Милан Чабаркапа

УВОД У ПРОГРАМИРАЊЕ КОРИШЋЕЊЕМ PYTHON-А

Један од основних задатака наставе програмирања је стицање фундаменталних знања и вештина, која бурна информатичка револуција неће учинити бескорисним. Оно што има највише изгледа да преживи све промене, које у великој мери потиру оно што је претходно створено, јесте област програмирања – алгоритмизација. Зато је један од основних циљева наставе програмирања развијање алгоритамске културе. На универзитетима широм света, до скоро се за учење алгоритама, у великој мери, користио псеудојезик. Често се постављало питање зашто се избегава коришћење неког популарног и моћног програмског језика? Зато, што:

- језик који је данас популаран сутра већ није;
- већина професионалних језика има врло компликовану синтаксу и структуру, који почетнику „замагљују“ алгоритме као основни циљ учења.

Али, учење алгоритама у псеудојезику има једну озбиљнуману мотивационог карактера – ученику је ускраћено задовољство да види како, и да ли, његов програм „ради“ на рачунару. Међутим, последњих година псеудојезик као да ишчезава из наставе, а место њега се користи Python, програмски језик широке намене и једноставне синтаксе, који на универзитетима и у школама широм света постаје доминантан у уводним курсевима из алгоритама и програмирања. Масачусетски технолошки институт (MIT) – водећи светски центар за инжењерско образовање у настави програмирања користи Python, који не само да је добар избор за учење основа програмирања, већ га све више користе професионалаци у разним областима примене: веб програмирању, системском програмирању, GUI апликацијама, роботици, програмирању игара, статистици, машинском учењу, итд. Чак су и популарне игре Battlefield 2, Civilization 4 и QuArK имплементиране коришћењем Python-а. Огроман је број компанија које користе овај програмски језик. Поменимо само неке: YouTube, Google, Dropbox, Yahoo, Disney, NASA, Nokia, IBM, Intel, Cisco, HP итд. Python је, због највећег раста популарности у 2018. години, од портала TIOBE, који мери рејтинг програмских језика, проглашен програмским језиком 2018. године.

За успешно напредовање у програмирању неопходно је ништа мање вежбе него за успех у спорту, музици, математици, итд. Зато у току учења треба експериментисати, решавати проблеме, отклањати уочене грешке, уместо пасивног читања у коме се задовољавате разумевањем синтаксе и готових решења проблема.

Инсталација Python-а под оперативним системом Windows

Пре него што почнете да учите да програмирате у програмском језику Python неопходно је да на вашем рачунару урадите следеће три ствари:

- преузмете Python инсталацију;
- инсталирате Python;
- тестирате Python на једноставним програмима.

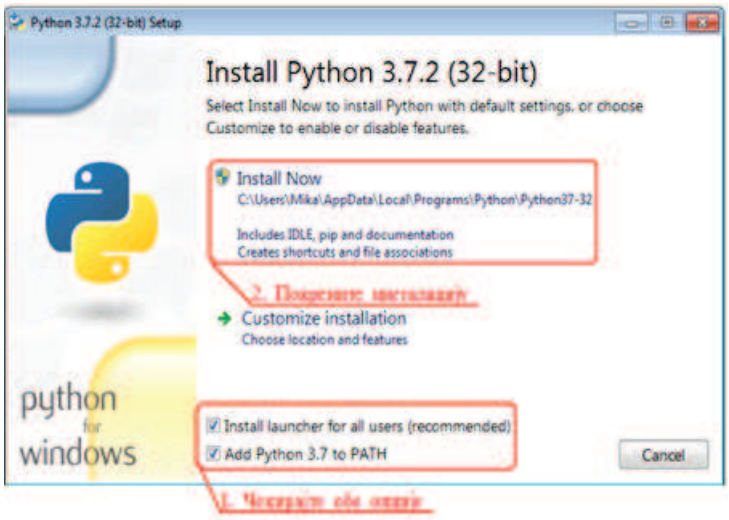
Python интерпретер је бесплатан и можете га преузети са странице `https://www.python.org/downloads/`. За кориснике Windows оперативног система постоје две верзије инсталатора: за 32-битне и за 64-битне системе. Ради инсталације поступите по следећем упутству:

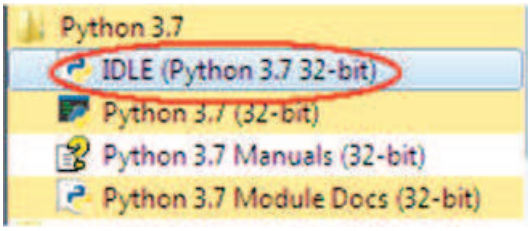
1. Покрените веб-читач, пређите на страницу `www.python.org/downloads` и преузмите инсталациони фајл за вашу верзију оперативног система. У примеру са слике преузет је инсталациони фајл за 32-битни Windows оперативни систем.

