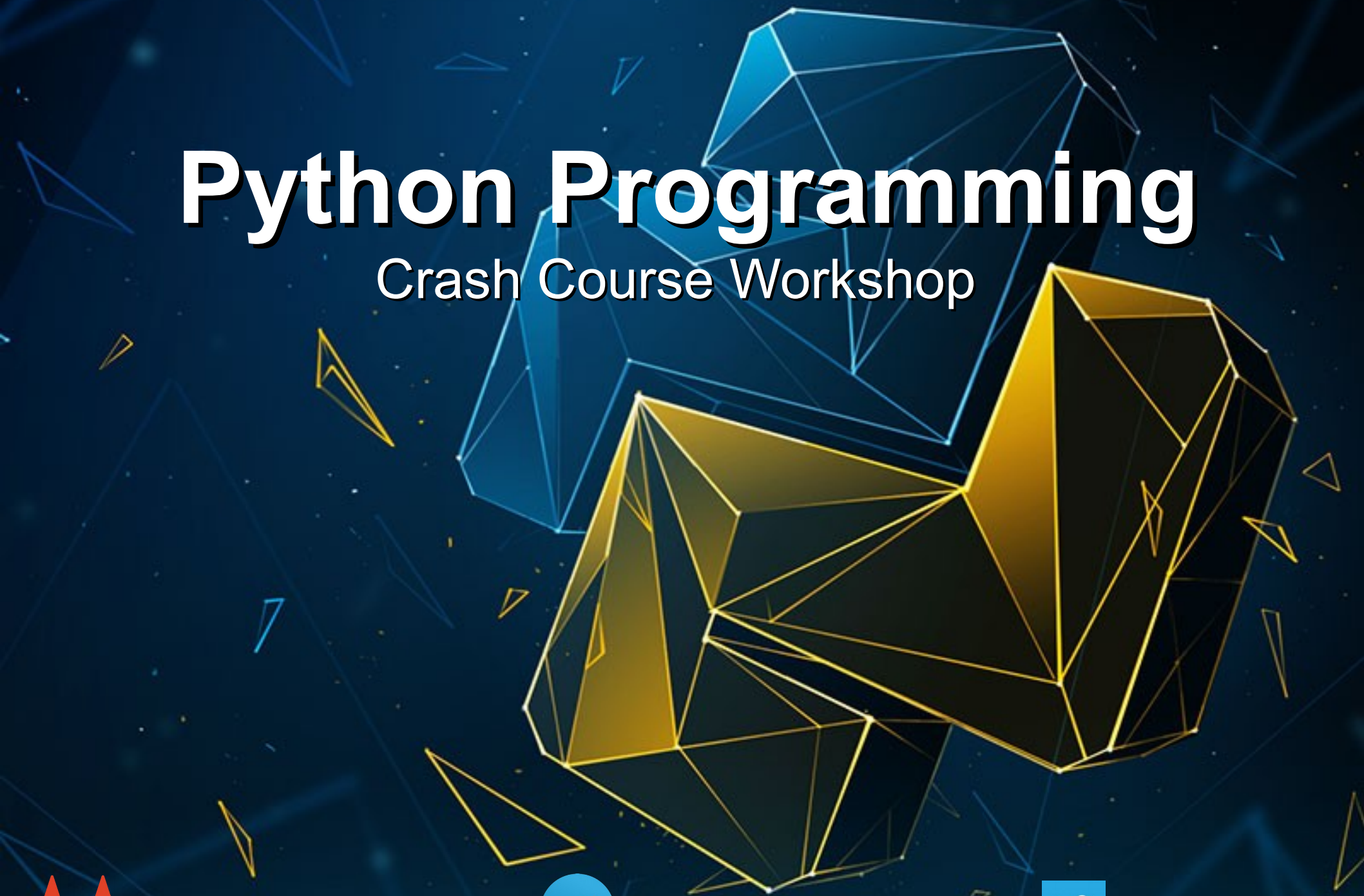
26-Nov-2018

Python Programming

Crash Course Workshop



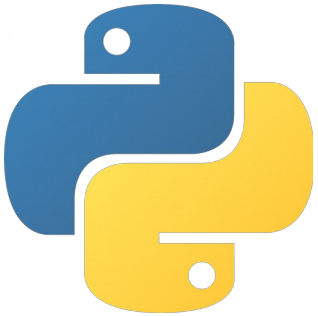
behzadim

mariebehzadi

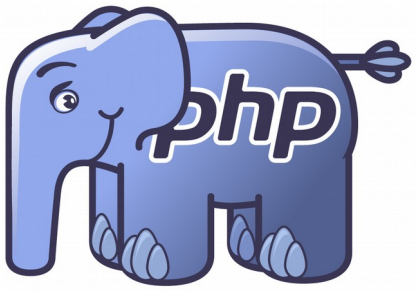
mariebehzadi

Scripting Language

- What is Scripting Language?
- Interpreted Programs vs Compiled Programs
- #!



#!/bin/bash



What is Python?

- Created in 1990 by Guido van Rossum
- Rossum was BDFL until 2018
- Origin name from British comedy group Monty Python
- An important goal of Python's developers is keeping it fun to use
- Interpreted high-level programming language
- General-purpose language
- Paradigm
  - Object-oriented
  - Imperative
  - Functional
  - Procedural
  - Reflective


```
import this
"""The Zen of Python, by Tim Peters. (poster by Joachim Jablon)"""

 1 Beautiful is better than ugly.
 2 Explicit is better than impl..
 3 Simple is better than cσмplεx.
 4 Cσмplεx is better than cOmp1|c@ted.
 5 Flat is better than nested.
 6 S p a r s e  is better than dense.
 7 Readability counts.
 8 Special cases aren't special enough to break the rules.
 9 Although practicality beats purity.
10 raise PythonicError("Errors should never pass silently.")
11 # Unless explicitly silenced.
12 In the face of ambiguity, refuse the temptation to guess.
13 There should be one-- and preferably only one --obvious way to do it.
14 # Although that way may not be obvious at first unless you're Dutch.
15 Now is better than ...                    never.
16 Although never is often better thanrightnow.
17 If the implementation is hard to explain, it's a bad idea.
18 If the implementation is easy to explain, it may be a good idea.
19 Namespaces are one honking great idea -- let's do more of those!
```

