MASTER MEIM 2021-2022

Python Programming Course Lesson 1

AN INTRODUCTION TO PYTHON

Lesson given by prof. Mariacarla Staffa

Prof. Computer Science at the University of Naples Parthenope

Introduction to Python

AIM

- Setup a Python development and write our first python program.

LEARNING OUTCOMES

At the end of the lesson, you are expected to:

- Install and run the Python interpreter.
- Create and execute Python programs using PyCharm

Abstractions and Models

Science is based on abstractions: explaining phenomena through models that describe them

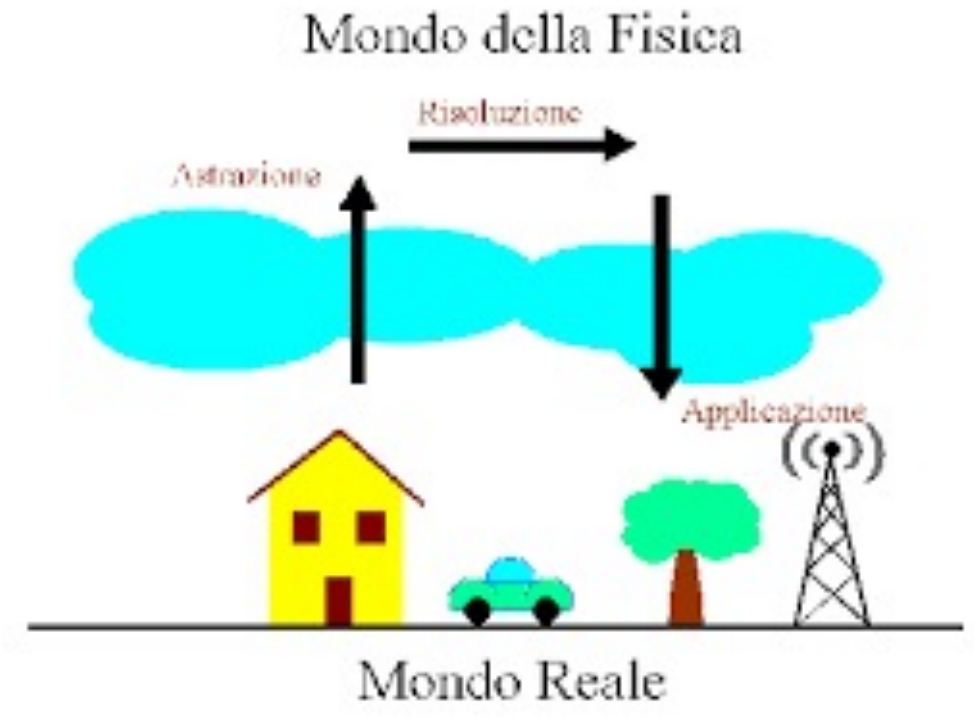
a physicist writes the equation that represents the motion of a body
a chemist describes the reaction that explains the formation of a substance
a biologist describes the process of developing an organism

The Abstactions of Informatics

- the abstractions of computer science can be mechanically
- It means that they can be animated, that is, you can "give them life and see what happens" without having to build a new physical representation of the abstraction itself each time.

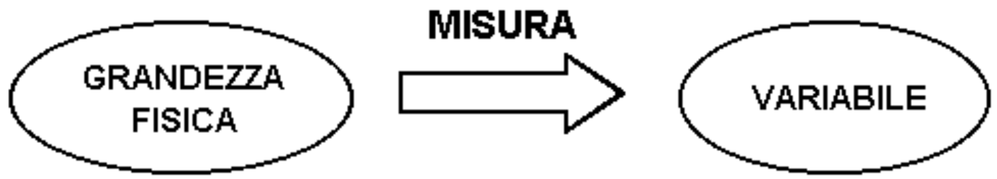
Computer Science Applications

- Abstracting a problem
- Resolution through a model (physicist/mathematical/empirical methods)
- Application (... also through computer science)



Real World – Physics World

- During abstraction, you want to create a representation of the physical magnitude (by measuring the operation) of which you have only a perception in the real world.
- The world of physics is not made up of objects like the real world, but of physical greatness. A physical magnitude is an attribute of the object that we're studying



Physics equations

Once the necessary measurements are made, formal equations of physics are exploited.

