ове петље:

```
for x in range[10, 5, -1]:  
    print(x)
```

0171 Написати код са for петљом који омогућава да се на екрану виде вредности за 1 веће од -10, 20 и 45.

0172 Написати код са for петљом који омогућава да корисник 5 пута унесе реч коју он жели.

0173 Направити код који исписује име корисника жељени број пута у две варијанте употребом обе врсте петљи.

0174 Написати код који омогућава да се четири пута испише име корисника, али обезбедити да корисник може да у свакој итерацији промени име које жели да се испише.

Бесконачне петље

Техника бесконачне петље

Осим у специјалним случајевима, петље морају имати унутар њих начин да се саме заврше, а то значи да унутар тела петље мора постојати нешто што ће променити вредност услова на False.

Ако петља нема начин да сама себе заустави, она се назива **бесконачна петља**.

Бесконачна петља наставља са радом све док се програм некако не прекине и генерално треба избегавати њено креирање.

Коришћење циљне промењиве унутар петље

У for петљи, смисао циљне промењиве је да упућује на сваки елемент низа док петља итерира.

Често се користи циљна промењива у рачунању или другим задацима унутар тела петље.

076 Приказ целих бројева и њихових квадрата

```
print('Broj\tKvadrat broja')  
print('-----')  
for broj in range(1, 5):  
    kvadrat = broj ** 2
```

```

    print(broj, '\t', kvadrat)
Broj    Kvadrat broja

```

```

-----
1        1
2        4
3        9
4       16

```

У телу петље се користи циљна променљива `broj` за рачунање вредности квадрата броја.

Пошто циљна променљива мења своју вредност сваком итерацијом (од 1 до 4) онда ће се сваком итерацијом мењати и вредност променљиве `kvadrat` (биће 1, 4, 9 и 16).

Корисничко управљање итерацијама

Често, програмер зна тачан број итерација које петља мора да изведе.

Али, понекад програмер мора да допусти кориснику да преузме контролу над бројем итерација у петљи.

077 Корисничка контрола броја итерација

```

print('Program prikazuje listu brojeva')
print('pocevsi od 1 i njihovih kvadrata.')
kraj = int(input('Do kojeg broja da idem? '))
print('Broj\tKvadrat broja')
print('-----')
for broj in range(1, kraj + 1):
    kvadrat = broj ** 2
    print(broj, '\t', kvadrat)
Program prikazuje listu brojeva
pocevsi od 1 i njihovih kvadrata.
Do kojeg broja da idem? 5
Broj    Kvadrat broja
-----
1        1
2        4
3        9
4       16
5       25

```

Постављање аргумената `range` функције

До сада `range` функција се користила за прављење низа са бројевима који су поређани од најмањег до највећег.

Функција `range` се може користити за генерисање секвенце бројева који су поређани од највећег до најмањег.

078 Рад са функцијом `range`

```

print('Broj\tKvadrat broja')
print('-----')
for broj in range(4, 0, -1):

```

```

    kvadrat = broj ** 2
    print(broj, '\t', kvadrat)
Broj      Kvadrat broja

```

```

-----

```

```

4         16
3          9
2          4
1          1

```

Као први аргумент, 4 означава прву вредност за циљну промењиву, други аргумент 0 ограничава до које вредности се може ићи а трећи аргумент -1 каже да се иде од највећег ка мањим бројевима у кораку по 1.

Питања и задаци за самосталан рад

Задаци

0175 Креирати бесконачну петљу која исписује дату бројчану вредност на екрану преко while петље.

0176 Креирати бесконачну петљу која исписује дату стринг вредност на екрану преко for петље.

0177 Петљу из задатка 1. модификовати тако да више не буде бесконачна петља.

0178 Коришћењем циљне промењиве x унутар for петље, направити програм који приказује половину квадрата сваког броја у датом опсегу целих бројева.

0179 Коришћењем циљне промењиве a унутар for петље, направити програм који из датог опсега целих бројева приказује све бројеве или од броја a до 0 (ако је a негативно) или од 0 до a (ако је a позитивно).