Zen of Python

Available Versions

Version history

Version	Release date	Supported until
2.2	2001-12-21[24]	2003-05-30[25]
2.3	2003-07-29[26]	2008-03-11[27]
2.4	2004-11-30[28]	2008-12-19[29]
2.5	2006-09-19[30]	2011-05-26[31]
2.6	2008-10-01[32]	2013-10-29[33]
2.7	2010-07-03[34]	2020[35]
3.0	2008-12-03[36]	2009-06-27[37]
3.1	2009-06-27[38]	2014-06[39]
3.2	2011-02-20[40]	2016-02-27[41]
3.3	2012-09-29[42]	2017-09-29[43]
3.4	2014-03-16[44]	2019-03-16[45]
3.5	2015-09-13[46]	2020-09-13[47]
3.6	2016-12-23[48]	2021-12[49]
3.7	2018-06-15[50]	2023-06[50]

Old version | Older version, still supported | **Latest version** | Future release

PYTHON 2 vs 3

2018 DIFFERENCES

PYTHON 2		PYTHON 3
Legacy	>	Future
It is still entrenched in the software at certain companies		It will take over Python 2 by 2020
Library	≠	Library
Many older libraries built for Python 2 are not forwards-compatible		Many of today's developers are creating libraries strictly for use with Python 3
0100 0001 ASCII	+	Unicode 0000 0000 0100 0001
Strings are stored as ASCII by default		Text strings are Unicode by default
5/2=2	≠	5/2=2.5
It rounds your calculation down to the nearest whole number		The expression 5 / 2 will return the expected result
print “hello”	≠	print (“hello”)
Python 2 print statement		The print statement has been replaced with a print () function

