

ПРИПРЕМА ЧАСА РАЧУНАРСТВО И ИНФОРМАТИКА ЗА ШЕСТИ РАЗРЕД	
Наставна тема:	Рачунарство
Редни број часа:	11.
Наставна јединица:	Понављање
Тип часа:	обрада
Циљ часа:	Упознавање ученика са наредбама за организацију циклуса (наредбе понављања) у програму Пајтон.
Исходи часа:	<ul style="list-style-type: none"> <li>○ Ученик разуме наредбу понављања <code>for</code> и уме на правилан начин да је примени у Пајтон програмском окружењу</li> <li>○ Ученик уме да једну или више наредби понавља одређени број пута у Пајтон језику</li> <li>○ Ученик уме да користи бројачке променљиве</li> <li>○ Ученик разуме наредбу понављања <code>while</code> и уме на правилан начин да је примени у Пајтон програмском окружењу</li> </ul>
Облици рада:	фронтални, индивидуални, рад у пару
Наставне методе:	демонстративна, метода разговора, метода практичног рада
Место реализације часа:	рачунарски кабинет
Корелација:	
Литература:	<a href="https://petlja.org/biblioteka/r/kursevi/prirucnik-python">https://petlja.org/biblioteka/r/kursevi/prirucnik-python</a>

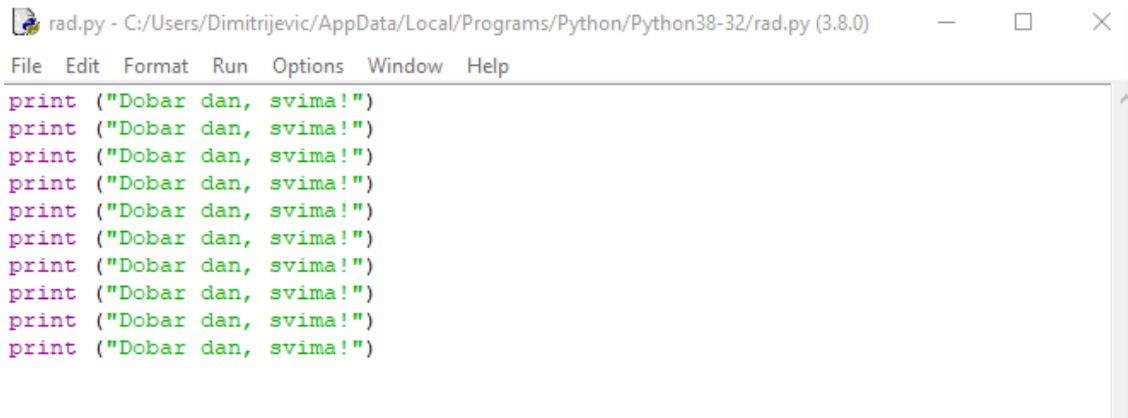
## ТОК ЧАСА

### УВОДНИ ДЕО ЧАСА (5 МИНУТА)

У уводном делу часа навести ученицима неке примере из свакодневног живота које се понављају одређени број пута при чему треба направити разлику случајева где знамо број понављања неке радње и где не знамо, односно где се нека радња понавља док важи неки услов. На пример, Милан није научио дефиницију, па је за домаћи мора преписати 20 пута и Милан је вежбао математику све док је није научио. Напоменути да су слично радили са Карелом и корњачом и да су се ту упознали са коришћењем петљи *for* и *while*, а да ће сада користити сличне наредбе у Python-у.