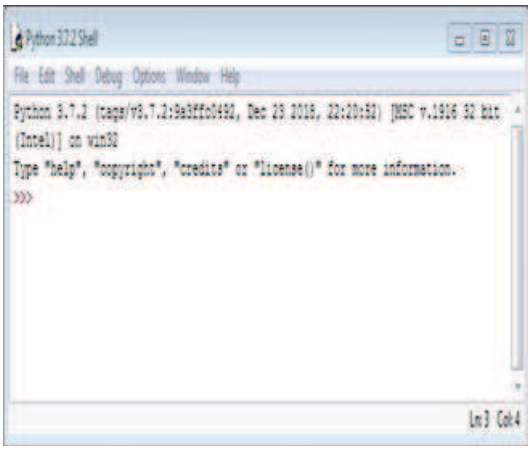
2. Покрените преузети инсталациони фајл и у првом инсталационом прозору чекирајте инсталацију за све кориснике и додавање путање у системску променљиву Path, а затим кликом на Instal Now покрените инсталацију.

3. Након инсталације Python интерпретера рестартујте рачунар, у Start менију пронађите Python групу и у њој кликните на IDLE (Python ...) да бисте покренули интегрисано развојно окружење IDLE (Integrated DeveLopment Environments).

4. Појавиће се прозор Python Shell у којем можете радити у интерактивном режиму, у коме после сваке инструкције Python интерпретер даје резултат њеног извршавања.







Пошто још нисте научили ниједну Python инструкцију, пробајте да у интерактивном режиму радите као да имате на располагању обичан калкулатор (систем то омогућава). Знак >>> који видите у прозору представља поруку да систем очекује да унесете инструкцију. Након што унесете инструкцију и притиснете

тастер **Enter** интерпретер је одмах извршава и у следећем реду исписује резултат. Пробајте да експериментишете задавањем неких израза чију вредност треба израчунати. На пример:

```
>>> 3+6
9
>>> 4*(3-9)
-24
>>> 7.08+3.22/4
7.885
```

Водите рачуна да када уносите реалне бројеве за раздвајање целобројног и разломљеног дела користите тачку, а не запету.

У Python-у можете израчунавати вредности израза какве се у C/C++ или Java-и не могу добити без специјалних библиотека за рад са великим бројевима. На пример, пробајте да израчунате 2 на 1024:

```
>>> 2**1024
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377767893424865485276302219601246094119453082952085005768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216
```

Било коју од унетих инструкција, ако треба да је поновите или модификујете, можете копирати у текућу командну линију ако курсор поставите на крај линије у којој је инструкција коју желите да копирате и притиснете **Enter**.

Python Shell (интерактивни режим) можете затворити кликом на дугме **Close** или комбинацијом тастера **Ctrl**+**D**.

Интерактивни режим рада је врло користан када радите са кратким изразима, упознајете нове могућности језика или желите да консултујете његову документацију.

Основни симболи Python-а

Изучавање било ког природног језика почиње изучавањем симбола од којих се граде његове речи, да би се од њих формирале реченице. Аналогно је и при изучавању програмских језика. Пре него што се упознамо како се граде исправне конструкције Python-језика, морамо знати који нам симболи стоје на располагању.

У Python-у се користе следећа слова:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

цифре: 0 1 2 3 4 5 6 7 8 9

и специјални симболи: _ + − ∗ / = < > [ ] . , ; : Ь ( ) { } итд.

Специјалним симболима се сматрају и нештампајући (невидљиви) симболи: празнина, симбол који означава нову линију, симбол за табулацију итд.

Скуп основних симбола Python-а је уређен и коначан, тј. сваки симбол има редни број.

Кључне речи су резервисане речи, јер имају специјално значење за интерпретатор и користе се само за намену за коју су дефинисане. То су: `and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield, True, False, None.`

Са значењем наведених кључних речи упознаћемо се у току изучавања конструкција језика.

Ради боље документованости и читљивости програма његови елементи се раздвајају празнинама и текстовима коментара.