LEARNTOCODEWITH.ME

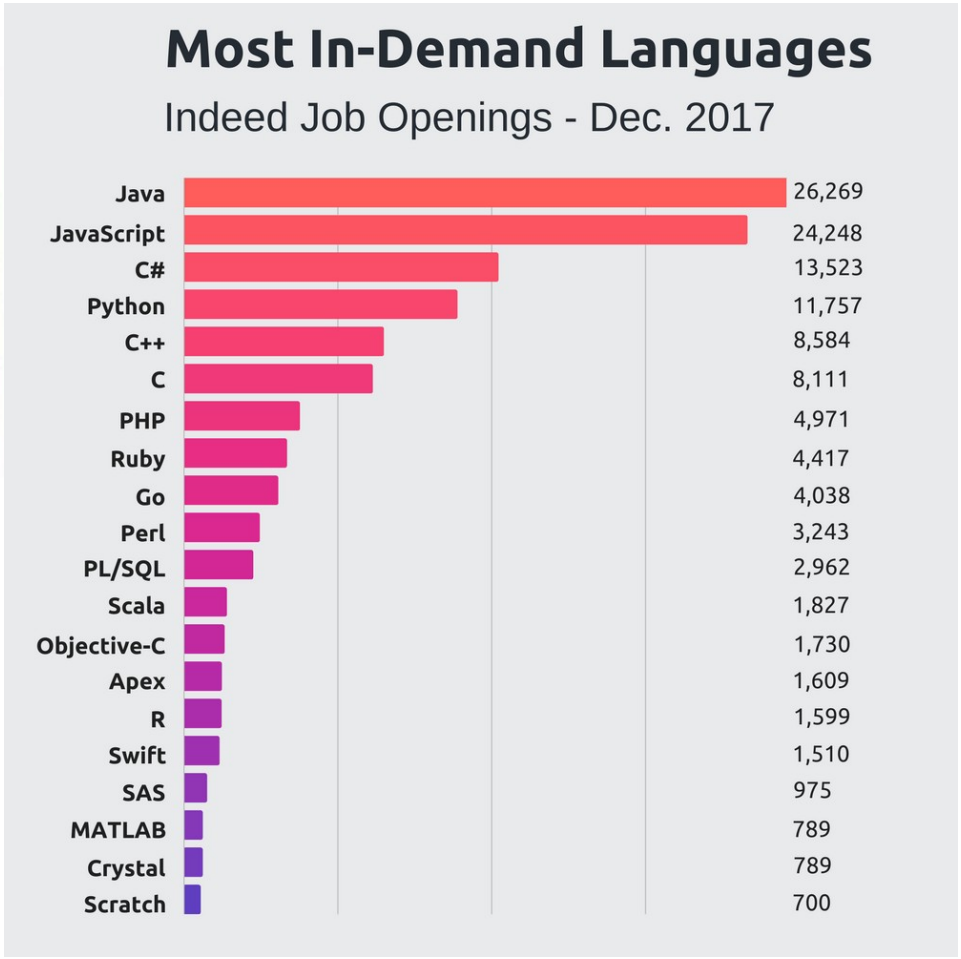
Python 2.7

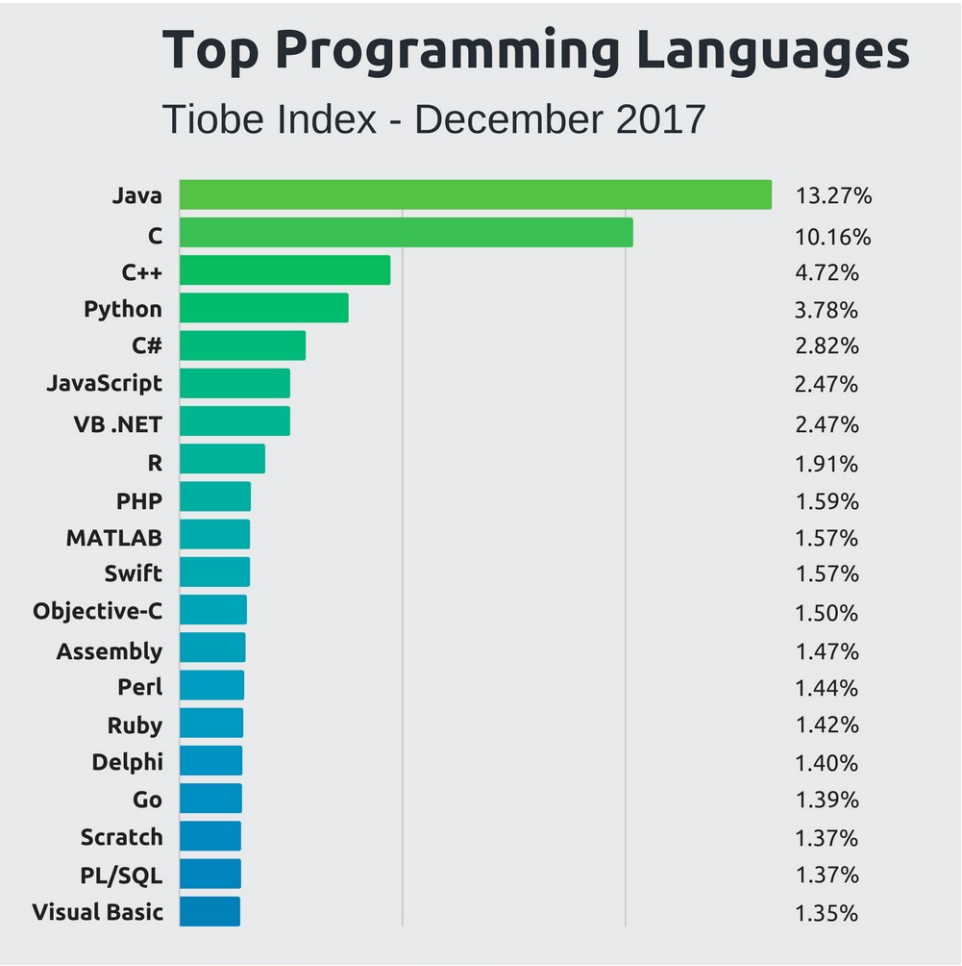
2016 → 71.9%
2017 → 63.7%

Why Python?

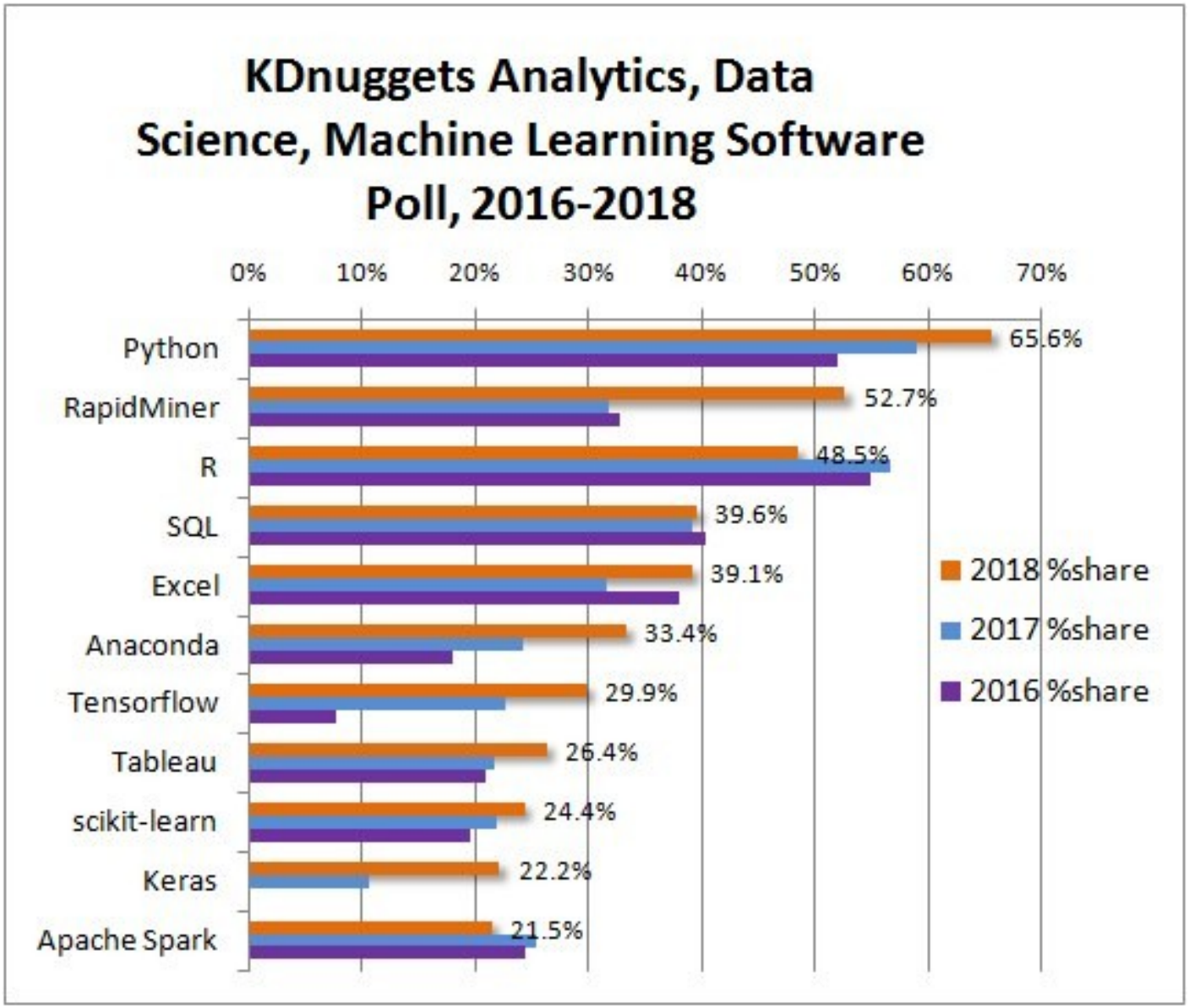
- Designed to be easy to learn and master
- Readable and Maintainable Code
- Multiple Programming Paradigms
- Robust Standard Library
- Extensive Support Libraries
- Compatible with Major Platforms and Systems
- Many Open Source Frameworks and Tools
- General Purpose Programming Language
- Adopt Test Driven Development
- Lots of 3rd Party Tools
- Active Open-Source Community