0180 Без обзира коју целобројну вредност корисник унесе, програм користи опсег вредности од 10 до 20. Написати код који омогућава да и поред унете корисникове вредности код ради у задатом опсегу приказујући на екрану бројеве из тог опсега. Да ли је ово пример корисничког управљања итерацијама?

0181 Написати код који за унуту позитивну целобројну вредност, сабира све бројеве између 0 и те вредности. Да ли је ово пример корисничког управљања итерацијама?

0182 Написати програм који сабира све позитивне целе бројеве које корисник уноси све док корисник не унесе негативну вредност. Приказати укупан збир унетих бројева.

0183 Ако се доток ваздуха у авионским моторима представи квадратима растућих бројева у

опсегу од 1 до 10, написати програм који приказује губитак дотока ваздуха ако се проценат губитака повећава као опсег целих бројева од 1 до 10 (за доток вредности 25 губитак је 5% од те вредности, за доток вредности 64 губитак је 8% од те вредности).

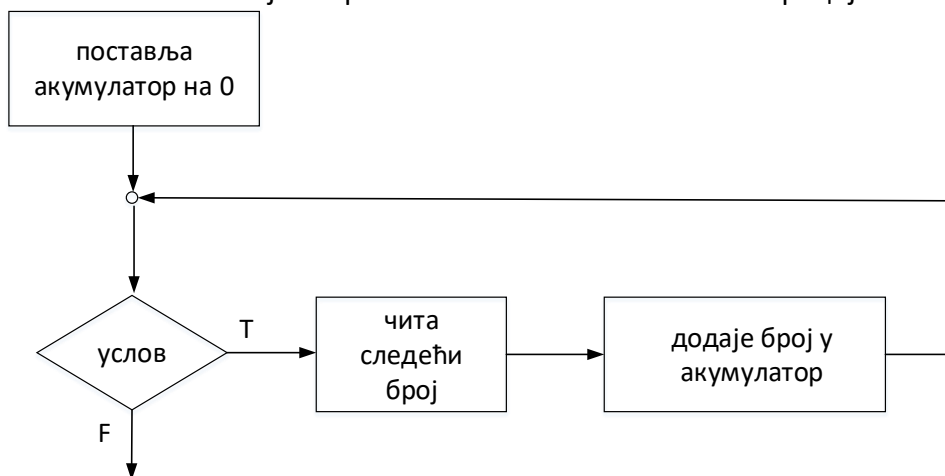
0184 За бројеве у опсегу од 1 до 10 прво приказати парне а затим и непарне бројеве (2, 4,...10, 1, 3,...).

Акумулатор у петљи

Рачунање тренутне суме

Тренутна сума (running total) је сума бројева који се сабирају при свакој итерацији у петљи. Промењива у коју се смешта тренутна сума се назива акумулатор.

У многим петљама је потребно извести математичке операције са низом бројева.



Тада се петљом чита један по један број у низу а једна промењива (акумулатор) чува суму прочитаних бројева.

Када се петља заврши, акумулатор ће садржати суму бројева које је петља прочитала.

Иницијализација акумулатора се изводи пре уласка у петљу и обично се акумулатор поставља на 0.

То је битно пошто се врши операција сабирања а неутралан број за сабирање је 0.

079 Сабирање бројева помоћу петље и акумулатора

```
max = 5
```

```
#inicijalizacija akumulatora
```

```
suma = 0
```

```
print('Ovaj program racuna sumu unetih brojeva.')
```

```
for brojac in range(max):
```

```
    broj = int(input('Unesi broj: '))
```

```
    suma = suma + broj
```

```
print('Suma je', suma)
```

Ovaj program racuna sumu unetih brojeva.

```
Unesi broj: 5
Unesi broj: -3
Unesi broj: 2
Unesi broj: 0
Unesi broj: 0
Suma je 4
```

У примеру, промењива бројас је циљна промењива а suma је акумулатор.

