Увод у

# Програмски језик *Python*

Александар Пајкановић

nanoluka.org
YouTube.com/nanoluka
instagram.com/nanolukaorg
twitter.com/nanolukaorg
github.com/nanoluka
nanolukaorg@gmail.com

Октобар 2020

# Кориштење овог документа

- **Лиценца**:
  - Документ и пратеће кôдове објављујем под лиценцом *Attribution-ShareAlike — CC BY-CA*. Ова лиценца дозвољава ремикс и прераду, као и комерцијално кориштење дјела, ако/док се **правилно назначава име аутора** и ако се **прерада лиценцира под истим условима**. Ова лиценца се често упоређује са „*copyleft*" лиценцирањем слободног софтвера или софтвера отвореног кôда. Сва дјела настала на основу дјела лиценцираног овом лиценцом, требало би да буду лиценцирана истом лиценцом, која, поред осталог, дозвољава комерцијално кориштење.
- **Навођење (цитирање)**:
  - Ако сте користили овај документ као извор у сопственим материјалима, молим да цитирате на сљедећи начин: А. Пајкановић, „Увод у програмски језик *C* кроз практичне примјере", доступно на github.com/nanoluka/python.git

# Програмирање и програмски језици

- Рачунар у општем смислу је машина способна да изврши било који алгоритам пратећи дефинисан скуп правила - ткз. Тјурингова машина (теоријска замисао).
- Алгоритам је недвосмислен метод рјешавања конкретног проблема
- Рачунар се већ читав вијек састоји од процесора и меморије - ткз. Фон Најманова архитектура (реализација)
- Програмирање, у практичном смислу, можемо рећи да представља састављање текстуалног (понегдје и графичког) описа жељеног понашања рачунара у смислу обраде корисничких података и представљања резултата тог процеса кориснику
- Програмски језик је скуп инструкција (кључних ријечи) које, ако су посложене у складу са унапријед дефинисаним правилима, рачунар може да прихвати - при чему ово „прихвати" значи да је у стању да, узев дате податке корисника на улазу, кориснику прикаже очекивани излаз

# Програмски језик *Python*

- Постоји много подјела програмских језика, најважнија је на:
  - језике вишег нивоа, и
  - језике нижег нивоа
- *Python* спада у језик вишег нивоа. Генерално, сви програмски језици за које сте чули су, у ствари, у истој групи.
- Друга важна подјела је према парадигми којој програмски језик припада, тренутно су актуелне четири парадигме (које се, често, међусобно пресјецају):
  - императивно програмирање,
  - функционално,
  - логичко, и
  - објектно-оријентисано.
- У нашем данашњем језику можемо да пишемо програме који припадају двјема парадигмама - функционалном и објектно-оријентисаном програмирању
- Ми, пак, данас нећемо користити ни једно ни друго

# Програмски језик *Python*

- Иначе, *Python* је 1991. објавио Гвидо ван Росум
- С обзиром да смо прошли пут причали о језику *C*, ево неколико особина *Python vs. C*:
  - лакше га је савладати, синтакса је корисницима читљивија
  - *Python* је интерпретирани језик, што значи да се једна по једна линија кôда преводе, док се *C* програм компајлира. Као посљедица, извршавање *Python* програма је спорије
  - иста чињеница узрокује лакше уочавање грешака
  - варијабле се не декларишу експлицитно, него имплицитно приликом прве употребе
  - нема показивача (напредна функционалност језика *C*, нисмо се тиме бавили у другом предавању, наводим само потпуности ради
  - аутоматско рјешавање потенцијалних проблема са меморијом
  - неупоредиво већа подршка најразличитијих библиотека и пакета у свим могућим доменима науке, технологије и живота уопште

# Програмски језик *Python*

- Укратко, поступак од идеје до реализације састоји се од четири корака:
  1. алгоритам,
  2. псеудо кôд,
  3. изворни кôд - овдје се пише синтакса (`.py`),
  4. и сад имамо три могућности:
     - ★ извршавање појединачних линија, директно из терминала,
     - ★ извршавање у виду скрипте, и
     - ★ превод у извршну датотеку (`.exe`) ← нећемо радити
- На том путу, сусрећемо се са разним невољама, а подијелићемо их у грешке:
  - синтаксичке, и
  - семантичке.
- Ова презентација не представља потпуне информације о рачунарима, програмирању и програмском језику *Python*, него служи као увод или преглед, како бисмо се упознали са основама и знали гдје и шта да тражимо како бисмо стварно научили.