Popularity


Most In-Demand Languages
Indeed Job Openings - Dec. 2017
Java 26,269
JavaScript 24,248
C# 13,523
Python 11,757
C++ 8,584
C 8,111
PHP 4,971
Ruby 4,417
Go 4,038
Perl 3,243
PL/SQL 2,962
Scala 1,827
Objective-C 1,730
Apex 1,609
R 1,599
Swift 1,510
SAS 975
MATLAB 789
Crystal 789
Scratch 700


Top Programming Languages
Tiobe Index - December 2017
Java 13.27%
C 10.16%
C++ 4.72%
Python 3.78%
C# 2.82%
JavaScript 2.47%
VB .NET 2.47%
R 1.91%
PHP 1.59%
MATLAB 1.57%
Swift 1.57%
Objective-C 1.50%
Assembly 1.47%
Perl 1.44%
Ruby 1.42%
Delphi 1.40%
Go 1.39%
Scratch 1.37%
PL/SQL 1.37%
Visual Basic 1.35%

Popularity


KDnuggets Analytics, Data Science, Machine Learning Software Poll, 2016-2018
0%
10%
20%
30%
40%
50%
60%
70%
Python
RapidMiner
R
SQL
Excel
Anaconda
Tensorflow
Tableau
scikit-learn
Keras
Apache Spark
65.6%
52.7%
48.5%
39.6%
39.1%
33.4%
29.9%
26.4%
24.4%
22.2%
21.5%
2018 %share
2017 %share
2016 %share

Productivity

- Reduced Development Time
  - Code is 2-10x shorter than C, C++, Java
- Improved Program Maintenance
  - High Readability
- Less Training
  - Easy to Learn


Python Applications
Web and Internet Development
Desktop GUI Applications
Scientific and Numeric
Software Development
Education
Business Applications
Games and 3D Graphics
Network Programming
Database Access

Python Basics (Comments)

Python Comments

```
1
2  price = 10.2
3  # increase price to 5%
4  price = price * 1.05
5
6  salary = 5000
7  salary = salary * 1.02  # increase salary 2% for the employee
8
9  def increase_salary(sal,rating,percentage):
10     """ increase salary base on rating and percentage
11     rating 1 - 2 no increase
12     rating 3 - 4 increase 5%
13     rating 4 - 6 increase 10%
14
15     """
```

Python Basics (Math Operations)

Numbers and math

Operator	Description
+ plus	Sum
- minus	Subtraction
/ slash	Floor division
* asterisk	Multiplication
** double asterisk	Exponentiation
% percent	Remainder
< less-than	Comparison
> greater-than	Comparison
<= less-than-equal	Comparison
>= greater-than-equal	Comparison

Python Basics (Variable and Types)

```
1   # Declare a variable and initialize it
2   f = 0
3   print(f)
4   # re-declaring the variable works
5   f = 'guru99'
6   print(f)
7
8
9
10
11
12
```

you can re-declare the variables, even-after if it is declared once. it works fine

```
Run  Python5.1
"C:\Users\DK      e                    thon Test\Py
 5/PythonC    de5/
0
guru99
```

Python Basics (Blocks)

```
1   def main():
2       print("Hello World!")
3   print("Guru99")
```