Појачани оператори доделе

Када се промењива која је на левој страни оператора доделе (=) појави и на десној страни ($x = x + 1$) то значи да се овакав израз може другачије представити са појачаним оператором доделе:

$x += 1$.

Овакав израз се чита: стара вредност промењиве се инкрементира (повећава за 1) и постаје нова вредност исте промењиве.

Списак свих појачаних оператора доделе: $+=$, $-=$, $*=$, $/=$, $%=$.

Стражари

Ако се током дизајнирања програма деси да није познат број вредности које ће се користити у низу или се зна да ће број вредности у низу бити другачији сваки пут када се програм стартује, постоји две могућности:

1. Питати корисника на крају сваке итерације да ли постоји још вредности за обраду
2. Питати корисника на почетку програма колико елемената има у низу

Обе могућности су нестабилне пошто зависе од корисника и зато је боља идеја користити стражара (sentinel).

Стражар је посебна вредност којом се означава крај низа елемената.

Када програм прочита вредност стражара, зна да је достигао крај низа, па се петља завршава.

080 Коришћење стражара

```
zbir = 0
strazar = int(input("Vrednost za kontrolu petlje: "))
while strazar != 0:
    zbir += strazar
    strazar = int(input("Vrednost za kontrolu petlje: "))

print('Zbir je', zbir)
Vrednost za kontrolu petlje: 4
Vrednost za kontrolu petlje: 6
Vrednost za kontrolu petlje: 2
Vrednost za kontrolu petlje: 0
Zbir je 12
```

081 Пребројавање непарних бројева у листи

```
neparni = 0
lista = [2, 5, 66, 33, 77]
```

```
for x in lista:
    if x % 2 != 0:
        neparni += 1

print("Neparni brojeva u listi ima", neparni)
```

Питања и задаци за самосталан рад

Задаци

0185 Написати програм који приказује колико је парних, непарних, нула а колико бројева већих од 10 у листи [-23, 55, 0, 55, 32, -102, 0, 1, 8, 0, 0]

0186 За дати опсег бројева колико је бројева са цифром јединице 5?

0187 Корисник уноси речи независне једне од других, колико је пута унео реч “skola”?

0188 Компанија уноси податке о запосленима тако што се прво унесе име запосленог а затим бројчани податак о том запосленом. Написати програм који броји колико пута је запослени са именом Zoran добио бројчани податак.

0189 Деца посматрају пролазак камиона са натписима (Sokovi, Peciva, Slatkisi) и броје камионе са таквим натписима. Написати апликацију која ће им помоћи да лакше преброје камионе са датим натписима и камионе које немају те натписе. На основу узорка од првих 20 камиона, апликација даје процентуално предвиђање колико камиона носи дати натпис.

Нестоване петље

Петље одобравања улаза

Програмерски израз “смеће уђе, смеће изађе” (garbage in, garbage out) (GIGO), се односи на чињеницу да рачунар не може да разликује доброг од лошег податка.

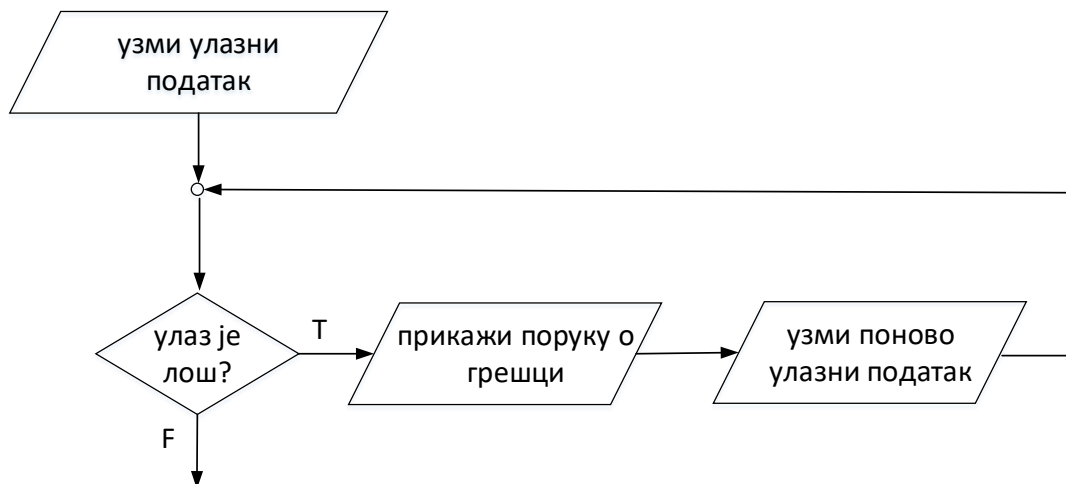
Ако корисник унесе лош податак, рачунар ће га обрадити и на излазу се добија лош резултат.