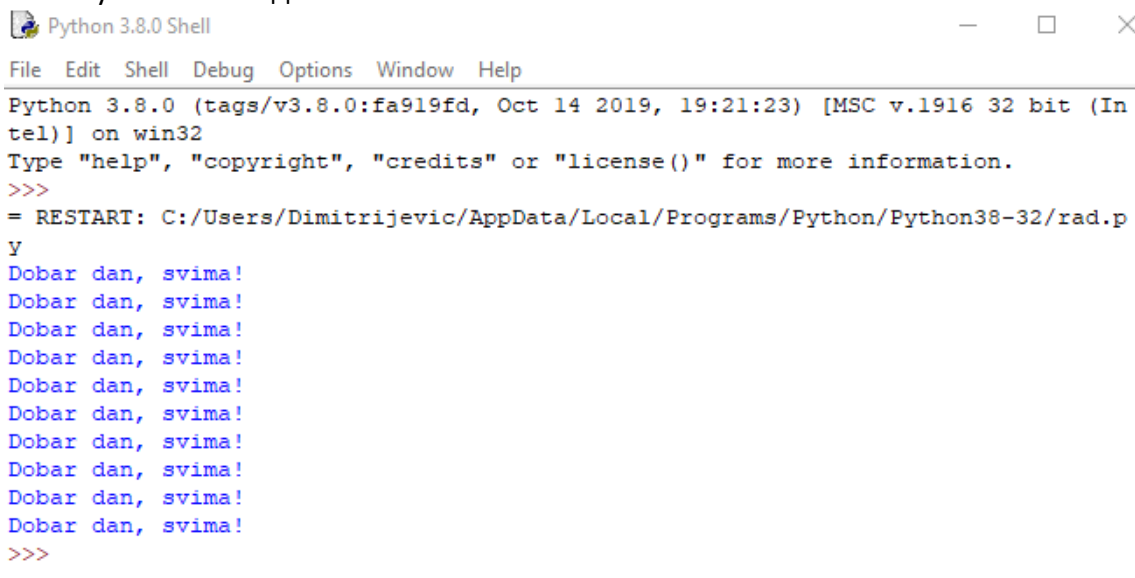
## ГЛАВНИ ДЕО ЧАСА (35 МИНУТА)

На почетку главног дела часа наставник демонстрира како би требало написати програм који одређени број пута на екрану испишује „Добар дан, свима!“. На основу онога што су ученици до сада научили, требало би само тражени број пута исписати наредбу `print ("Dobar dan, svima!")`.

A screenshot of a text editor window titled "rad.py - C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py (3.8.0)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code inside consists of ten lines, each starting with "print" followed by a double-quoted string "Dobar dan, svima!".

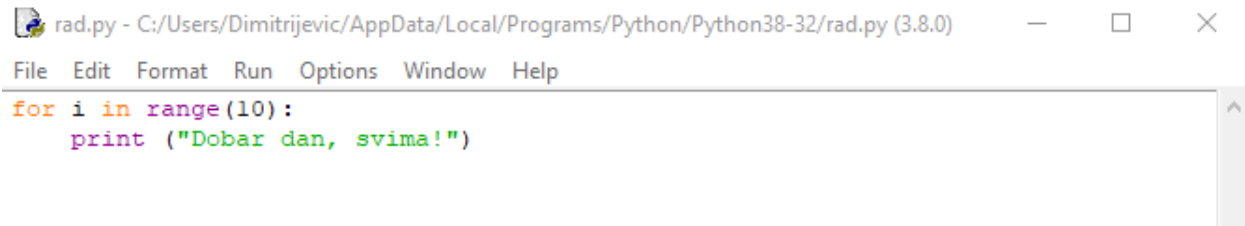
```
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
print ("Dobar dan, svima!")
```

Резултат овог кода биће:

A screenshot of a "Python 3.8.0 Shell" window. It shows the execution of the script. The first line is the Python version and build information. The second line is a prompt to type "help", "copyright", "credits" or "license()". The third line shows the script being restarted. The output consists of ten lines of "Dobar dan, svima!".

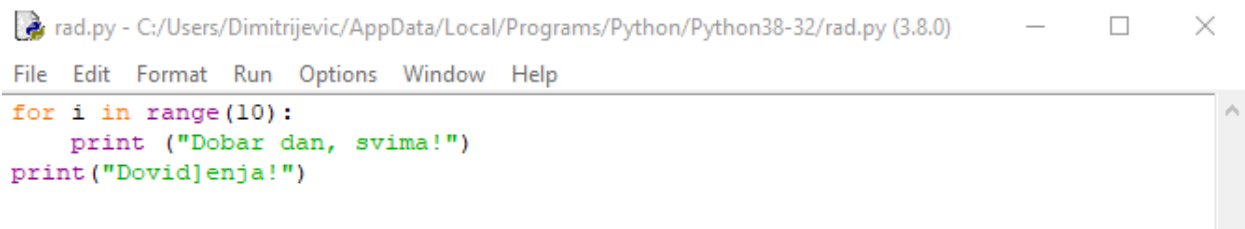
```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
>>>
```

Нагласити ученицима то да овај начин доводи до решења али да ипак није практично куцати исту ствар толики број пута, као и то да када би корисник требало да прво унесе колико пута жели да му се испише ова порука, не би било могуће направити програм на тај начин. За овакве проблеме се користе петље тј. наредбе које омогућавају да се нека наредба или неки блок наредби понове више пута. Ово је случај када је потребно да се нешто понови задати број пута. За тако нешто користи се наредба облика `for i in range(n):`. Наредни програм 10 пута испишује поруку `Dobar dan, svima!`, коришћењем петље `for`.