Коментари

Велике и компликоване програме је врло често тешко разумети. Зато се ради појашњавања смисла појединих делова програма могу користити коментари. Коментаром се у Python-у сматра сваки низ симбола који почиње симболом #. На пример:

```
# U sledecem odeljku upoznacemo se sa
# pojmom promenljive u Python-u
```

Коментаре Python интерпретер игнорише, јер су они намењени онима који читају програм. Иако ће вам се чинити да је писање коментара сувишно, нарочито када су програми једноставни, навику да их пишете треба да развијате, јер ће вам олакшати модификацију и отклањање грешака у комплекснијим програмима.

Промењљиве

Када ваш програм треба да реализује нека израчунавања мораћете да вредности података које учествују у израчунавањима чувате на неки начин. У Python-у се за њихово чување користе *промењљиве*. Сваку промењљиву карактеришу име и тип вредности која се чува у објекту на који промењљива реферише. На пример, ако у Python-у напишете инструкцију доделе:

```
x=10
```

у меморији се креира објекат у коме је регистрована целобројна вредност 10, на који реферише (другим речима „чију адресу садржи“) промењљива x.

Када се након ове доделе у неком изразу у програму наведе име промењљиве x, *користи се вредност на коју реферише, а не име промењљиве.* На пример, ако се у програму појави израз:

`x+1`

израчунава се његова вредност: `10+1 -> 11`.

Елемент програмског језика, који се назива *променљива*, добио је име због тога што се, у току извршавања програма, могу мењати објекти (вредности) на које реферише.

Име променљиве је произвољан низ слова, цифара и симбола подвлачења. Име променљиве *не сме* почињати цифром. У својству имена се не смеју користити кључне речи.

Знак за подвлачење се најчешће користи да у именима, која се формирају од више речи, замени празнину, јер је празнина недозвољен симбол имена. Према томе, некоректно је ако у својству имена уведемо: `Porez na promet`. Уместо тога може се ставити: `PorezNaPromet`, `Porez_na_promet` или `poreznapromet`. Пожељно је да име исказује што потпунију информацију о објекту на који се односи, тако да име треба формирати од више речи као у наведеним примерима.

Свако име мора бити јединствено. Велика и мала слова у именима се разликују, тако да `ABC`, `abc`, `ABc` означавају различита имена.

Пример 1. Дозвољена имена су:

`Skola` `Kamata` `Zatezna_kamata` `ZateznaKamata` `zateznakamata`

а недозвољена:

`break` – кључна реч;

`3dan` – почиње цифром;

`X+y` – садржи недозвољен знак `+`;

`elif` – кључна реч.

Input() функција

Када креирате програм, врло често ћете имати потребу да учитавате податке, јер програм и пишете ради обраде улазних података. У Python-у учитавање можете реализовати коришћењем `input()` функције, која зауставља програм и чека да корисник унесе податак употребом тастатуре. Након што корисник унесе податак и притисне `Enter`, функција прихвата улазну вредност и ставља на располагање програму.

Пробајте да у интерактивном режиму унесете следеће две инструкције:

```
>>> ime=input("Kako se zoves:  ")   # funkcija input ucitava string
Kako se zoves:  Vuk
>>> print(ime, "je perspektivan programer!")  # print ispisuje string
Vuk je perspektivan programer!
```

У првој инструкцији функција `input()` исписује свој аргумент – низ знакова: `Kako se zoves:`, који је порука кориснику којом се тражи да унесе име.

Након што корисник унесе име и притисне `Enter` у меморији се креира објекат који ће садржати унети низ знакова, а адреса објекта се додељује промењивој `ime`. После ове доделе вредност која се налази у објекту постаје доступна посредством промењиве `ime`, другим речима када се у програму појави промењива `ime` користиће се вредност регистрована у објекту на коју показује промењива. *Убудуће када се каже да промењива има неку вредност, мисли се на вредност која се налази у објекту на који реферише (чију адресу садржи) промењива.* Општи облик доделе вредности промењивој коришћењем `input()` функције је следећи:

```
<ime promenljive>=input(<poruka>)
```

Име промењиве се увек пише на левој страни.

