СТРУЧНИ РАД

УДК 004.43 PYTHON
004.42

ЗАШТО ЈЕ ПРОГРАМСКИ ЈЕЗИК ПАЈТОН ДОБАР ЗА УЧЕЊЕ ПРОГРАМИРАЊА

Небојша Васиљевић, nebojsa.vasiljevic@mtt.gov.rs, Република Србија,
Министарство спољне и унутрашње трговине и телекомуникација

Избор програмског језика за учење програмирања значајно зависи од околности, па се не може дати универзалан одговор који језик је најбољи, али постоје програмски језици који се по својим карактеристикама издвајају као погодни за учење програмирања.

Често се програмирање учи у функцији неког конкретнијег циља, што онда условљава избор програмског језика. На пример, ако је главни мотив за учење програмирање прављење Андроид апликација, програмирање ће се учити на Јави, а ако вам је нагло искрсла потреба да се сналазите са макроима за *Microsoft Word* или *Excel*, онда ћете прве кораке у програмирању направити кроз Visual Basic. У оваквим случајевима се питање избора програмског језика очито ни не поставља.

Први кораци у програмирању могу се стећи и са алатима који се заснивају на слагању графичких елемената или који користе посебно дизајниран програмски језик. У таквим случајевима ми заправо не бирамо програмски језик, већ алат у целини и неки алати овог типа, као што је Скреч (енгл. *Scratch*[1]) могу бити заиста добар избор за прве кораке у програмирању, посебно код млађих узраста. Међутим, пре или касније у учењу

1 http://scratch.mit.edu/

програмирања морамо доћи до класичног програмског језика и ту се враћамо на питање које нам је овде тема.

Ако желите да се посветите упознавању идеја и стицању вештина програмирања више него савладавању самог програмског језика, и при томе желите да изаберете програмски језик који је солидно заступљен у пракси, тешко да можете да направите бољи избор од Пајтона (енгл. *Python*[2]).

Уколико желите да се упознате са програмским језиком Пајтон пре него што наставите да читате или у неком тренутку касније, препоручујемо вам да погледајте интерактивне туторијале за Пајтон на *Codeacademy*[3].

У наставку су посебно истакнуте и образложене поједине каратеристике програмског језика Пајтон које га чине добрим избором за учење програмирања.

1. Сврстава се у 10 највише коришћених програмских језика

За учење програмирања треба изабрати језик који живи у пракси и широко је заступљен. Подршка алата и библиотека, присуство на разним платформама, искуства колега, расположива литература и реално очекивање да језик неће пасти у заборав у наредних пет или десет година су важне претпоставке код избора програмског језиика.

Наравно, не тврдимо да праг за избор програмског језика треба буде баш првих 10, али је чињеница да Пајтон улази у првих 10 у готово свим методологијама мерења.[4]

Овде намерно нисмо истакли аргумент да треба учити програмски језик који ће се највероватније користити касније у професионалном раду. Фокус учења програмирања не треба да буде на мајсторству коришћења програмског језика, већ на концептима који се могу изражавати у разним програмским језицима, а један треба изабрати за учење. Чак и онај ко никада не буде професионалан програмер има користи од разумевања основних концепата програмирања.

2.Скрипт језик са динамичким типовима и аутоматским управљањем меморијом

Чињеницу да је Пајтон скрипт језик са динамичким типовима неко ће видети као предност, а неко каоману, али за увод у програмирање преовлађују предности.

Пре свега, најједноставнији примери су заиста једноставни, без „вишка кода“ за који у првим лекцијама морамо да кажемо „ово мора тако да стоји, касније ћемо научити чему служи“. Пример за први програм у Пајтону би могао да буде:

```
print('Zdravo svima!')
```

За разлику од еквивалентног програма у, на пример, програмском језику *C++*:

```
#include <iostream.h>
int main()
  {
    cout << «Zdravo svima!»;
  }
```

Пајтон може да се извршава у интерактивној конзоли (познатој и као *Python Shell*), тако да једноставне примере одмах извршавате и видите резултат. Ово је изузетно практично за

2 http://www.python.org/

3 http://www.codecademy.com/tracks/python

4 Чланак *Measuring programming language popularity* на Википедији (http://en.wikipedia.org/wiki/Measuring_programming_language_popularity) наводи пет извора мерења популарности програмских језика и у време писања овог текста (мај 2013. године) свих пет извора

Пајтон сврстава међу првих 10 најпопуларнијих програмских језика

прве кораке, али и касније када нешто желите да пробате или проверите.

Програми у Пајтону су текстуалне датотеке са екстензијом „*.py*“. Нема компилације ни додатних датотека које се при компилацији формирају нити обавезног креирања пројекта за сваки пример и слично.

Пајтон као интерпретатор не достиже ефикасност језика који се компилирају попут *C++*, нити језика попут Јаве или *C#* који имају динамичко превођење (енгл. *just in time compilation*), али је довољно ефикасан за многе примене, укључујући учење програмирања[5].