For example, Newton's law:

$$F = M \cdot a$$

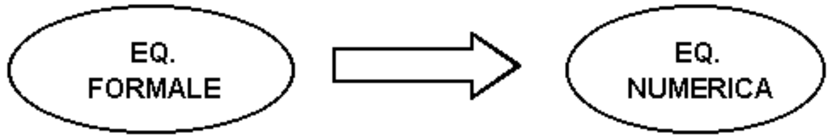
is a formal equation.

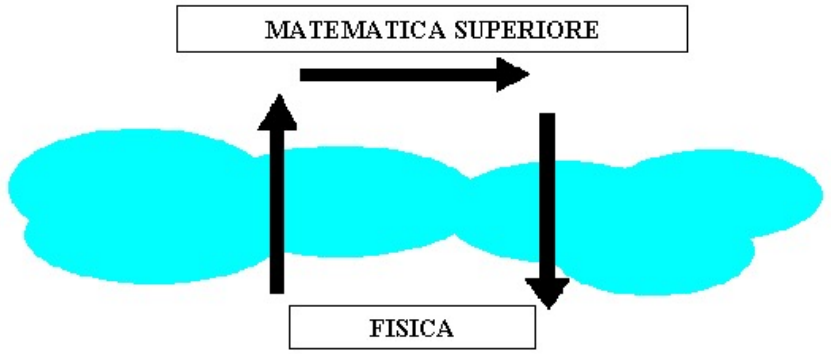
Each symbol corresponds to a physical magnitude that must be measurable.

If this results in a non-measurable one, it could be said that the equation is false.

Physics equations – Numeric Equation

- To the formal equation we move to the numerical equation in which each symbol is associated with a number and its corresponding measure unit.
- However, the world of physics is not able to solve all the problems of the real world, or rather it needs more precise mathematical tools such as infinitesimal calculation and differential equations.

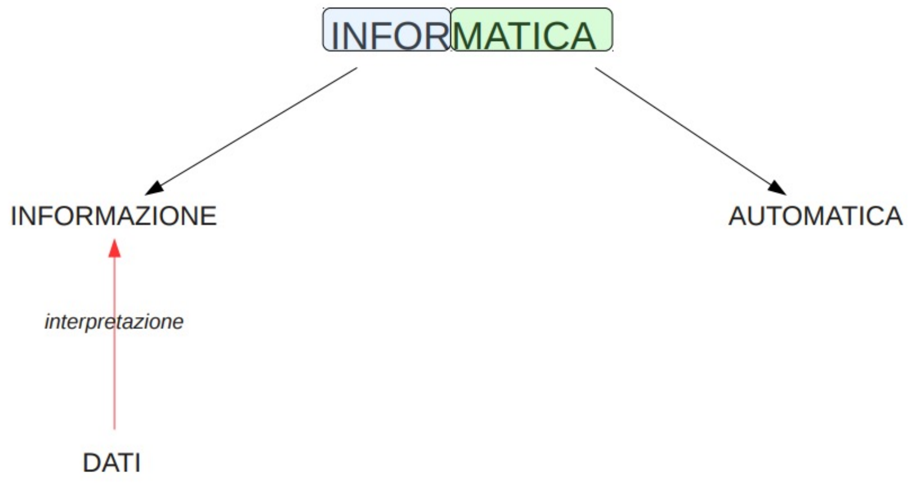




From Equations
to
Computer Science Applications

What is computer science?

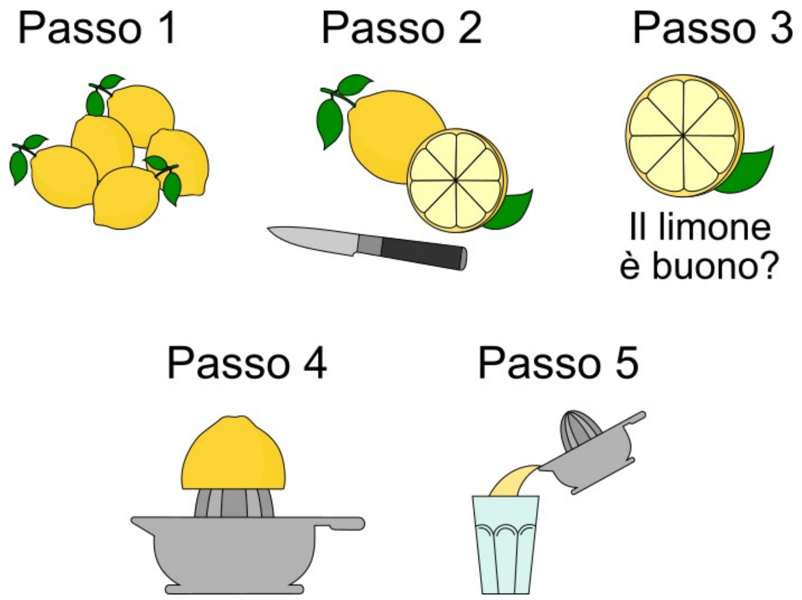
Computer science is the science of the representation and processing of information



Computer science is the study of algorithms

What is an algorithm?