Овде се мисли чак и на случајне грешке које корисник направи погрешним укуцавањем са тастатуре.

Зато је неходно дизајнирати програм тако да се лош податак никако не може ни почети обрађивати, а то значи да је неопходно проверити све улазне податке пре обраде.

Ако је улазни податак лош, треба га одбацити и тражити од корисника да унесе добар податак.

Овај процес се назива одобравање улаза (input validation) а понекад и уређивач грешака (error handler).



Прву узимање улазног податка се назива примарно читање (priming read) и тај податак се први проверава у петљи.

Нестоване петље

Нестована (угнеждена) петља је петља унутар друге петље.

082 Могући резултати у фудбалу

```

for domacin in range(2):
    for gost in range(2):
        print(domacin, ': ', gost)
  
```

0 : 0

0 : 1

1 : 0

1 : 1

Види се да постоје две петље.

Прва петља управља бројем голова за домаћина а друга за госте.

Када променљива domacin добије вредност 0, gost ће извршити обе итерације са вредностима 0 и 1.

Када променљива domacin добије вредност 1, gost ће извршити обе итерације са вредностима 0 и 1.

Петљ gost је унутрашња петља а домаћин је спољашња петља.

Каже се да унутрашња петља брже итерира него спољна.

Множењем бројева итерације сваке од петљи, добија се укупан број обављених итерација у коду.

083 Цртање карактерима правоугаоника димензија 6x8

```
for red in range(8):
    for kolona in range(6):
        print('*', end='')
    print()
```

```
*****
*****
*****
*****
*****
*****
*****
*****
```

084 Цртање правоуглог троугла карактерима

```
broj_koraka = 6
for a in range(broj_koraka):
    for b in range(a):
        print('#', end='')
    print('#')
```

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

085 Цртање косе линије карактерима

```
broj_koraka = 6
for a in range(broj_koraka):
    for b in range(a):
        print(' ', end='')
    print('#')
```

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

086 Цртање обрнутог правоуглог троугла карактерима

```
broj_koraka = 6
for a in range(broj_koraka, 0, -1):
    for b in range(a):
```



```

        print('#', end='')
    print('#')
#####
#####
#####
###
##
#

```

Питања и задаци за самосталан рад

Задаци

0190 Нацртати следећи облик:

```

*
**
***
****
*****
****
***
**
*

```

0191 Нацртати следећи облик:

```

1
22
333
4444
55555
666666
7777777
88888888
999999999

```

0192 Корисник погађа један од три понуђена броја (0, 1, 2) и за то има само један покушај.
Написати програм који реализује игру.

0193 Нацртати следећи облик:

```

#
# #
#   #
#####

```

0194 Написати програм погађања замишљеног броја између 0 и 100, тако што рачунар задаје замишљени број играчу који га може погодити до 7 покушаја.

0195 Написати програм погађања замишљеног броја између 0 и 100, тако што играч задаје замишљени број рачунару који га може погодити до 7 покушаја.

Излазак из петље

Безусловно искакање из петље

Понекад је неопходно направити излаз из петље и пре него петља дође до свог природног краја. Такав излаз из петље се назива безусловно искакање из петље (реч безусловно описује команде којима није потребан никакав услов да би се извршиле).