Чињеница да променљиве не морају да се декларишу и да немају статички одређен тип почетницима олакшава програмирање и растерећује их потребе за разумевањем често сложених синтаксних конструкција за специфицирање типова. Узмимо пример Пајтон програма:

```
def sum(x, y):
  return x + y

a = sum(42, 47)
b = sum('Maja', 'Marina')
print(a, b)
```

Релативно је лако и разумети и објаснити да ће овај програм да испише:

```
89 MajaMarina
```

Приметите да нисмо морали да кажемо ништа о полиморфизму, генеричким типовима, шаблонима и слично. Такође приметимо да свака вредност са собом носи свој тип, тако да је операција „+“ на одговарајући начин примењена на целобројне вредности и на ниске. Пајтон не прави разлику између примитивних типова и објеката и нема много варијација сличних типова. Не размишљамо о томе да ли да користимо *float* или *double* (за програмски језик *C* на ову тему може цела лекција да се исприча) или пак *int, long, Integer, Long* или *BigInteger* (у програмском језику Јава).

Пајтон има аутоматско управљање меморијом, без експлицитне алокације и деалокације меморије и потребе за експлицитним коришћењем показивача.

3. Једноставна и разумљива синтакса

Синтакса програмског језика Пајтон је прегледна и разумљива, што ћемо илустровати на примеру програма који одређује број појављивања задатог карактера у нисци:

```
def prebroj(karakter, niska):
    brojac = 0
    for k in niska:
        if karakter == k:
            brojac = brojac +1
    return brojac

print(prebroj('a', 'aparat'))
```

Претходни програм ће исписати број 3, јер се карактер *'a'* три пута појављује у нисци *'aparat'*.

Пајтон користи карактеристично груписање назубљивањем, што је прегледно за читање, нема додатних заграда нити кључних речи, али је потребно да се корисник навикне на овакав начин записивања.

5 Поједине алтернативне имплементације Пајтона попут *PyPy* (pypy.org) или *Cython* (www.cython.org) подржавају одређене методе компилације. *PyPy* подржава динамичко превођење и може се користити као замена за стандардну имплементацију (коју обично називамо *CPython*) са солидним нивоом компатибилности, док *Cython* преводи Пајтон kôd у програмски језик *C* и обично се користи за превођење појединих модула који се онда укључују у *CPython* као екстензије.

4. Ниске, листе, н-торке, речници и скупови су уграђени типови са једноставном синтаксом за коришћење

Ниске и основни типови колекција, као што су листе променљиве дужине, н-торке, речници (асоцијативни низови) и скупови представљају уграђене типове добро уклопљене у систем типова и синтаксу језика, са свим потребним операторима и предефинисаним фунцијама.

Овим је омогућено да се основни типови колекција од почетка користе у решавању проблема када је то погодно, уместо да се уводе тек у напредним лекцијама, након што се научи како се имплементирају или зато што се налазе у посебним библиотекама које треба на посебан начин користити.

Данас је примерено учити како се користи сортирање за решавање проблема пре него што се уче алгоритми за сортирање, како се користе речници пре него што се учи имплементација хеш табела, итд. То је исто као што су у неком тренутку људи почели да уче више програмске језике, а да нису морали претходно да науче асемблер и имплементацију језичких процесора.

За учење програмирања је потребно да се основни типови колекција третирају као елементарне градивне јединице које се користе у решавању проблема и програмски језик треба да омогући њихово коришћење без икакве претпоставке разумевања додатних напредних концепата (шаблона, генеричких типова и слично). Ово је пример где се у променљивој *ocene* имену сваког ученика додељује листа оцена:

```
ocene = {}
ocene['Ivan'] = [3, 5, 4]
ocene['Ana'] = [5, 4, 5, 5]
```

што може и краће да се напише овако:

```
ocene = {'Ivan': [3, 5, 4],
'Ana': [5, 4, 5, 5]}
```

Сличан кôд у Јави би изгледао овако:

```
Map<String, List<Integer>> ocene
=
    new HashMap<String,
List<Integer>>();
ocene.put("Ivan", Arrays.
asList(3, 5, 4));
ocene.put("Ana", Arrays.
asList(5, 4, 5, 5));
```

При томе *Arrays.asList(3,5,4)* даје листу фиксне дужине, а ако хоћемо листу променљиве дужине, онда бисмо морали да кажемо *new ArrayList<Integer>(Arrays.asList(3,5,4))*. Јава има добру подршку за основне типове колекција, али је неопходно претходно савладати орећене напредније концепте програмског језика да би се та подршка користила са разумевањем.

Природна подршка за основне типове колекција омогућава да се структуре података које се појављују у елементарним алгоритмима једноставно конструишу од основних типова колекција и других уграђених типова, без дефинисања нових типова. На пример, дрво аритметичког израза *5 * (2 + 3)* би могло да се конструише овако:

```
izraz = ['*', 5, ['+', 2, 3]]
```

Наравно, Пајтон омогућава дефинисање нових типова, односно класа, што ћемо касније и илустровати.