A **well-ordered** and **finished** set of **unambiguous** and **actually calculable** operations that, applied to a set of initial conditions, produces a result and ends in a <u>finite amount of time</u>.



Introduction to Programming

- An algorithm is a finite sequence of effective steps that solve a problem.
  - A step is effective if it is unambiguous and possible to perform.
  - The number of steps must be finite (rather than infinite) so that all of the steps can be completed.
  - An algorithm must be translated into a computer program before a computer can be used to solve a problem.
- A computer program is a sequence of instructions that control the behaviour of a computer.

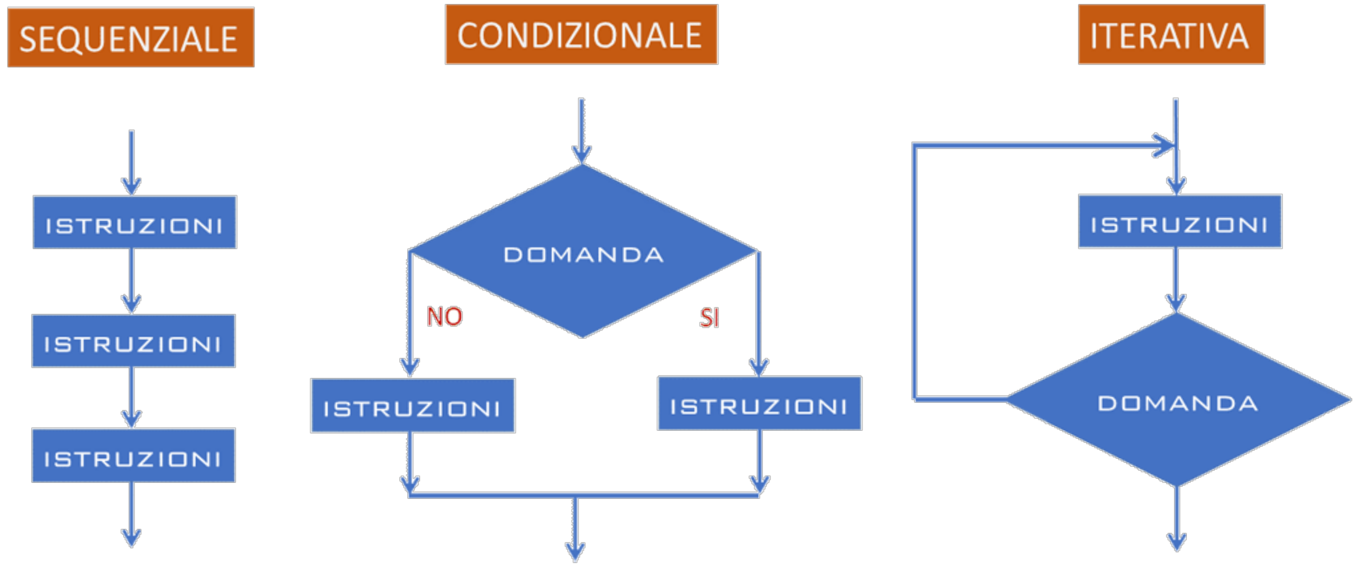
Formal properties of algorithms

- The first property is obviously correctness: An algorithm must provide a correct solution to the problem and terminate
- But efficiency is also of paramount importance, which is measured in relation to the space resource and with respect to the time resource
- The efficiency of an algorithm indicates how thrifty it uses the resources available
- To quantify the efficiency of an algorithm as the number of data on which it acts changes, it is used to try to express its computational complexity