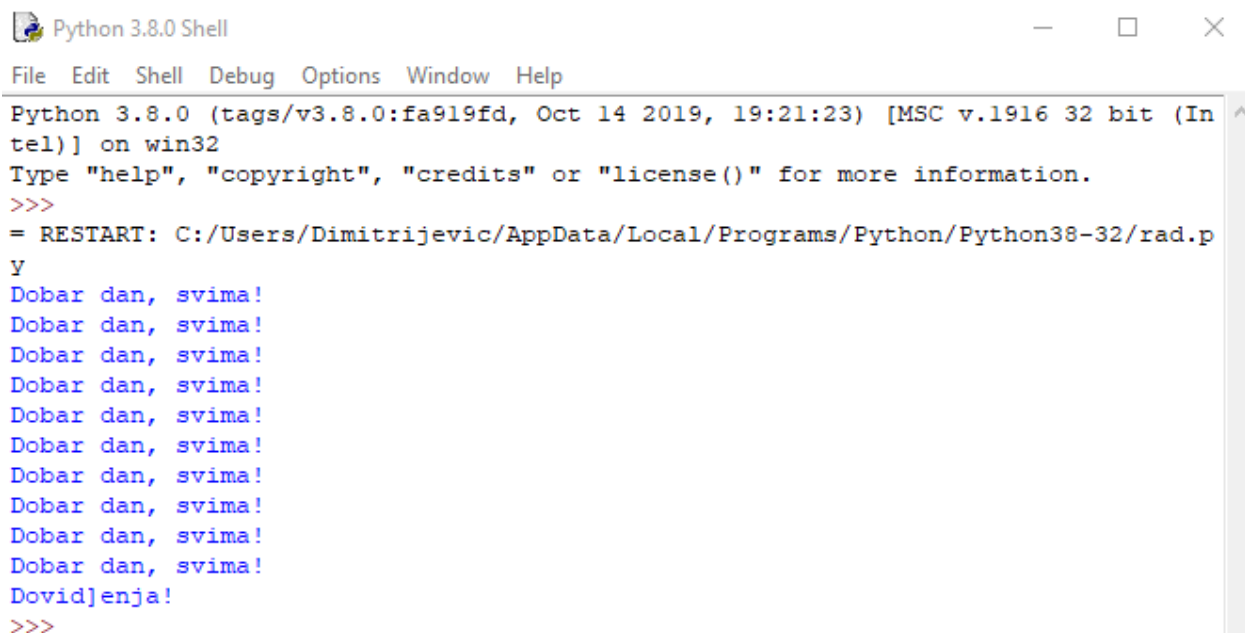
```
rad.py - C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py (3.8.0)
File Edit Format Run Options Window Help
for i in range(10):
    print ("Dobar dan, svima!")
```

Након петље може се исписати додатна порука **Dovidjenja!** на следећи начин.



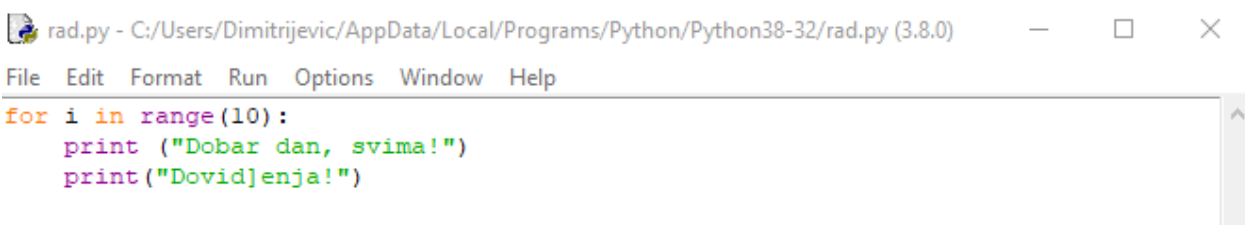
```
rad.py - C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py (3.8.0)
File Edit Format Run Options Window Help
for i in range(10):
    print ("Dobar dan, svima!")
print("Dovidjenja!")
```

Резултат извршења биће следећи.



```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dobar dan, svima!
Dovidjenja!
>>>
```

Приметити да је испис поруке **Dobar dan, svima!** увучен у коду, док испис поруке **Dovidjenja!** није био увучен што значи да се та наредба не понавља 10 пута као за прву поруку. Показати ученицима шта би се десило да је и наредба за испис друге поруке била увучена, тј. да нам код изгледа овако.