5. Успешно замењује калкулатор

Ако Пајтон користимо у интерактивном режиму (*Python shell*), он може да нам замени калкулатор. На пример:

```
>>> 2+2
    4
```

„>>>“ је одзивни знак (енгл. *prompt*) и он

стоји испред онога што смо ми откуцали, а у следећој линији је резултат.

Калкулатор често користимо у решавању задатака попут: „Ако је човек током времена *t*=3.5s прешао пут од *s*=4.5m, колика му је просечна брзина?“ Формула за брзину је *v=s/t*, па кад заменимо бројеве и то унесемо у „калкулатор“ добијамо:

```
>>> 4.5/3.5
    1.2857142857142858
```

Пајтон је „паметнији“ од обичног калкулатора, па можемо писати и:

```
>>> t=3.5
>>> s=4.5
>>> s/t
    1.2857142857142858
```

Ако имамо резултате пет мерења времена за које је човек претрчао 100 метара, овако бисмо могли да израчунамо брзине:

```
>>> merenja = [15.3, 11.7, 21.9,
13.2, 14.6]
>>> s = 100
>>> [s/t for t in merenja]
[6.5359477124183005,
8.547008547008547,
4.566210045662101,
7.575757575757576,
6.8493150684931505]
```

Увођење појединих елемената програмирања као проширених могућности калкулатора, полазећи од проблема за чије решавање је ученик навикао да користи калкулатор, представља ефикасан метод да се ученик доведе до првих линија кода уз минимално оптерећивање новим концептима.

6. Омогућава вишеструку доделу

Да би променљиве *a* и *b* замениле вредности, у Пајтону је довољно написати:

```
a, b = b, a
```

Без вишеструке доделе, морали бисмо уз помоћну променљиву да користимо три доделе. И функција може да врати вишеструку вредност (у ствари враћа н-торку), као на пример *divmod* која рачуна и целобројни количник и остатак при дељењу:

```
d, m = divmod(p, q)
```

Еуклидов алгоритам за највећи заједнички делилац[6] би у Пајтону могао да се кодира овако:

```
def nzd(a,b):
    while b!=0:
        a, b = b, a % b
    return a
```

Вишеструка додела је један детаљ који олакшава изражавање и помаже да се више концентришемо на суштину алгоритма.

7. Математички је доследан

Од петог разреда основне школе ученицима је познат појам остатка при дељењу који је једнак нули у случају да је дељеник дељив са делиоцем, а иначе је већи од нуле и мањи од делиоца. У многим програмским језицима, укључујући Пајтон, за операцију остатка при дељењу се користи знак „%“. Тако је *5 % 3 = 2*. Пајтон даје математички коректан резултат и када је дељеник негативан, што у многим другим програмским језицима није случај (као што су *C*, *C++*, *C#* или Јава).

На пример, у Пајтону је *-1 % 3* једнако *2*, за разлику од неких других програмских језика где је резултат *-1*. Дакле, можете слободно да

6 За опис Еуклидовог алгоритма можете погледати поглавље Алгоритми у онлине материјалима проф. Цветане Крстев, http://poincare.matf.bg.ac.rs/~cvetana/Nastava/Materijal/Algoritmi_1213.pdf

напишете:

```
pozicija = (pozicija + pomeraj)
% sirina
```

уместо да се на следећи начин штитите од негативних вредности за случај када је померај негативан:

```
pozicija = (pozicija + pomeraj +
sirina) % sirina
```

Можда изгледа као детаљ, али модуларна аритметика (рачунање са остацима при дељењу) је део математике који се интензивно користи већ од почетних нивоа програмирања и није згодно када ученику морате да објашњавате зашто нешто није онако како се учило из математике и како то треба да превазиђу.

Пајтон такође уводи посебан оператор за целобројно дељење „//“[7]. Модуларна аритметика у Пајтону је проширена и на бројеве у покретном зарезу, тако да задатак „Колико дасака дужине *X* може да се исече из дрвета дужине *Y* и колики је део који остаје, где дужине не морају да буду цели бројеви“ може да се реши на следећи начин:

```
print(X // Y, X % Y)
```

У математици се често појављују неједнакости попут $a < b < c$, што се на други начин може написати као $a < b$ и $b < c$. Већина програмских језика не подржава први начин записивања вишеструких релација, већ омогућава само запис који је базиран на другом начину где су раздвојене две релације. У Пајтону слободно можете написати:

```
if a < b < c:
  levi, srednji, desni = a, b, c
```

Пајтон подржава велике бројеве у целобројној аритметици, што значи да ће математички коректно израчунати вредност попут $\frac{2013^{1000}}{2013^{999}}$:

```
>>> print(2013**1000 //
2013**999)
    2013
```

Пајтон интерно разликује обичне и велике целобројне вредности, али није неопходно да програмер о томе води рачуна.

Математичка доследност Пајтона вам помаже да се лакше надовежете на оно што је ученик већ научио из математике и да се мање бавите триковима програмског језика, што вам даје више простора да посветите пажњу проблему који решавате и концептима програмирања.

