Python

Literatura: https://petlja.org/BubbleBee/r/kursevi/interactive-book-py6

Инсталација

- Инсталацију можете преузети са http://www.python.org/downloads/ и препоручујемо да користите развојно окружење IDLE које је саставни део имплементације коју сте инсталирали.
- Обратите пажњу да треба да инсталирате верзију 3, а не верзију 2.

- Програмирање у Пајтону, приручник за шести разред је дело аутора Филипа Марића и ауторског тима Фондације Петља. Овај приручник се даје на коришћење под лиценцом Creative Commons Attribution 4.0 International License.

1. Основне аритметичке операције и примена

- Рачунар или компјутер (енгл. computer) је справа која рачуна тј. справа која је направљена тако да може веома брзо и ефикасно да изводи рачунске операције над бројевима.
- Рачунање се назива и аритметика (од грчке речи ἀριθμός тј. аритмос која значи број, бројање, рачунање), а рачунске операције се називају и аритметичке операције.

Сабирање, одузимање, множење

- Основна аритметичка операција је **сабирање**. Збир бројева 3 и 5 се у математици представља као *3 + 5*. У програмском језику Python користи се готово идентичан запис 3 + 5.
- Поред сабирања можемо разматрати **одузимање**. Разлика бројева 8 и 2 се у математици представља као *8 - 2*. У програмском језику Python користи се готово идентичан запис 8 – 2.
- Још једна од основних операција је и **множење**. Производ бројева 4 и 6 се у математици представља као *4 · 6*. У програмском језику Python множење се означава помоћу оператора * и производ бројева 4 и 6 се записује као

  4 * 6.

* Програмски језик Python, наравно, уме и да дели, да израчунава остатак при дељењу и цео део количника и много штошта друго. О овим операцијама ћемо говорити на неком од наредних часова.

Сабирање, одузимање, множење

Ако на свом рачунару покренеш интерпретатор за програмски језик Python, вредност неког израза (на пример, `3 + 5` или `4 * 6`) можеш израчунати тако што тај израз просто укуцаш (иза знакова `>>>`) и притиснеш тастер Enter. На пример,

```
Python 3.4.3 (default, Mar 26 2015, 22:03:40)
[GCC 4.9.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 3 + 5
8
>>> 4 * 6
24
```

Да би се одштампала вредност неког израза, рачунару мораш некако рећи да он ту вредност одштампа. *Штампај* се на енглеском језику каже *print*, па се онда вредност израза може добити на следећи начин.

```
1 print(3 + 5)
2 print(4 * 6)
3
```

```
8
24
```

Сложени изрази, приоритет оператора и заграде

- Некада је потребно да решавамо задатке који укључују више рачунских операција и тада можемо користити сложеније изразе, потпуно исто како смо навикли у математици. На пример, посматрајмо следећи задатак који је преузет из једне збирке задатака из математике.

Израчунај производ збира бројева 874 и 437 и разлике бројева 915 и 364.

- У математици би се одговарајући израз записао као *(874+437)·(915-364)*. Ако се сетиш да се множење изражава знаком * и ако ти кажемо да у програмском језику Python можеш употребљавати заграде на исти начин као у математици, онда ти је јасно да претходни математички задатак можеш лако решити тако што употребиш израз (874 + 437) * (915 - 364), тј. наредни програм: print((874 + 437) * (915 - 364))

Променљиве - имена међурезултата

- Писање сложених израза се може избећи, а програм се може начинити мало разумљивијим ако међурезултате именујемо. Погледајмо наредни пример програма који такође решава претходни задатак.

```
1 zbir = 874 + 437
2 razlika = 915 - 364
3 proizvod = zbir * razlika
4 print(proizvod)
5
```

- Имена која смо дали међурезултатима се у програмирању називају **променљиве**.
- Променљиве су јако важан концепт о коме ће бити много више речи касније. До тада ћемо их користити на потпуно исти начин у математици - само као имена придружена одређеним вредностима.

Променљиве - имена међурезултата

Збиру смо доделили име `zbir`, разлици име `razlika`, а производу име `proizvod` (уместо `zbir` и `razlika` могли смо, на пример, користити и имена `prvi_cinilac`, `drugi_cinilac`). Иако се на овај начин добија програм који мало дужи него полазни, он је мало разумљивији, јер се његовим читањем може јасно видети да се прво тражи израчунавање збира, затим разлике и затим њиховог производа. То се десило пре свега захваљујући пажљивом одабиру имена која смо употребили и веома је важно у програмима користити имена која читаоцу програма дају назнаке шта тај програм заправо израчунава. Рачунар једнако успешно извршава програм, ма која имена да употребиш. Ипак, имај на уму да програме читају и људи који те програме пишу, исправљају и дорађују, а њима је прилично важно да текст програма лако разумеју. У већини случајева си читалац програма управо ти, тако да давањем илустративних имена променљивима данас помажеш заправо себи у будућности.

Променљиве - имена међурезултата

- Имена која користимо смеју да садрже само слова, цифре и подвлаке (знак _) - не смеју да садрже размаке, цртице ни остале интерпункцијске знаке.