# Потребни алати

- Рачунар - бар током овог предавања, али може и телефон или таблет - само је унос проблематичан
- *Windows* оперативни систем - може, наравно, и неки други али данас рад демонстрирамо овако
- *Python* има двије актуелне верзије (2.7 и 3), с тим да се већ годинама упорно труде да потисну 2.7 - па је треба избјегавати за нове пројекте
- *Python* као апликација за *Windows* бесплатно је доступан на https://www.python.org.
- Познавање енглеског језика
- Добра воља, чврста одлука да се не одустаје и упоран рад

# Примјер 1 - Испис текста

- *File* $\rightarrow$ *New File*

## HelloICBL

```
print("Hello ICBL!")
```

- *File* $\rightarrow$ *Save*
- *Run* $\rightarrow$ *Run Module*

## Output

```
Hello ICBL!
>>>
```

# Типови података

- Основна подјела сигнала:
  - аналогни - звук, бежични пренос података, зрачење, слика
  - дигитални - запис аналогних, али користећи само ограничен број нивоа - ако су само два нивоа, онда су то бинарни сигнали
- Данашњи комерцијално доступни рачунари су дигитални и разумију искључиво бинарне податке
- Математички запис је дат прије два вијека, данас познат као бинарна или Булова алгебра
- Једна бинарна цифра назива се бит (енгл. *binary digit* $\rightarrow$ *bit*)
- Осам бита је бајт (енгл. *byte*), kilo, mega, итд.
- Подаци су осмобитни, 32-битни, итд.
- Корисни записи су још и октални и хексадецимални
- Децимални број 6, осмобитно се представља као 00000110, број 126 пишемо 01111110, а -126 је 10000010
- Означени и неозначени

# Примјер 2 - Типови података

## DataTypes

```
x = 5
print(type(x))
x = "Hello World"
x = 20
x = 20.5
x = ["apple", "banana", "cherry"]
x = ("apple", "banana", "cherry")
x = range(6)
x = {"name" : "John", "age" : 36}
x = {"apple", "banana", "cherry"}
x = True
```

# Примјер 2 - Типови података

## Output

```
<class 'int'>
<class 'str'>
<class 'int'>
<class 'float'>
<class 'list'>
<class 'tuple'>
<class 'range'>
<class 'dict'>
<class 'set'>
<class 'bool'>
>>>
```

# Примјер 3 - Аритметичке и логичке операције

## ALU

```
a = 2
b = 4
c = 'a'
d = 6.2
print("zbir: a+b = " + str(a+b))
print("eksponent: a**b = " + str(a**b))
print("proizvod: a*c = " + str(a*c))
print("kolicnik - cjelobrojno: a//b = " + str(a//b))
print("kolicnik - decimalno: a/d = " + str(a/d))
print("jednakost: a == b = " + str(a==b))
print("binarno I: a & b = " + str(a & b))
print("logicko I: a and b = " + str(a and b))
```

# Преклапање оператора, кастовање

- Када помножимо два броја, добијемо производ
- Шта ће бити ако помножимо број и слово?
- Или број и ријеч?
- Претварање једног типа података у други назива се кастовање
- Објекти:
  - промјенљивих вриједности - *mutable*
  - непромјенљивих вриједнсоти - *immutable*

# Контрола тôка

- Рачунари не одлучују. Бар не још увијек
- Рачунари поступају по инструкцијама
- Инструкција може да се изврши или не изврши у зависности од услова, односно може да се изврши једна или друга, опет, наравно, зависно од стања нечег другог:
  - ако је температура већа од $22^o$ C, искључи грijање
  - ако је температура мања од $18^o$ C, укључи грijање
  - не извршавај ништа, док се не притисне овај тастер
  - изврши сабирање свих бројева у овом низу, тј. 10 000 сабирања
- Графички приказ тôка извршавања назива се дијаграм тôка
- Тôк контролишемо користећи се гранањем (`if-else`, `switch`) и петљама (`for`, `while`)

# Примјер 4 - Унос података

## DataInput

```
x = input("prvi broj: ")
y = input("drugi broj: ")
x = int(x)
y = int(y)
z = x + y
print("Zbir je: " + str(z))
```

# Примјер 5 - Гранање

## Branch

```
x = input("prvi broj: ")
r = input("radnja: ")
y = input("drugi broj: ")
x = int(x)
y = int(y)
if r == "+":
    print("zbir: " + str(x+y))
elif r == "-": print("razlika: " + str(x-y))
elif r == "*": print("proizvod: " + str(x*y))
elif r == ":": print("kolicnik: " + str(x/y))
else: print("Neispravna radnja!")
```