8. Подржава угњеждене функције

Програмски језик Паскал (енгл. Pascal) подржава угњеждене функција (дефинисање функције унутар друге функције), али и поред тога што је Паскал утицао на готово све касније програмске језике када су у питању елементи структурираног и процедуралног програмирања, многи програмски језици немају подршку за угњеждене функције. Угњеждене функције могу бити корисне код рекурзивних алгоритама, јер дају елегантну могућност издвајања рекурзивне функције унутар главне функције, као што видимо у примеру бинарног претраживања[8]:

7 Од верзије 3.0 програмског језика Пајтон, оператор „/“ за целобројне операнде даје резултат у покретном зарезу, док су у ранијим верзијама оператори „/“ и „//“ за целобројне операнде радили исто.

8 За опис алгоритма бинарног претраживања можете погледати поглавље Низови у онлине материјалима проф. Цветане Крстевм, http://poincare.matf.bg.ac.rs/~cvetana/Nastava/Materijal/NizoviListe_1213.pdf

```
def trazi(lista, trazeno):
    def traziRek(levi, desni):
        if desni < levi:
            return -1
        srednji = (levi + desni)
// 2
        if trazeno >
lista[srednji]:
            return
traziRek(srednji + 1, desni)
        elif trazeno <
lista[srednji]:
            return
traziRek(levi, srednji - 1)
        else:
            return srednji


    return traziRek(0,
len(lista) - 1)
```

Да функција *traziRek* није угњеждена у функцију *trazi*, било би потребно да се функцији *traziRek* додају параметри *lista* и *trazeno*, што би смањило прегледност кода. Треба имати у виду да може бити и више од две вредности којима треба стално приступати из рекурзивне функције. Уз помоћ објектно-оријентисаног програмирања можемо дефинисати класу са рекурзивном методом, где се овакве вредности чувају у објекту, али је то мање елегантно и тера нас да ученика упознајемо са концептима објектно-оријентисаног програмирања само да бисмо исказали рекурзиван алгоритам.

9. Пајтон подржава и процедурално и објектно-оријентисано и фунцкијско програмирање

Пајтон омогућава да у програм укључите концепте који су вам потребни, а да вас остали не оптерећују. Лако можете програмирати чисто процедурално, без дефинисања класа и метода, што се могло видети у претходним примерима. У програмским језицима Јава и *C#* морате дефинисати барем једну класу у програму и не можете дефинисати функцију осим као статичку методу.

То не значи да Пајтон не подржава објектно оријентисано програмирање; шта више, свака вредност је објекат. На пример, оператор *a+b* се увек своди на позив методе *a.__add__(b)*, укључујући и уграђене типове, што значи да следеће две линије кода рачунају збир бројева 7 и 9:

```
>>> a, b = 7, 9
>>> a.__add__(b)
16
```

У Пајтону се методе дефинишу као функције унутар класе, при чему је први параметар резервисан за објекат (инстанцу класе) над којим се метод позива, а променљиве инстанце се динамички формирају при извршавању метода, најчешће у оквиру конструктора:

```
import math
class tacka:
    def __init__(self,x,y):
        self.x = x
        self.y = y

    def rastojanje(self, t):
        return math.
sqrt((self.x-t.x)**2 + (self.y-
t.y)**2)

a = tacka(1,4)
b = tacka(5,2)
print(a.rastojanje(b))
```

Претходни програм исписује *4.47213595499958*.

Пајтон подржава и основне елементе функцијског програмирања, као што су функције првог реда, ламбда изрази

и лексичка затварања[9] (енгл. *closures*, угњеждене функције имају својства лексичких затварања). Пајтон је погодан да се илуструју елементи функцијског програмирања и искористе у решевању проблема, мада није у целини дизајниран као функцијски језик. Класичан пример функцијског програмирања, где се примењује функција на све елементе листе и тако добија нова листа, изгледа овако:

```
>>> list(map(lambda x: 2*x,
[1,2,3]))
[2, 4, 6]
```

За израз попут претходног Пајтон има посебну синтаксу која је слична математичкој нотацији за конструисање скупова, као на пример $\{2x \mid x \in \{1, 2, 3\}\}$, па је разумљива и у случају да претходно нисте ништа чули о функцијском програмирању:

```
>>> [2*x for x in [1,2,3]]
[2, 4, 6]
```

Концепти објектно-оријентисаног програмирања нису неопходни на првим корацима у програмирању и могу се оставити за касније фазе учења програмирања. Пајтон омогућава да прве кораке у програмирању започнете са процедуралним програмирањем уз могућност да укључите по нешто од функцијског програмирања што неће оптеретити ученика, већ ће само учинити примере јаснијим. Објектно-оријентисано програмирање можете укључити у тренутку када сматрате да је погодно.