Рецимо да постоје и нека правила која се односе на имена (каже се и *идентификаторе*) која можемо користити. Прво, постоји разлика између великих и малих слова и није исто да ли смо употребили `zbir` или `Zbir`. Препоручује се да у именима користимо само слова енглеске абецеде (тзв. ошишану латиницу), бројеве и подвлака (симбол `_`) који ћемо користити да повежемо више речи у једно име. У именима не можемо користити размаке, зарезе и слично, нити име можемо започети цифром. Дозвољена имена су, на пример, `x`, `obim`, `drugi_sabirak`, `broj_sekundi`, `a2`, а недозвољена су, на пример, `3d_grafika` (јер почиње цифром), `prvi sabirak` (јер садржи размак) и `jezik_c#` (јер садржи недозвољени знак `#`).

Задатак 1

Смедеревска тврђава

Смедеревска тврђава има облик троугла страница 550m, 502m и 400m. Колики је обим тврђаве (када шеташ око тврђаве, колико ћеш метара прећи)?

Задатак 2

Воћњак са јабукама

Пера је засадио 380 стабала јабуке. Ђура је засадио 142 стабла јабука више од Пере, а Мика је засадио два пута више од Пере. Колико су стабала засадили заједно?

Задатак 3

Река Морава

Велика Морава је дугачка 185km и настаје од Јужне Мораве која је 90km дужа, и Западне Мораве која је 123km дужа од ње. Колика је укупна дужина ове три реке?

Напредно коришћење функције *print*

Немањићи

Стефан Немањић је постао краљ Србије 1217 и владао је 11 година. После њега је Радослав владао до 1234. година, па Владислав који је владао 9 година и предао престо брату Урошу Првом који је владао до 1276. У којим временским периодима су владали ови српски краљеви?

Напредно коришћење функције *print*

```
1  Stefan_pocetak = 1217
2  Stefan_kraj = 1217 + 11
3  Radoslav_pocetak = Stefan_kraj
4  Radoslav_kraj = 1234
5  Vladislav_pocetak = 0
6  Vladislav_kraj = 0
7  Uros_pocetak = 0
8  Uros_kraj = 0
9  print("Стефан:", Stefan_pocetak, "-", Stefan_kraj)
10 print("Радослав:", Radoslav_pocetak, "-", Radoslav_kraj)
11 print("Владислав:", Vladislav_pocetak, "-", Vladislav_kraj)
12 print("Урош:", Uros_pocetak, "-", Uros_kraj)
13
```

Напредно коришћење функције *print*

- Приметимо да смо у претходном задатку додали испис имена краљева и цртица између почетка и краја њихове владавине, тако што смо тај текст који смо желели да се испише ставили под знаке навода (нпр. навели смо “Стефан: ”). О раду са текстом ће више речи бити касније.
- Такође, приметимо да смо овај пут навели неколико ствари унутар *print*, раздвојених зарезима. У тим ситуацијама *print* штампа сваку од њих, развајајући их размацима (на пример, када се изврши нареба **print(“baci”,5)** ispisuje se **baci 5**).

Напредно коришћење функције *print*

Као што смо рекли, ствари наведене унутар `print` раздвајају се са по једним размаком. То се може променити тако што се на крају `print` наведе `sep=""` и унутар наводника наведе шта ће се користити да раздвоји делове. На пример, ако се наведе `print(1, 2, 3, sep="")` исписаће се `123`, а ако се наведе `print(1, 2, 3, sep=", ")` исписаће се `1, 2, 3`. Након сваког извршавања `print` прелази се у нови ред (наредни позиви `print` штампаће свој резултат у наредном реду). И то се може променити тако што се на крају `print` наведе `end=""` и унутар наводника оно што ће се користити након целог исписа. На пример, `print(1, 2, end="")` проузрокује да се након исписа не пређе у нови ред, већ да наредни испис иде непосредно након вредности `2`.

Цели и реални бројеви

До сада смо у задацима користили само природне бројеве. Језик Python подржава и рад са целим бројевима и они се записују на исти начин као у математици. На пример, вредност израза `3 - 8` је `-5` док је вредност израза `(-3) - (-8)` број `5`.

Реалне бројеве је такође веома једноставно користити, једино што се уместо децималног зареза на који смо навикли у математици мора користити децимална тачка. Тако се, на пример, број *2,5* записује као `2.5`.

Куповина намирница

Марко је купио 0,45 kg саламе, 0,25 kg сира и два паковања од по 0,3 kg шунке. Колико је тешка кеса коју носи кући?

Покрени програм | Корак по корак

```
1 salama = 0.45
2 sir = 0.25
3 sunka = 0.3
4 print(salama + sir + 2 * sunka)
5
```

Цели и реални бројеви - задатак

Банковни рачун

Ђура је уплатио летовање пре него што је добио плату и ушао је у тзв. дозвољени минус тј. након те уплате дуговао је банци 12.376,5 динара. Три дана касније на рачун му је уплаћена плата од 43.386,9 динара. Колико му је тада било стање на рачуну.

Допуни наредни програм тако да коректно решава тражени задатак

Покрени програм | Корак по корак

```
1 stanje_pre =
2 uplata =
3 stanje_posle =
4 print(stanje_posle)
5
```