У Пајтону се користи команда `break`, која прекида извршавање тренутне итерације петље, изазива искакање из петље и стартује следећу команду после тела петље.

Команда `break` се може користити и у `for` и у `while` петљи.

087 Проста употреба команде `break`

```
for x in range(5):  
    break  
print("U petlji smo stigli do", x + 1, ". iteracije.")
```

Даје:

U petlji smo stigli do 1.iteracije.

У примеру 01, бројач `x` добија вредност 0 у првој итерацији петље и одмах се извршава команда `break` која изазива искакање из петље и не дозвољава даље итерације петље.

Безусловни скок на следећу итерацију петље

Понекад је неопходно прекинути само текућу итерацију у петљи и омогућити нормалан наставак рада петље до краја њеног извршавања.

Такав скок у петљи се назива безусловни скок на следећу итерацију петље.

У Пајтону се користи команда `continue` која прекида извршавање текуће итерације у петљи и изазива старт следеће итерације петље.

Команда `continue` се може користити и у `for` и у `while` петљи.

088 Проста употреба команде `continue`.

```
for x in range(5):  
    print("Pocetak iteracije")  
    continue  
    print("Kraj iteracije")  
  
print("U petlji smo stigli do", x + 1, ". iteracije.")  
Pocetak iteracije  
Pocetak iteracije  
Pocetak iteracije  
Pocetak iteracije  
Pocetak iteracije  
U petlji smo stigli do 5 . iteracije.
```

089 Употреба команде break са for петљом

```
for slovo in 'Python':  
    if slovo == 'h':  
        break  
    print ('Slovo :', slovo)
```

090 Употреба команде break са while петљом

```
x = 10  
while x > 0:  
    print ('Trenutna vrednost:', x)  
    x -= 1  
    if x == 5:  
        break
```

```
print("Gotovo!")
```

091 Употреба команде continue са for петљом

```
for slovo in 'Python':  
    if slovo == 'h':  
        continue  
    print ('Slovo :', slovo)
```

092 Употреба команде continue са while петљом

```
x = 10  
while x > 0:  
    x -= 1  
    if x == 5:  
        continue  
    print ('Trenutna vrednost:', x)
```

```
print("Gotovo!")
```

Питања и задаци за самосталан рад

Задаци

0196 Написати програм који обрађује листу ["Beograd", "Nis", "Smederevo", "Novi Sad"] тако да када се појави вредност "Smederevo" прекида се извршење петље и прикаже се порука о лепоти тог града.

0197 Написати програм који у датом опсегу бројева сабира само непарне бројеве безусловним прескакањем парних бројева у опсегу.

0198 Написати програм који безусловним скоковима од стринга "H*aj??ja?ci???" приказује само

“Najjaci”.

0199 Корисник укуцава реалне бројеве. Написати програм који исписује на екрану само оне који немају децималан део (или су цели или су са уписаним 0 иза децималне тачке).

0200 Користећи безусловне скокове и Булове промењиве направити програм који не допушта даљи рад кориснику све док не унесе одговарајућу шифру.

0201 Направити програм који коришћењем безусловних скокова и Булових промењивих очекује двостепену тачну лозинку од корисника. Унос једног дела лозинке је независан од другог дела. Корисник неће добити приступ све док не унесе исправна оба дела лозинке.

0202 Корисник уноси целе бројеве један по један. Све док су ти бројеви мањи од дате вредности, они ће се сабирати међусобно. Када корисник унесе неку одређену вредност, програм се завршава.

0203 Направити игру погађања замишљеног броја између два људска играча. Ко први погоди исти замишљени број добија бод. Победник је онај ко први сакупи три бода.

Дефиниција секвенци

Секвенце (низови)

Секвенца (низ) је објекат који садржи велики број ствари који су заправо подаци.

Ствари у низу се смештају један после другог.

Пајтон омогућава велики број начина на које се изводе операције на стварима у низу.