10. Подржава класичан текстуални улаз и излаз и елементарну дводимензионалну графику

Примери и задаци који се раде приликом учења програмирања често имају једноставну комуникацију са корисником: треба да се учитају неке улазне вредности и да се испишу неки резултати. За то је најједноставније користити класичан текстуални улаз и излаз. Пајтон у стандардној библиотеци има добро покривену подршку за текстуални улаз и излаз, што можемо илустровати примером:

```
a = float(input("Unesi dužinu
komada drveta: "))
b = float(input("Unesi dužinu
daske: "))
print("{0:.0f} dasaka i komad od
{1:.2f}".format(a//b, a%b))
```

Извршавање претходног програма би на конзоли изгледало овако:

```
Unesi dužinu komada drveta: 4
Unesi dužinu daske: 0.7
5 dasaka i komad od 0.50
```

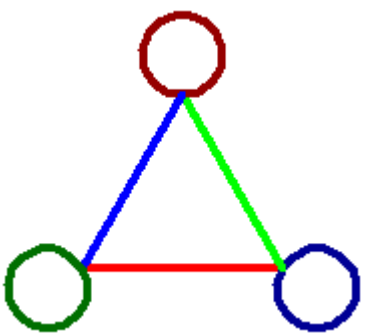
Поред текстуалне комуникације са корисником, за учење програмирања је згодно користити и примере са цртањем. Пајтон у стандардној библиотеци има подршку за такозвану „корњача графику“ (енгл. *turtle graphics*) познату из програмског језика *LOGO*. Овај метод програмирања графике заснива се на метафори кретања објекта који оставља траг, за шта је првобитно коришћен лик корњаче, од чега потиче назив. Суштина је да се без познавања правоуглог координатног система и тригонометријских функција, а са једноставним програмским конструкцијама, могу нацртати занимљиви геометријски облици. На пример:

```
import turtle
koja = turtle.Turtle()
koja.pensize(4)
for boja1,boja2 in
[(„red","darkblue"),
(„green","darkred"),
```

9 http://en.wikipedia.org/wiki/Closure_(computer_science)

```
                              („blue”,
„darkgreen”)]:
      koja.color(boja1)
      koja.forward(100)
      koja.right(120)
      koja.color(boja2)
      koja.circle(20)
      koja.left(240)
```

даје резултат:



За учење програмирања се често користи и библиотека PyGame[10] која има додатну подршку за дводимензионално (*2D*) цртање и *2D* игре.

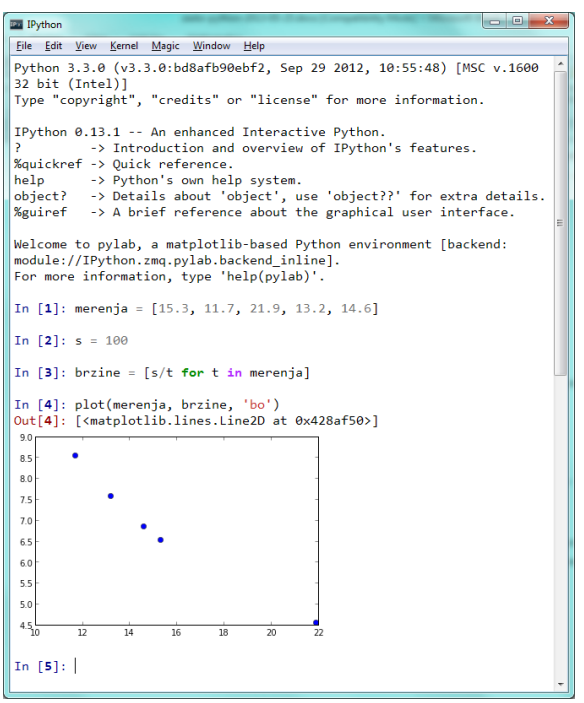
Напомињемо и да програм у програмском језику Пајтон не мора да се определи да ли ће бити конзолски или графички, већ просто може нешто да учита са конзоле, а после нешто да нацрта. За цртање преко модула *turtlemуртле* ће се аутоматски отворити посебан прозор.

11. Богата стандардна библиотека, велика подршка додатних библиотека и алата

Богате библиотеке шире спектар проблема који се могу решавати уз релативно једноставно програмирање. Ученици могу лако да цртају графиконе, тродимензионалне (*3D*) сцене, обрађују фотографије или управљају роботима.

10 http://www.pygame.org/

Уз основну инсталацију Пајтона долази интегрисано развојно окружење *IDLE* које је довољно за увод у програмирање, а можете се определити и за неко напредније развојно окружење. Подршку за Пајтон имају практично сва озбиљнија развојна окружења која подржавају више протрамских језика, укључујући два генерално најпознатија развојна окружења: *Microsoft Visual Studio* кроз додатак *Python Tools for Visual Studio*[11] и *Eclipse* кроз додатак *PyDev*[12]. Уколико желите да користите неко компактније развојно окружење које је прављено само за Пајтон, можете пробати *PyScripter*[13] ili *Spyder*[14]. У понуди су и комерцијална развојна окружења специјализована за Пајтон попут *JetBrains PyCharm*[15].