Унето име се у претходном и у следећа два примера исписује инструкцијом, која користи функцију `print()`:

```
>>>print(ime, "je perspektivan programer!")
```

У програмерској терминологији низ знакова се назива `string`. Када се "string" пише у програмском коду назива се `string literal`. У Python програму стринг литерал мора имати граничне симболе – пар апострофа (') или пар наводника (''). У претходном примеру стринг литерали су ограничени наводницима:

```
"Kako se zoves:  "
"je perspektivan programer!"
```

али могли су се користити и string literali ограничени паром апострофа:

```
'Kako se zoves:  '
'je perspektivan programer!'
```

Инструкција доделе вредности промењивој

Инструкција доделе вредности промењивој служи за израчунавање вредности израза и доделу промењивој на левој страни знака доделе: =. Општи облик ове наредбе је:

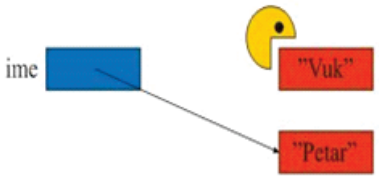
```
<ime promenljive>=<izraz>
```

У инструкцији доделе Python интерпретер прво израчунава вредност израза на десној страни од знака =, затим у меморији креира објекат који ће садржати добијену вредност и тек онда промењивој додељује адресу објекта. Када промењива први пут добије вредност каже се да је промењива *дефинисана* или *иницијализована*. Треба да имате на уму да знак = не означава једнакост (као у математици) већ доделу.

Ако наставимо рад у интерактивном режиму вредност промењиве `ime` можемо променити и доделити јој другу вредност, на пример:

```
>>> ime='Petar'
>>> print('I', ime, "misli da je perspektivan programer!")
I Petar misli da je perspektivan programer!
```

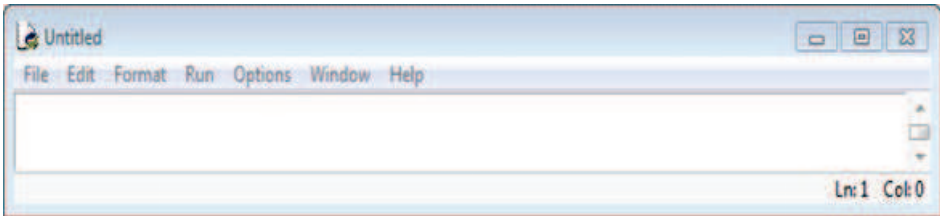
Након ове доделе мења се садржај промењиве `ime`, уместо на објекат који садржи стринг `"Vuk"`, променљива показује на објекат који садржи стринг `"Petar"`. Python интерпретер аутоматски уклања објекат који садржи стринг `"Vuk"`, јер на њега више не реферише ниједна променљива.



Никад немојте наводити име променљиве којој нисте доделили вредност, јер ће вам тада Python дати поруку да променљива са тим именом није дефинисана.

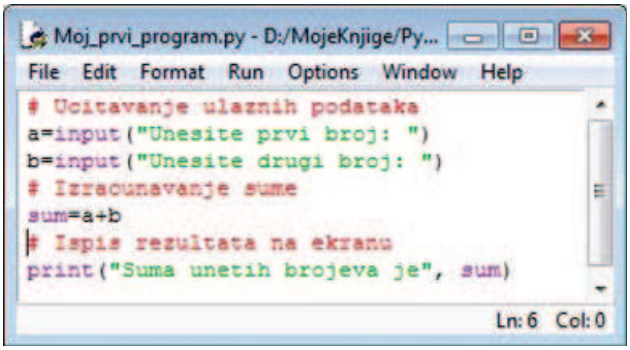
Рад у програмском режиму

У интерактивном режиму могу се писати врло прости програми. За писање сложених програма користи се програмски режим рада у коме се прво напише комплетан програм, сачува у фајлу на диску и тек онда даје на извршавање. Ради илустрације рада у програмском режиму написаћемо једноставан програм који сабира два броја.