Постоји неколико различитих типова објеката низова у Пајтону.

Два типа низова су основна: торке и листе, обе су секвенце које могу садржавати различите типове података.

Торке

Торке су врста секвенци (низова) и оне су непромењиве (immutable).

То значи да када се једном направе, не могу се променити.

Када се креира торка, њени елементи се затварају у заграда:

```
>>> moja_torka = (1, 2, 3, 4, 5)
>>> print(moja_torka)
(1, 2, 3, 4, 5)
>>>
```

У првој линији се креира торка додељивањем секвенце елемената промењивој `moja_torka`.

093 Итерација преко торке

```
imena = ('Miki', 'Kiki', 'Jovan')
for n in imena:
    print(n)
```

Miki
Kiki
Jovan

Торке подржавају индексирање, методе попут `index`, уграђене функције(`len`, `min`, `max`), изразе одсецања, оператор `in`, операторе `-` и `+`.

Али не подржавају методе попут `append`, `remove`, `insert`, `reverse`, `sort`.

Ако се жели направити торка само са једним елементом, мора се навести зарез после вредности елемента: `моја_torka = (1,)`

Торке имају смисла пошто је обрада торки бржа од обраде листа па се често користе за смештање великог броја података које се никада неће брисати или модификовати.

Рад са торкама

Оператор `*` се до сада користио као симбол за бинарну операцију множења.

Али, када је на левој страни `*` симбола секвенца (попут торке) а на десној страни је операнд у виду целог броја, овај симбол постаје оператор понављања.

Оператор понављања прави више копија торке а затим их удружује.

Општи формат оператора понављања: `торка * n`, где је `n` број копија торке.

094 Оператор понављања

```
>>> brojevi = (0, ) * 5
>>> print(brojevi)
(0, 0, 0, 0, 0)
>>>
```

Види се да израз `(0) * 5` прави пет копија торке `(0,)` а затим их удружује у једну торку, а крајња торка је додељена променљивој `brojevi`.

095 Оператор понављања код торки

```
>>> brojevi = (1, 2, 3) * 3
>>> print(brojevi)
(1, 2, 3, 1, 2, 3, 1, 2, 3)
>>>
```

096 Садржај постојеће торке доделити новој торци.

```
a = (1, 4, 8, 10)
b = a
print(b)
Даје: (1, 4, 8, 10)
```

097 Приказати поруку преко празне торке.

```
a = ()
if a == ():
    print("Torka je prazna")
else:
```

```
print("Torka nije prazna")
```

098 Торка са различитим типовима података

```
a = (1, "tekst", 2.0965, "", (445, 78))  
print(a)
```

099 Креирање торке преко функције range().

```
a = tuple(range(6))  
print(a)
```

0100 Креирање торке функцијом range() са опсегом.

```
a = tuple(range(5, 11))  
print(a)
```

Питања и задаци за самосталан рад

Задаци

0204 Написати програм који приказује на екрану следећу секвенцу (1, 4, 8, 10) као торку и уз то и одговарајућу текстуалну поруку.

0205 Написати програм који елементе секвенце (1, 4, 8, 10) повећава за 1 и приказује добијену секвенцу као нову торку.

0206 Написати програм који уноси све позитивне целе бројеве од 0 до унетог целог броја у торку.

0207 Написати програм који за унети број мањи од 10, уноси све бројеве између 0 и тог броја у једну торку а остале до 10 у другу торку.

0208 Ако је унети број паран, креирати торку са парним бројевима од 0 до тог броја, у супротном то исто урадити са непарним бројевима.

0209 Написати програм који упоређује елементе две торке (састављене од целих бројева). Резултат упоређивања може бити „veći prvi“ ако је већи елемент у првој торци, „veći drugi“ ако је већи елемент у другој торци, „isti su“ ако су оба елемента истих вредности. Приказати на екрану резултате упоређивања.

0210 Корисник уноси цео двоцифрени број са тастатуре. У прву торку сместити цифру јединица па цифру десетица а у другу торку исте цифре у обрнутом редоследу.