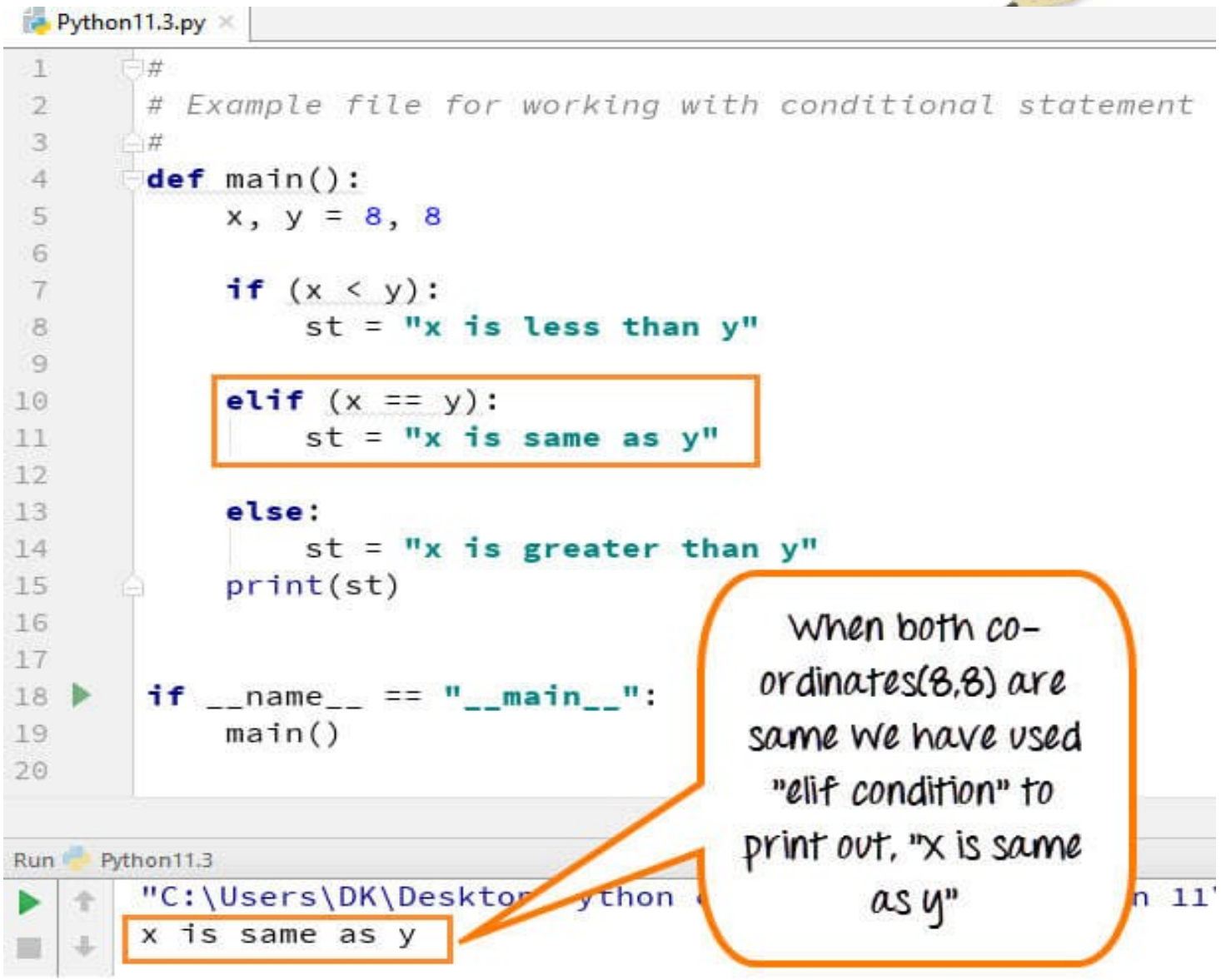
main()

Run Code4_1

```
"C:\User  \DK                    code\Python T
 4/Code4/Code4_1.py"
Guru99
```

Why only "guru99" get printed out?

Python Basics (Conditions)



Python Basics (Loops)

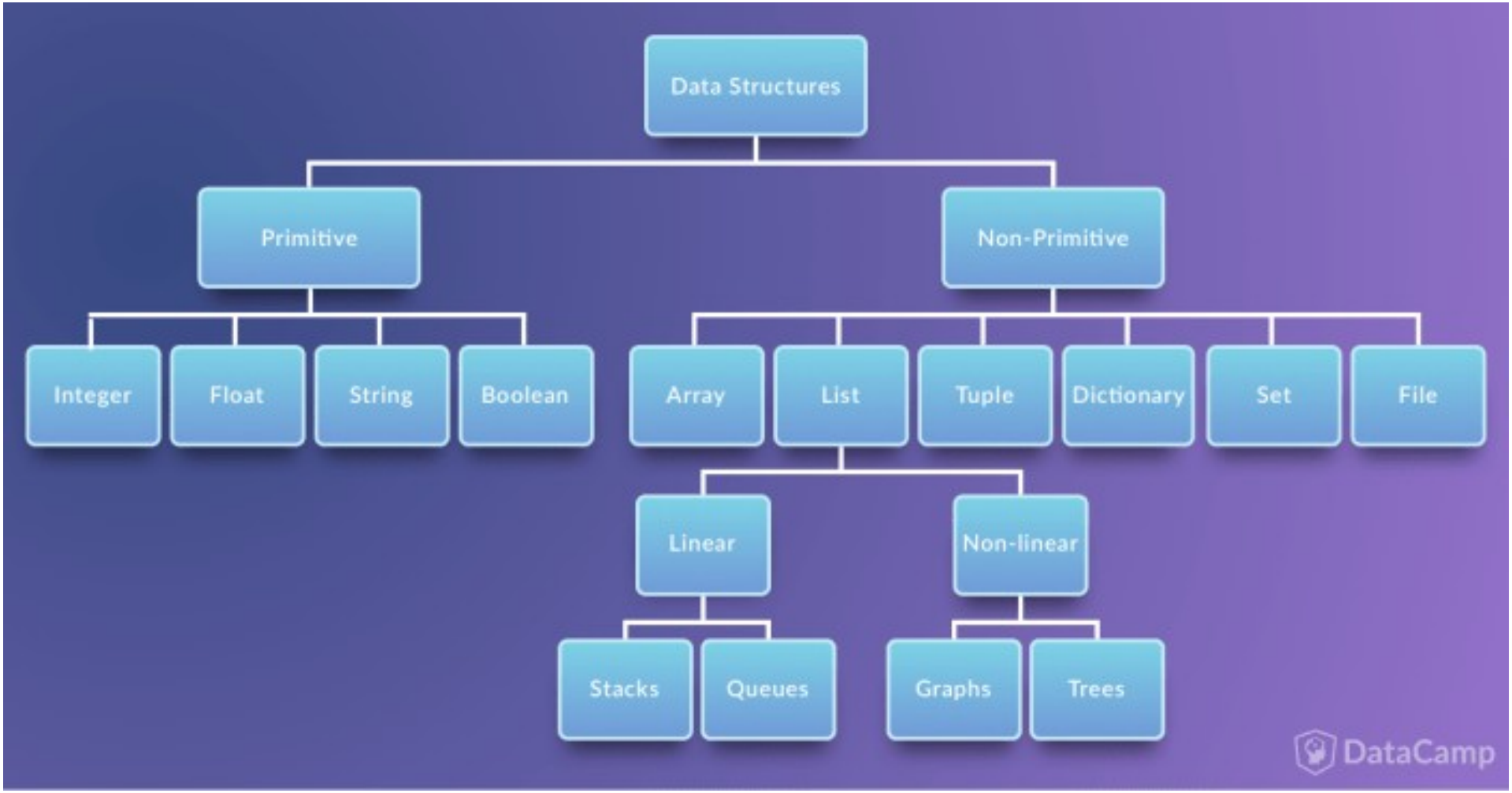
```
In [30]: my_list=range(0,10,2)
         for i in my_list:
             print(i)

         0
         2
         4
         6
         8
```