11 http://pytools.codeplex.com/
12 http://pydev.org/
13 https://code.google.com/p/pyscripter/
14 https://code.google.com/p/spyderlib/
15 http://www.jetbrains.com/pycharm/

Уколико вам је потребан унапређен интерактиван рад, уз интеграцију са библиотеком за цртање графикона, на располагању вам је *IPython*[16]. На слици 1 са леве стране је приказана *IPython Qt Console*[17], а оно што се налази на слици је пример из поглавља *Uspešno zamenjuje kalkulator*, при чему је додата још једна линија кода за исцртавање графикона.

За програмирање графичке корисничке сумеђе (*GUI*) уз основну инсталацију Пајтона долази библиотека *Tkinter*, а може се користити и подршка за неколико вишеплатформских GUI библиотека, као што су wxWidgets[18] и Qt[19].

За основно *3D* цртање погодна је библиотека VPython[20] која је врло једноставна за коришћење.

Пајтон је интегрисан и у многе апликације, као што је одличан софтвер за *3D* моделовање *Blender*[21] који је бесплатан и отвореног је кода. За веб програмирање је на располагању више платформи заснованих на програмском језику Пајтон, а најпознатија је *Django*[22].

Додатне библиотеке се обично морају инсталирати преко основне инсталације Пајтона да бисте били у могућности да их користите. Процедура инсталације је најчешће једноставна, а ако ипак не желите да инсталирате једну по једну додатну библиотеку, можете се определити и за неку од независних дистрибуција Пајтона poput *WinPython*[23] или *Python(x,y)*[24], а ако користите *Linux* проверите које библиотеке ваша дистрибуција *Linix-a* већ обухвата.

12. Широко се користи за увод у програмирање

Пајтон је чест избор првог програмског језика и за врло младе узрасте и за почетне курсеве на универзитетима. Као пример ћемо навести курс *Introduction to Computer Science and Programming*[25] са *Massachusetts Institute of Technology (MIT)* за који су слободно доступни снимци свих предавања и сви пратећи материјали. Листу универзитета на којима се користи Пајтон можете наћи на веб страни *The Python Wiki - Schools using Python*[26].

Аутори многих уводних књига за учење програмирања намењених деци опредељују се за Пајтон, а пример су књиге *Python for Kids*[27] ili *Hello World! Computer Programming for Kids and Other Beginners*[28], као и бесплатне књиге у електронском издању попут *Invent Your Own Computer Games with Python*[29].

За учење програмирања кроз програмирање игара треба поменути и књигу *Program Arcade Games with Python and Pygame*[30] са слободно доступном HTML верзијом.

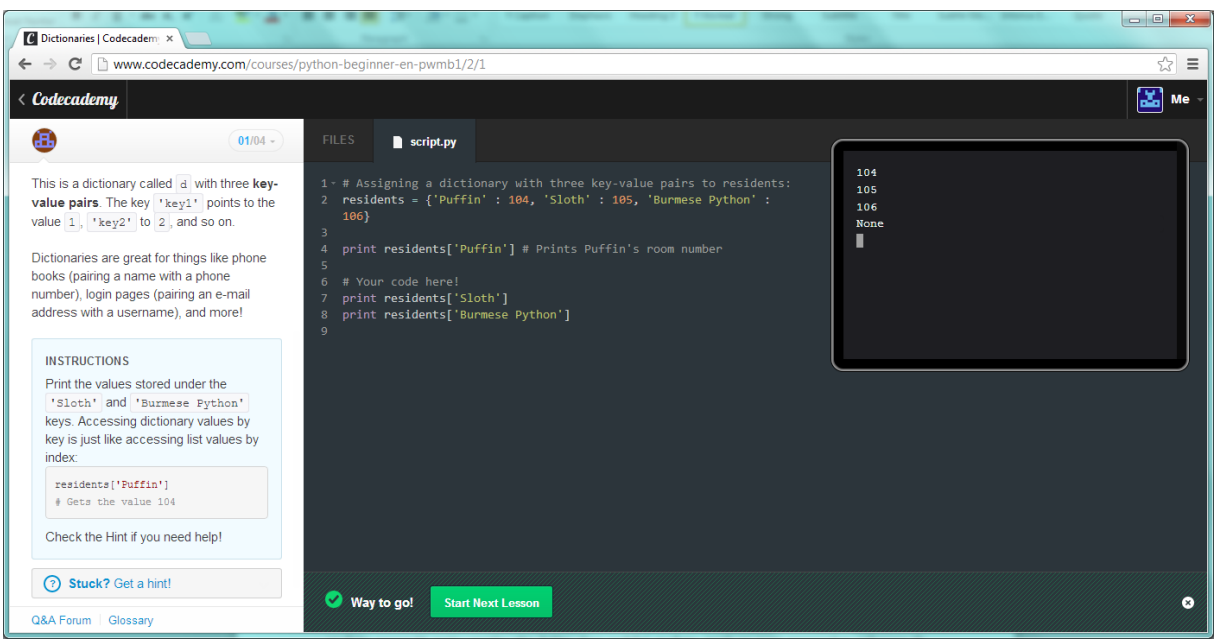
На крају ћемо поменути и књигу *Think Python: How to Think Like a Computer Scientist*[31] која је објављена под лиценцом *Creative Commons Attribution-NonCommercial*,