0211 Корисник уноси двоцифрени и троцифрени број. Цифре бројева сместити у различите торке. На екрану приказати поруке ако су највећи и најмањи бројеви у тим торкама исти.

Торке

Итерација по торци са for петљом

Познате су технике приступа појединачном знаку унутар стринга.

Многе од тих техника се могу применити и у торкама.

0101 Итерација по торци са for петљом

```
broj = (99, 100, 101, 102)
```

```
for n in broj:  
    print(n)
```

99

100

101

102

Индексирање

Други начин да се приђе појединачном елементу у торци је са индексом.

Сваки елемент у торци има индекс који одређује позицију елемента у торци.

Индекси започињу са 0, први елемент има индекс 0, други 1, итд.

Последњи елемент у торци од n елемената има индекс n -1.

moja_torka = (10, 20, 30, 40) има 4 елемента у торци и њихови индекси су 0, 1, 2 и 3.

Елементи ове торке се могу одштампати на овај начин:

```
print(moja_torka[0], moja_torka[1], moja_torka[2], moja_torka[3])
```

0102 Штампанье елемената торке помоћу петље

```
moja_torka = (10, 20, 30, 40)
```

```
indeks = 0
```

```
while indeks < 4:  
    print (moja_torka[indeks])  
    indeks += 1
```

10

20

30

40

Такође се могу користити негативни индекси за идентификовање позиције елемента релативно у односу на крај торке.

Пајтон интерпретер додаје негативне индексе на дужину торке за одређивање позиције елемента.

Индекс -1 идентификује последњи елемент у торци, -2 идентификује следећи до њега, итд.

0103 Негативни индекси у торци

```
>>> moja_torka = (10, 20, 30, 40)
>>> print(moja_torka[-1], moja_torka[-2], moja_torka[-3], moja_torka[-4])
40 30 20 10
```

Ако се напише погрешан индекс за торку, нпр у претходном случају : `print(moja_torka [4])`, појавиће се објава о грешци а то се за овај случај назива изузетак о грешци индекса (`IndexError exception`).

0104 Функције над торкама

```
def main():
    A = (1, 2, 3)
    print("Torka je", A)
    print("Duzina torke je", len(A))
    print("Najveci element u torci je", max(A))
```

`main()`

0105 Креирање торке са 5 копија торке ("а", "с").

```
def main():
    A = ("a", "c")
    B = A * 5
    print(B)
```

`main()`

0106 Штампанье елемената торке помоћу петље while.

```
def main():
    moja_torka = (1, 2, 3, 4)
    indeks = 0
    while indeks < 4:
        print (moja_torka[indeks])
        indeks += 1
```

`main()`

Питања и задаци за самосталан рад

Задаци

0212 Из торке (100, 200, 50), добити најмањи и највећи елемент. Убацити те елементе у нову торку. Приказати садржај торке.

0213 Креирати торку од n копија торке (10, 55, 33).

0214 Одштампати елементе торке (1, 2, 3, 4, 5) помоћу for петље.

0215 Приказати елементе торке (1, 2, 3, 4, 5) коришћењем for петље и само негативних индекса.

0216 Корисник уноси позитиван број. Креирати торку која се састоји од највеће цифре унетог броја, поновљене онолико пута колико је цифара у унетом броју.

0217 Корисник уноси три различита једноцифрена позитивна броја. Направити торку са свим троцифреним комбинацијама унетих цифара без понављања цифара.

Коришћена литература

- Tony Gaddis “Starting Out with Python”
- Allen B.Downey “Think Python”
- Laura Cassel, Alan Gauld “Python Projects”
- Magnus Lie Hetland “Python Algorithms”
- John V. Guttag “Introduction to Computation and Programming using Python”
- David Beazley, Brian K. Jones “Python Cookbook”
- Charles R. Severance “Python for Everybody”
- python.org
- flowgorithm.org, Devin Cook
- realpython.com
- w3schools.com
- learnpython.org
- coursera.org