PythonExample.py

```
1   # Program published on https://beginnersbook.com
2
3   # Python program to calculate the sum of n Natural Numbers
4
5   # n denotes upto which number you want to calculate the sum
6   # for example, if n is 5 then the sum of first 5 natural numbers
7   num = int(input("Enter the value of n: "))
8   hold = num
9   sum = 0
10
11  if num <= 0:
12      print("Enter a whole positive number!")
13  else:
14      while num > 0:
15          sum = sum + num
16          num = num - 1;
17      # displaying output
18      print("Sum of first", hold, "natural numbers is: ", sum)
```

Python Basics (Data Structures)



Python Basics (Data Structures)

Built-in Data Structures

- List
- Tuple
- Dictionary
- Set

- `[1, 2, 3]`
- `(1, 2, 3)`
-
```
{
    'a': 1,
    'b': 2,
    'c': 3,
}
```
- `{1, 2, 3}`

Python Basics (Data Structures)

List Vs Set Vs Dictionary Vs Tuple

Lists	Sets	Dictionaries	Tuples
List = [10, 12, 15]	Set = {1, 23, 34} Print(set) -> {1, 23,24} Set = {1, 1} print(set) -> {1}	Dict = {"Ram": 26, "mary": 24}	Words = ("spam", "egss") Or Words = "spam", "eggs"
Access: print(list[0])	Print(set). Set elements can't be indexed.	print(dict["ram"])	Print(words[0])
Can contains duplicate elements	Can't contain duplicate elements. Faster compared to Lists	Can't contain duplicate keys, but can contain duplicate values	Can contains duplicate elements. Faster compared to Lists
List[0] = 100	set.add(7)	Dict["Ram"] = 27	Words[0] = "care" ->TypeError
Mutable	Mutable	Mutable	Immutable - Values can't be changed once assigned
List = []	Set = set()	Dict = {}	Words = ()
Slicing can be done print(list[1:2]) -> [12]	Slicing: Not done.	Slicing: Not done	Slicing can also be done on tuples
Usage: Use lists if you have a collection of data that doesn't need random access. Use lists when you need a simple, iterable collection that is modified frequently.	Usage: - Membership testing and the elimination of duplicate entries. - when you need uniqueness for the elements.	Usage: - When you need a logical association b/w key:value pair. - when you need fast lookup for your data, based on a custom key. - when your data is being constantly modified.	Usage: Use tuples when your data cannot change. A tuple is used in comibnation with a dictionary, for example, a tuple might represent a key, because its immutable.

More Details (Functions)

Python10.1.py

```
1  #define a function
2  def func1():
3      print ("I am learning Python Function")
4
5  func1()
6  #print func1()
7  #print func1
8
9
```