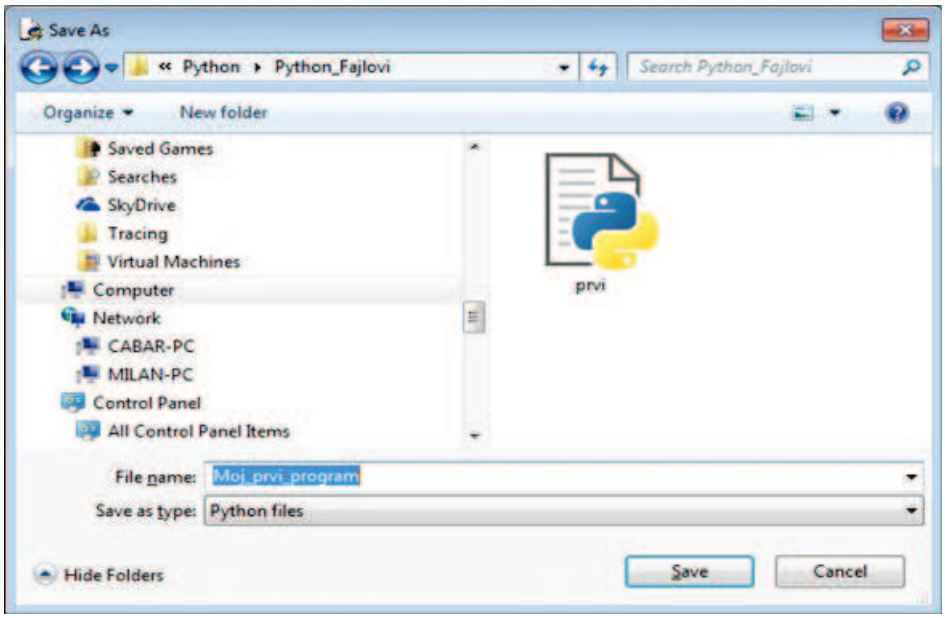


1. Пређите у програмски режим избором у `File` менију ставке `New File` или истовременим притиском на тастере `Ctrl+N`. Појавиће се прозор за креирање програма.

2. Унесите следећи програм:

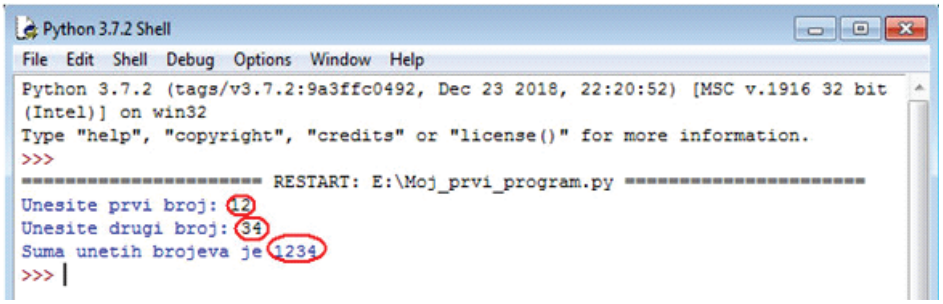


3. Сачувајте програм избором у `File` менију ставке `Save As...` или истовременим кликом на тастере `Ctrl+S`. Појавиће се дијалог за избор имена фајла у коме је програм, као и фолдера у коме треба да се сачува.



Назовите фајл Moj_prvi_program. Систем ће том имену аутоматски додати .py. Фајлови програма се подразумевајуће смештају у системски фолдер – у коме је инсталација Python IDLE. За ваше програме боље би било да креирате посебан фолдер.

4. Да бисте задали извршавање програма из менија `Run` изаберите опцију `Run Module` или притисните на функционални тастер `<F5>`. Резултат извршавања програма појавиће се у прозору Python Shell (прозор интерактивног режима).



Извршавањем програма добили сте вероватно неочекиван резултат – сабирањем два унета броја добили сте број настао њиховим надовезивањем – од 12 и 34 добили сте 1234. До тога је дошло због тога што `input()` функција унети податак програму предаје као стринг (низ знакова). Када се над два стринга примени операција сабирања `'+'` добија се нови стринг настао дописивањем другог стринга на први. Грешка се може отклонити тако што ће се учитани стрингови претворити у нумеричке податке: целобројног типа применом функције `int()` или реалног типа применом функције `float()`. Модификујте програм тако што ћете улазне стрингове конвертовати у целобројне податке.

5. Да бисте се вратили у програмски режим затворите Python Shell (прозор интерактивног режима) кликом на `Close` дугме и унесите следећу исправку:

```
a=int(input("Unesite prvi broj:  "))
b=int(input("Unesite drugi broj:  "))
```

На овај начин у обе линије стринг кога уноси корисник, а који прихвата функција `input()` и предаје функцији `int()` претвара се у цео број. Сада променљиве `a` и `b` реферишу на целе бројеве тако да операцијом сабирања:

```
sum=a+b
print("Suma unetih brojeva je", sum)
```

добија се и приказује тачан резултат. Ако програм треба да ради са реалним бројевима уместо функције `int()` треба применити функцију `float()`.

6. Тестирајте програм након корекција.

Контролна питања

1. Како се формира име променљиве у Python-у?
2. Која од наведених имена могу бити имена променљивих, а која не:

а) `Disk` б) `8086Procesor` в) `Rok` г) `R*d*s` д) `P4` ?

3. Објасните зашто наредна имена нису дозвољена:

а) `2019godina` б) `Prihod-Rashod` в) `while` г) `2end`.

Математичка гимназија, Краљице Наталије 37, Београд
E-mail: mcabark@gmail.com