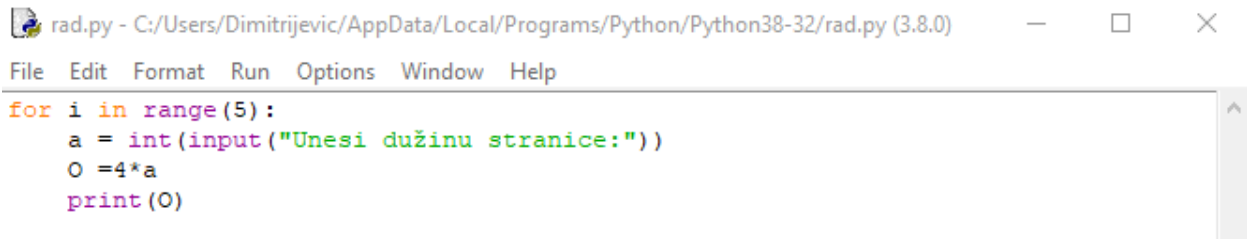


```
rad.py - C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py (3.8.0)
File Edit Format Run Options Window Help
for i in range(10):
    print ("Dobar dan, svima!")
    print("Dovidjenja!")
```

### Пример 1.

Написати програм који израчунава обиме пет различитих квадрата чије дужине страница уноси корисник.

Задатак самостално решавају ученици, док наставник помаже и на крају показује следеће решење.



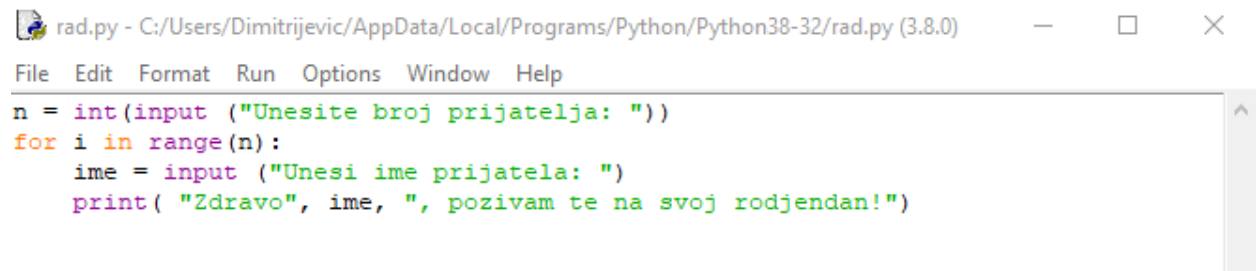
```
rad.py - C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py (3.8.0)
File Edit Format Run Options Window Help
for i in range(5):
    a = int(input("Unesi dužinu stranice:"))
    O = 4*a
    print(O)
```

Нагласити посебно да су све наредбе у телу петље увучене у односу на петљу, али и да су међусобно поравнате, што мора увек бити случај (у супротном се добија порука да програм није исправан).

### Пример 2.

Написати програм који од корисника тражи да унесе број пријатеља које ће позвати на рођендан и који свакоме појединачно исписује позив.

Задатак самостално решавају ученици, док наставник помаже и на крају показује следеће решење.



```
rad.py - C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py (3.8.0)
File Edit Format Run Options Window Help
n = int(input("Unesite broj prijatelja: "))
for i in range(n):
    ime = input("Unesi ime prijatelja: ")
    print("Zdravo", ime, ", pozivam te na svoj rođendan!")
```

Овде треба нагласити да уместо константне вредности унутар range треба навести променљиву n, што значи да ће се петља извршавати различит број пута (у зависности од броја n који корисник уноси). Такође, приметити да променљива ime не чува број него текст. Приликом уноса вредности употребили смо само `input(...)`, а не `int(input(...))`, нити `float(input())`, јер након уноса текста није потребно из њега прочитати ни целобројну ни реалну бројевну вредност (чему служе функције int и float).

Даље треба размотрити детаљније како функционише петља `for i in range(n):` којом смо постизали да се нешто понови n пута. Током извршавања такве петље променљива i редом узима вредности 0, 1, 2 итд., све до вредности n-1. На пример, ако је n једнако 3, тада ће се тело петље извршити три пута и у првом извршавању тела петље променљива i ће имати вредност 0, у другом вредност 1, а у трећем вредност 2. Позив `range(n)`, дакле, формира колекцију бројева 0, 1, 2, ..., n-1 из које затим бројачка променљива i узима редом једну по једну вредност.