16 http://ipython.org/
17 http://ipython.org/ipython-doc/dev/interactive/qtconsole.html
18 http://www.wxwidgets.org/, подршка за Пајтон: http://wxpython.org/
19 http://qt-project.org/, подршка за Пајтон: http://qt-project.org/wiki/PySide или http://www.riverbankcomputing.com/software/pyqt/
20 http://www.vpython.org/
21 http://www.blender.org/
22 https://www.djangoproject.com/
23 https://code.google.com/p/winpython/
24 https://code.google.com/p/pythonxy/
25 http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/
26 http://wiki.python.org/moin/SchoolsUsingPython
27 http://jasonrbriggs.com/python-for-kids/
28 http://www.manning.com/sande/
29 http://inventwithpython.com/
30 http://programarcadegames.com/
31 http://www.greenteapress.com/thinkpython/

што значи да можете да је преправљате и прилагођавате, докле год наводите оригиналног аутора и преправљени производ не користите за комерцијалне сврхе.

13. Успешно се интегрише у интерактивне онлајн садржаје

Поред тога што основна имплементација Пајтона подржава разне оперативне системе, постоји и имплементација која се извршава



Слика 2. Изглед екрана у једној од лекција за Пајтон на ***Codeacademy***

унутар веб прегледача. На сајту *CodeSculptor*[32] можете у веб прегледачу извршавати Пајтон програме. *Rice*[33] универзитет користи *CodeSculptor* у свом курсу *An Introduction to Interactive Programming in Python* који је доступан и онлине на *Coursera*[34] платформи.

Захваљујући овој и другим технологијама које омогућавају интеграцију програмског језика Пајтон у онлајн системе, Пајтон је постао један од најбоље подржаних програмских језика у интерактивним онлине едукативним садржајима.

Уколико до сада то нисте урадили, погледајте лекције Пајтона на *Codeacademy*[35]. Овај онлајн туторијал вам омогућава да куцате одговоре на питања и примере и да извршавате кôд директно у веб прегледачу (Слика 2). Довољно је да одете на сајт, ништа не морате да инсталирате нити да копирате или прекуцавате кôд из примера, већ само треба да пратите инструкције, допуњујете примере који се појављују директно у едитору и дајете одговоре, а систем ће вас исправљати ако нешто нисте добро урадили.

Треба истаћи и издање *How to Think Like a Computer Scientist: Interactive Edition*[36]. Поред тога што имамо уводне видео клипове за одређене теме, примери у Пајтону се могу

32 http://www.codeskulptor.org/
33 http://www.rice.edu/
34 https://www.coursera.org/
35 http://www.codecademy.com/tracks/python
36 http://interactivepython.org/

извршавати директно на веб страни, мењати и снимати промене. То укључује и примере са графиком, а у појединим случајевима је дата анимација са извршавањем корак по корак и приказивањем међурезултата. Ово је комбинација квалитетног уџбеника за увод у рачунарство и најмодерније интерактивне технологије, а уз то је садржај потпуно отворен - осим текста уџбеника и цела технолошка платформа је отвореног кода - тако да можете ако желите направити своју верзију лекција или потпуно нове лекције на истој платформи. Ово интерактивно издање можете користити преко свог персоналог рачунара или таблета, а у крајњој линији и са мобилног телефона, мада је за куцање програма ипак практичнија тастатура.

Интерактивни онлајн садржаји се могу користити и за самостално учење и за рад на часу, где под надзором наставника ученици пролазе садржај, али и за усавршавање и припрему наставника.

14. Пајтон помаже да се усресредите на учење програмирања, а не програмског језика

Када се сумирају сви изнети аргументи, најкраће што се може рећи је: Пајтон вас не оптерећује. У учењу програмирања мање ћете се бавити програмским језиком, а више алгоритмима и концептима које треба разумети.

У претходним деценијама су се као најчешћи избор за увод у програмирање истицали разни програмски језици у разним периодима, а међу њима можемо посебно издвојити Паскал, *Basic* і *Logo*. Можемо рећи да Пајтон обухвата главне добре особине сваког од ова три програмска језика, када је у питању учење програмирања:

- Пајтон омогућава елегантно исказивање алгоритма у самом програмског коду, чиме се поноси и Паскал;
- једноставна синтакса и искоришћене предности језика који се интерпретира су разлози због којих је *Basic* својевремено постао популаран за учење програмирања;
- учење програмирања у програмском језику *Logo* се заснива на примерима и задацима цртања геометријских облика помоћу „корњаче“ уз једноставне синтаксне конструкције, што је једнако изводљиво и у Пајтону.

Поред наведеног Пајтон подиже ниво апстракције увођењем основних типова колекција као уграђених типова и многим другим детаљима, тако да се у уводу у програмирање може отићи корак ближе решавању интересантних проблема.