Function definition

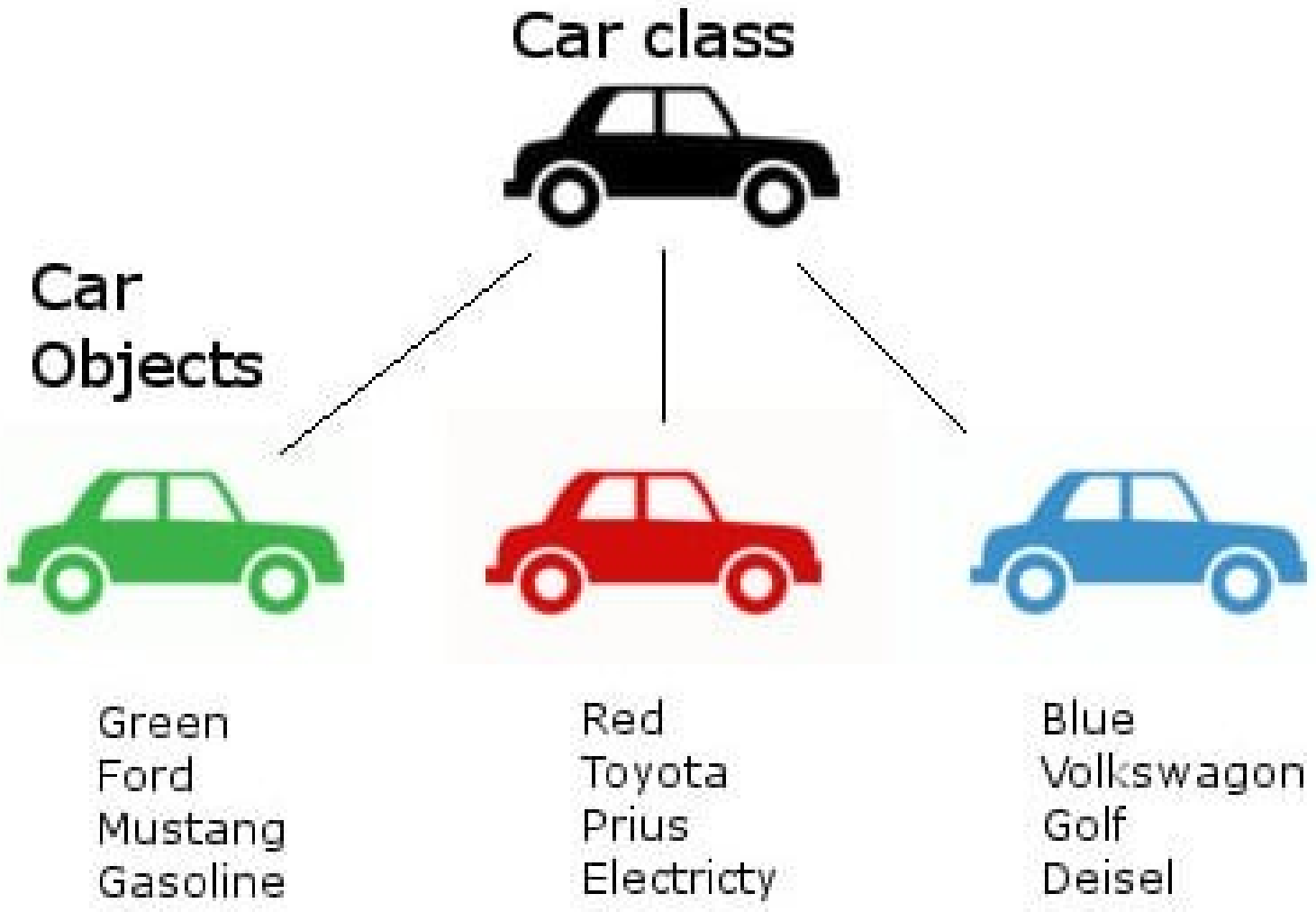
Function Call

Run Python10.1

```
"C:\Users\DK\Desktop\Python code\Python Test\Python 10\Python10
 10/Python10 Code/Python10.1.py"
I am learning Python Function
```

Function output

More Details (Class)



More Details (Exception Handling)

try:

{ Run this code

except:

{ Execute this code when there is an exception

else:

{ No exceptions? Run this code.

finally:

{ Always run this code.

More Details (Module)

- logically organize your Python code
- Grouping related code → easy, high readability
- Simply, a module is a file consisting of Python code
- A module can define functions, classes and variables.
- A module can also include runnable code.

```
import module1[, module2[,... moduleN]
```

```
from modname import name1[, name2[, ... nameN]]
```

```
from modname import *
```

The Python Standard Library

- Python comes standard with a set of modules, known as the “standard library”
- Incredibly rich and diverse functionality available from the standard library
- All common internet protocols, sockets, CGI, OS services, GUI services (via Tcl/Tk), database, Berkeley style databases, calendar, Python parser, file globbing/searching, debugger, profiler, threading and synchronisation, persistency, etc

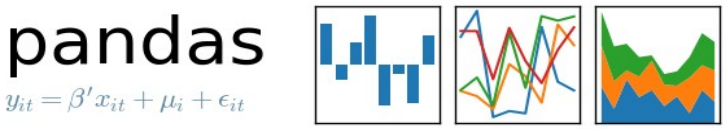
https://docs.python.org/3/library/

3rd Party Libraries

https://matplotlib.org/

http://www.numpy.org/

https://www.scipy.org/



https://pandas.pydata.org/

https://scikit-learn.org/

https://pypi.org/project/opencv-python/

Numpy

- Numerical Python
- Why NumPy?
- Designed for scientific computing
- Open-Source
- ndarray is the core object
- Advanced functions to work with arrays
- Universal Functions (ufunc)
- Basic Linear Algebra
- Basic Statistical Operations
- Random Simulation
- discrete Fourier transforms
- mathematical, logical, shape manipulation
- Numeric + Numarray ~ NumPy

Numpy / Linear Algebra

Example Solving a Matrix Equation

Use a matrix equation to solve $\begin{cases} -2x + 4y = 2 \\ -3x + 7y = 7. \end{cases}$

The matrix form of the equation is

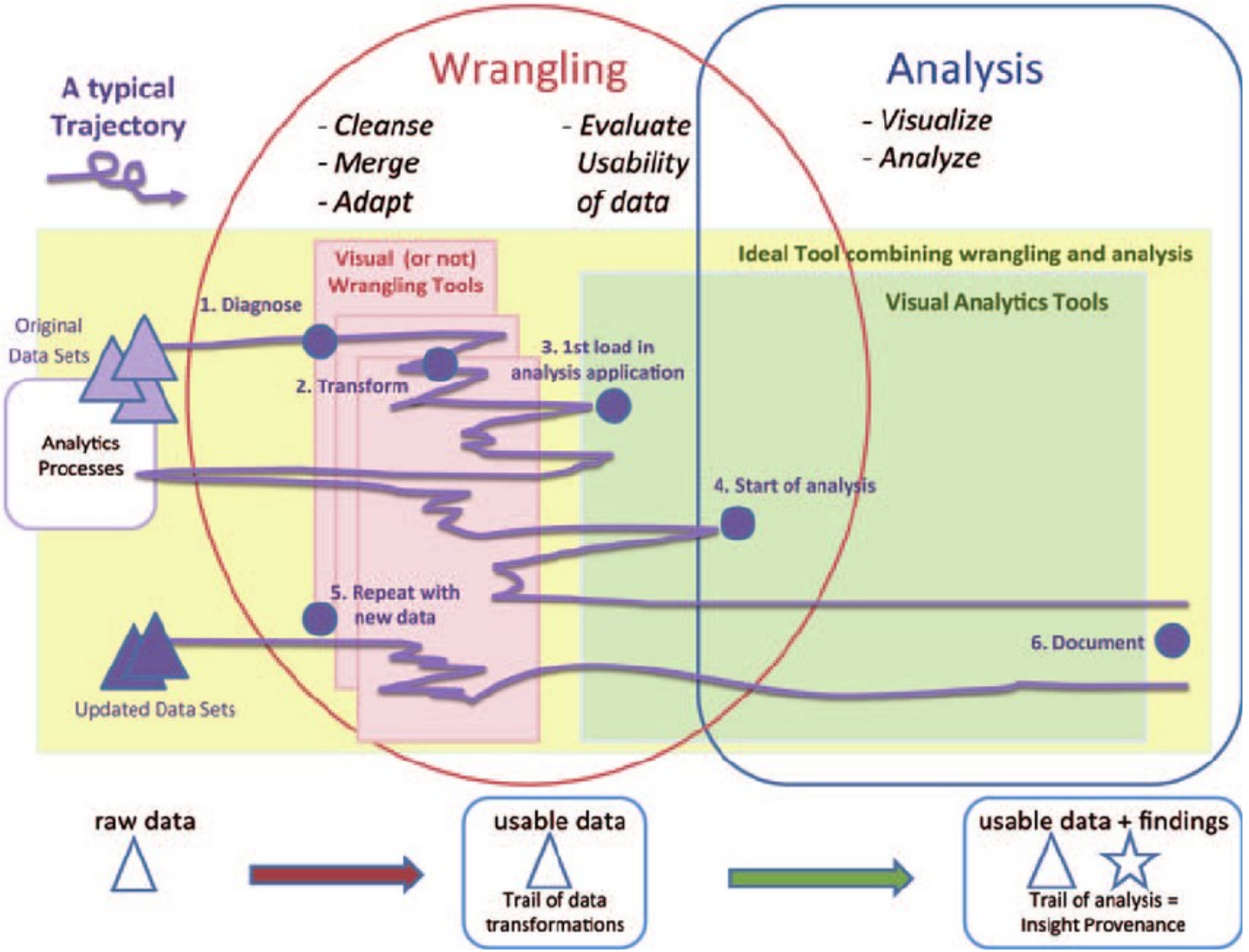
$$\begin{bmatrix} -2 & 4 \\ -3 & 7 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \end{bmatrix}.$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -2 & 4 \\ -3 & 7 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 7 \end{bmatrix} = \begin{bmatrix} -7/2 & 2 \\ -3/2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 7 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$$