Осим са једним, функцију **range** могуће је позвати и са два и са три параметра. Када се наведу два аргумента  $a$  и  $b$ , тј. позивом **range(a, b)**, врши се набрајање свих целих бројева од  $a$  до  $b-1$ , док се у случају када се наведу три аргумента  $a$ ,  $b$  и  $k$ , тј. позивом **range(a, b, k)** врши набрајање сваког  $k$ -тог елемента од  $a$  до  $b-1$ .

Дакле,

- позив **range(n)** гради колекцију  $1, 2, \dots, n-1$ ;
- позив **range(a, b)** гради колекцију  $a, a+1, \dots, b-1$ ;
- позив **range(a, b, k)** гради колекцију  $a, a+k, a+2k, \dots, a+nk$ , где је последњи број одабран тако да буде последњи у овом низу који је строго мањи од  $k$ .

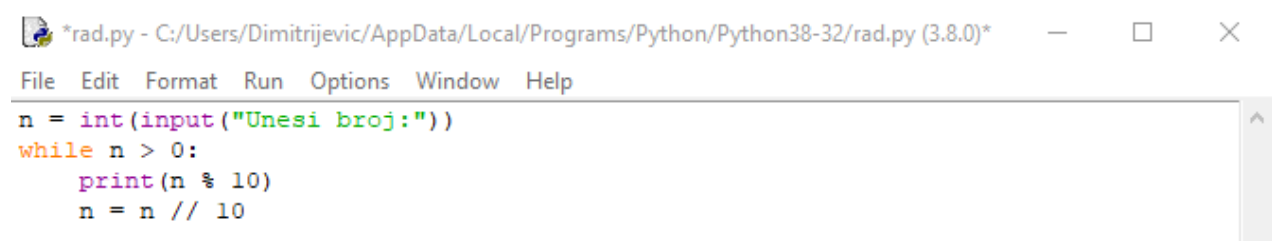
Уколико буде било довољно времена илустровати различите употребе позива **range**.

Након тога још једном нагласити да се петља **for** користи да се неке наредбе понове одређен, познат број пута, да се бројачком променљивом прође кроз неку правилан низ бројева (узастопне бројеве или бројеве набројане са неким кораком). Међутим, у неким ситуацијама је потребно богатије контролисање понављања, тј. потребно је неке наредбе понављати све док је испуњен неки услов. За то се користи тзв. условна петља тј. петља **while**.

Употребу **while** петље илустровати следећим примером.

### Пример 3.

Написати програм који одређује и исписује цифре датог природног броја (већег од 1), кренувши од цифре јединица. На пример, ако корисник унесе број 1234, програм треба да испише 4, 3, 2, 1 (сваку цифру у посебном реду).



```
*rad.py - C:/Users/Dimitrijevic/AppData/Local/Programs/Python/Python38-32/rad.py (3.8.0)*
File Edit Format Run Options Window Help
n = int(input("Unesi broj:"))
while n > 0:
    print(n % 10)
    n = n // 10
```

Овде наставник извршава програм корак по корак и прати вредности променљивих. На пример, ако се унесе број 123, приликом првог уласка у петљу проверава се да ли је  $123 > 0$  и пошто јесте, исписује се  $123 \% 10$  тј. 3, а број  $n$  се замењује са  $123 // 10$  тј. 12. Након тога се проверава да ли је  $12 > 0$  и пошто јесте, исписује се  $12 \% 10$  тј. 2, а број  $n$  се замењује са  $12 // 10$  тј. 1. Након тога се проверава да ли је  $1 > 0$  и пошто јесте, исписује се  $1 \% 10$  тј. 1, а број  $n$  се замењује са  $1 // 10$  тј. 0. На крају се проверава да ли је  $0 > 0$  и пошто није, петља се завршава. Дакле, програм исписује цифре 3, 2 и 1, баш како је и тражено.

### ЗАКЉУЧНИ ДЕО ЧАСА (5 МИНУТА)

У закључном делу часа укратко поновити најважније везано за петље, утврдити када се користи **for**, а када **while** петља. Разјаснити могуће нејасноће и оставити следеће задатке за домаћи.

1. Напиши програм који исписује све парне бројеве прве стотине.
2. Напиши програм који учитава два броја а и b и затим исписује све бројеве од а до b (укључујући и њих).
3. У току је распродаја ципела. Напиши програм који корисницима омогућава да израчунају снижену цену ципела које желе да купе. У петљи се уноси цена ципела и проценат снижења и исписује се снижена цена. Програм престаје са радом када се унесе цена 0.