# Примјер 6 - Петља

## Loop

```
while 1:
    x = input("prvi broj: ")
    r = input("radnja: ")
    y = input("drugi broj: ")
    print("\n\ntreba da izracunamo: " + x + r + y )
    if r == "+": print("zbir: " + str(x+y))
    elif r == "-": print("razlika: " + str(x-y))
    elif r == "*": print("proizvod: " + str(x*y))
    elif r == ":": print("kolicnik: " + str(x/y))
    else: print("Neispravna radnja!")
    izlaz = input("Kraj?\t")
    if izlaz == "d" or izlaz == "D":
        break
```

# Енкапсулација и апстракција

- Тако далеко сам видио, зато што сам стојао сам на плећима дивова
- Све што постоји, а да је створено људском руком, настало је комбинацијом већ постојећег
- Зашто да измишљамо топлу воду?
- Дајте ми ослонац, помјерићу свијет
- Право питање је: шта је проблем?
- Видјели смо који су типови података, видјели смо које су доступне аритметичке операције.
- Како ћемо израчунати коријен? Синус? Интеграл?
- Управљати роботом?
- Програмирање игара?

# Библиотеке

- `numpy` - нумерички прорачуни
- `matplotlib` - графичка представа података
- `math` - напредније математичке операције
- `sys` - за рад са системским ресурсима
- `pytorch` - машинско учење
- `scipy` - још напредније математичке операције
- `pyserial` - комуникација преко серијског порта
- `random` - случајне вриједности
- `time` - в реално вријеме
- `pyfirmata` - комуникација серијским портом са Ардуином
- и бесконачно много других...
- кључна ријеч `import`
- *GNU/Linux* оперативни системи се много лакше сналазе са свим овим ресурсима

# Примјер 7 - Кориштење библиотека и функција

## Function

```
import random

broj = input("Koliko kockica?\n")
broj = int(broj)

ponovo = "d"
while ponovo == "d" or ponovo == "D":
    print("Bacam kockicu...")
    for i in range(broj):
        print("Kockica " + str(i+1) + ": " + str(rando
    ponovo = input("Ponovo?\n")
```

# Примјер 8 - Писање и функција

## Function

```
def identitet(ime, prezime):
    print(ime + " " + prezime)
    x = 5

identitet("Nikola", "Tesla")
x = 6
print(x)
```

- Домен вриједности промјенљиве:
  - локални, и
  - глобални

# Уграђени (енгл. *embedded*) системи

- Уграђени системи су рачунарски системи намијењени за извршавање специфичне функције у реалном времену, и могу да буду сасставни дио обимнијих електромеханичких (над)система.
- Рачунари су, на примјер, системи опште намјене, дакле нису „уграђени” у овом смислу.
- Махом су дигитални, а централни дио је увијек микроконтролер.
- Микроконтролер је процесор са периферијама и (малом) меморијом.
- *Internet of Things* - интернет свега, паметни системи
- Аутомобилска индустрија (енгл. *automotive*)
- Интересантан свијет, на граници између хардвера и софтвера, рачунарска електроника, рачунарски инжењеринг, мехатроника, роботика.
- Ардуино је сјајан први корак на том путу.

# Примјер 9 - Трепћући Ардуино

## Blink

```
import pyfirmata
import time

board = pyfirmata.Arduino('COM3')

while True:
    board.digital[13].write(1)
    time.sleep(1)
    board.digital[13].write(0)
    time.sleep(1)
```

# Примјер 10 - Игре

- Једноставније игре, текстуалне, као што је бацање коцкице малопријe већ можете да развијате, имате све што је потербно - питање је само идеје.
- An Introduction to Interactive Programming in Python
- CodeSkulptor
- *Demos:*
  - *Galaxy Invaders*
  - *Pyman*
  - *Rice Racer*

# Референце

- Google!!!
- Python
- Школа кôда,
- LearnPython
- Coursera и edX
- Утичнионица- основно о Ардуину, са Електротехничког факултета у Бањој Луци
- 1000 за будућност- увод у рад са:
  - Пи рачунар
  - програмски језик *Python*
- Званична презентација *Raspberry Pi*
- Увод у електронику - серија Три минута електронике
- Књига о LTspice, софтверском симулатору електричних кола