Pandas

- data manipulation and analysis
- data structures and operations for manipulating numerical tables and time series
- The name is derived from the term "panel data"
- DataFrame is the Core Object
- Suitable for Data Wrangling
- Open-Source

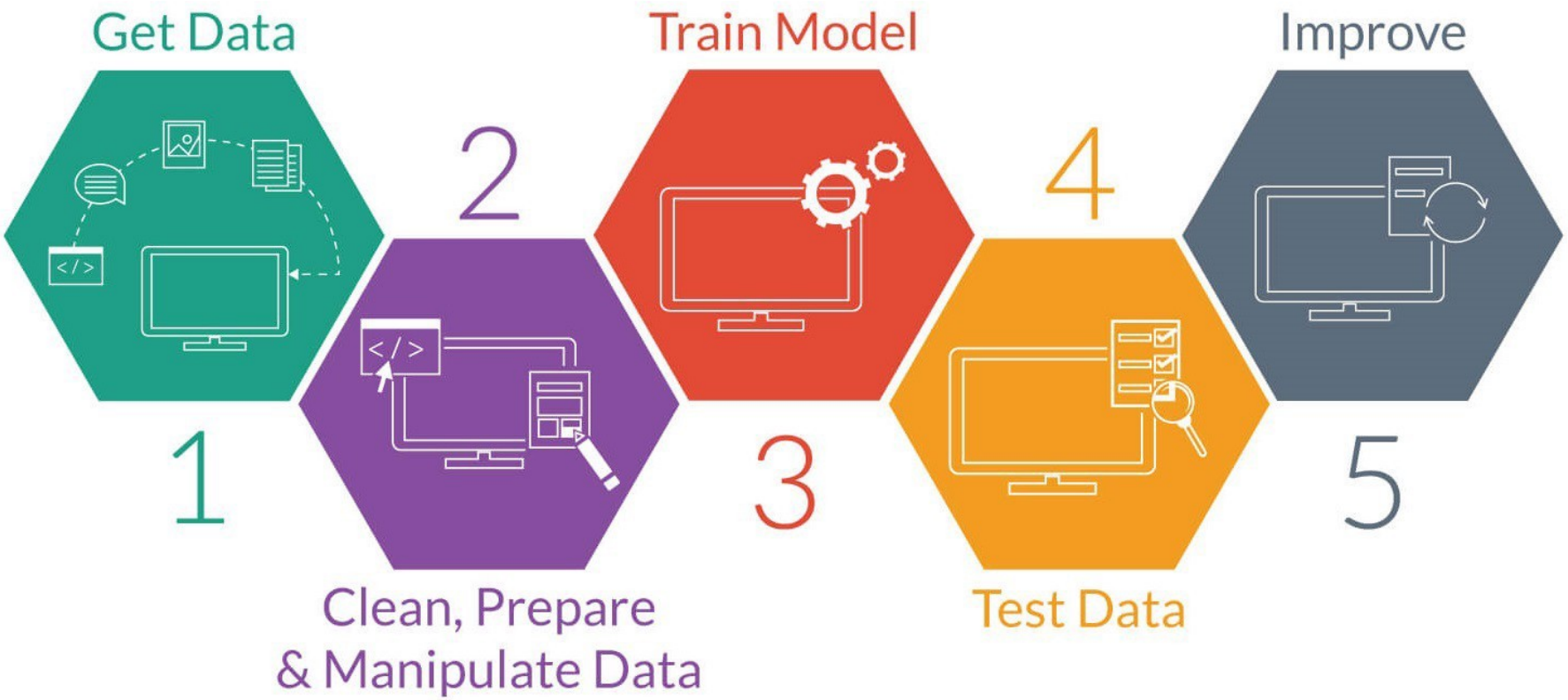
Pandas (Data Wrangling)



scikit-learn

- free software machine learning library for the Python
- Used in data mining and data analysis
- Built on NumPy, SciPy, and matplotlib
- Open-Source
- classification, regression and clustering algorithms
  - support vector machines
  - random forests
  - gradient boosting
  - k-means
  - DBSCAN

scikit-learn (Data Modeling & Prediction)




https://shirazlug.ir/
contact@shirazlug.ir
shirazlug
shirazlug.ir
shirazlug_ir