Разумевање основних концепата програмирања је корисно и за оне који неће бити програмери, а та врста општеобразовног знања улази у оквир нечега што се назива информатичко размишљање (енгл. *computational thinking*) (Wing 2006) и може се дефинисати:

„Информатичко размишљање је мисаони процес који се користи у формулисању проблема и њихових решења тако да је решење представљено у облику у коме се може ефективно спровести средством обраде података.“ [37]

Општеобразовни приступ учењу програмирања води нас ка томе да се са што мање програмирања решавају што занимљивији проблеми, а Пајтон се са својом разумљивом синтаксом, логичношћу, богатом стандардном библиотеком и додатним алатима одлично уклапа у овакав приступ.

37 Jan Cuny, Larry Snyder and Jeannette M. Wing, “Demystifying Computational Thinking for Non-Computer Scientists,” у припреми , 2010

Литература

Крстев, Цветана. 2012. Алгоритми и програмирање. http://poincare.matf.bg.ac.rs/~cvetana/Nastava/Materijal/Algoritmi_1213.pdf (приступљено 18. IX 2013)

Крстев, Цветана. 2012. Програмирање – део 3. http://poincare.matf.bg.ac.rs/~cvetana/Nastava/Materijal/NizoviListe_1213.pdf (приступљено 18. IX 2013)

Miller, B., and Ranum D. *How to Think Like a Computer Scientist: Interactive Edition*. http://interactivepython.org/runestone/static/thinkcspy/toc.html (приступљено 18. IX 2013)

Pérez, F., and Granger, B. E. 2007. IPython: A System for Interactive Scientific Computing, *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21-29. http://ipython.org (приступљено 18. IX 2013)

van Rossum, G., and Drake, F. Jr. 2011. *An Introduction to Python - The Python Tutorial*. Bristol: Network Theory Ltd.

van Rossum, G., and Drake, F. Jr. 2011. *The Python Language Reference Manual*. Bristol: Network Theory Ltd.

Wikipedia. Closure (computer science). http://en.wikipedia.org/wiki/Closure_(computer_science) (приступљено 18. IX 2013)

Wikipedia. Measuring programming language popularity. http://en.wikipedia.org/wiki/Measuring_programming_language_popularity (приступљено 18. IX 2013)

Wing, Jeannette M. 2006. Computational thinking. *Commun. ACM* 49 (3) : 33-35. http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf (приступљено 18. IX 2013)

Веб локације

Blender project – Free and Open 3D creation software. http://www.blender.org/ (приступљено 18. IX 2013)

Codeacademy Python tutorials. http://www.codecademy.com/tracks/python (приступљено 18. IX 2013)

CodeSculptor. A browser-based Python interpreter. http://www.codeskulptor.org/ (приступљено 18. IX 2013)

Coursera online courses. https://www.coursera.org/ (приступљено 18. IX 2013)

Scratch – Imagine, Program, Share. http://scratch.mit.edu/ (приступљено 18. IX 2013)

Django. A high-level Python Web framework. https://www.djangoproject.com/ (приступљено 18. IX 2013)

IPython Interactive Computing. http://ipython.org/ (приступљено 18. IX 2013)

IPython Qt Console. http://ipython.org/ipython-doc/dev/interactive/qtconsole.html (приступљено 18. IX 2013)

JetBrains PyCharm. Python IDE & Django IDE for Web developers. http://www.jetbrains.com/pycharm/ (приступљено 18. IX 2013)

MIT OpenCourseWare – Introduction to Computer Science and Programming. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/ (приступљено 18. IX 2013)

Qt Project. A cross-platform application and UI framework. http://qt-project.org/ (приступљено 18. IX 2013)

PyDev. A Python IDE for Eclipse. http://pydev.org/ (приступљено 18. IX 2013)

PyScripter. An open-source Python IDE. https://code.google.com/p/pyscripter/ (приступљено 18. IX 2013)

Python(x,y). Scientific-oriented Python Distribution based on Qt and Spyder. https://code.google.com/p/pythonxy/ (приступљено 18. IX 2013)

Python Programming Language – Official Website.

http://www.python.org/ (приступљено 18. IX 2013)

Python Tools for Visual Studio. http://pytools.codeplex.com/ (приступљено 18. IX 2013)

The Python Wiki – Schools using Python. http://wiki.python.org/moin/SchoolsUsingPython (приступљено 18. IX 2013)

Rice University. http://www.rice.edu/ (приступљено 18. IX 2013)

Spyder. Scientific PYthon Development EnviRonment. https://code.google.com/p/spyderlib/ (приступљено 18. IX 2013)

VPython. 3D Programming for Ordinary Mortals. http://www.vpython.org/ (приступљено 18. IX 2013)

WinPython. Portable Scientific Python 2/3 32/64bit Distribution for Windows. https://code.google.com/p/winpython/ (приступљено 18. IX 2013)

wxWidgets. Cross-Platform GUI Library. http://www.wxwidgets.org/ (приступљено 18. IX 2013)

Примљено: 16. V 2013.
Прихваћено: 24. VI 2013.