Design an algorithm

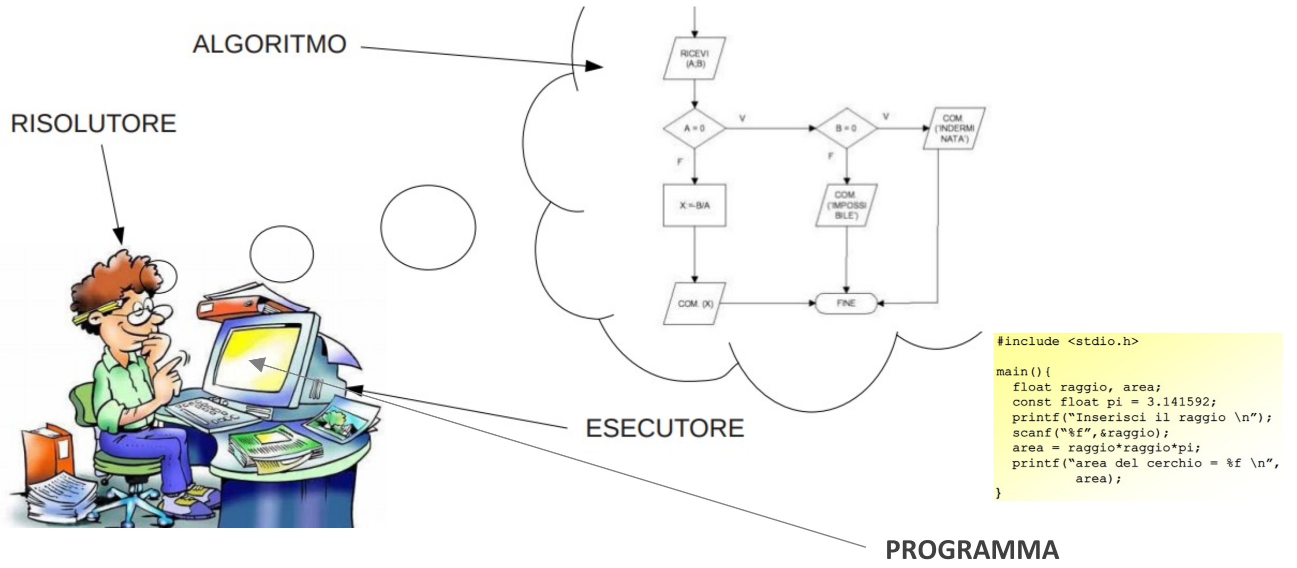
- Finding the right algorithm to solve a given problem is the most creative part of a computer scientist's work.
- Each algorithm can be broken down into three basic types of operations

  - Sequential operations
  - Conditional operations
  - Iterative operations

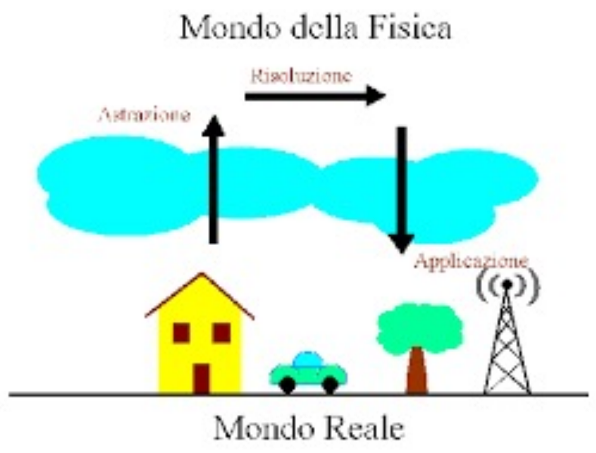


What is a program?

- The algorithm is an abstract representation of the solution of a problem, the program is the concrete expression of the algorithm
- A program is the encoding of an algorithm in a language that the computer can understand and execute



```
#include <stdio.h>

main(){
  float raggio, area;
  const float pi = 3.141592;
  printf("Inserisci il raggio \n");
  scanf("%f",&raggio);
  area = raggio*raggio*pi;
  printf("area del cerchio = %f \n",
          area);
}
```



2. Programming Languages

- Computers cannot write programs on their own as they don't understand human needs unless we communicate with the computer through programming languages.

A programming language is a *computer language engineered to communicate instructions to a machine.*

- Programs are created through programming languages to control the behavior and output of a machine through accurate algorithms, similar to the human communication process.

2.1 Machine Language

- Machine language, also called machine code, is a low-level computer language that is designed to be directly understandable by a computer and it is the language into which all programs must be converted before they can be run.
- It is entirely comprised of binary, 0's and 1's.
- In machine language, all instructions, memory locations, numbers and characters are represented in 0's and 1's.

```
00000100 10000000
```

2.1 Machine Language

- Advantages

it can run and *execute very fast* as the code will be directly executed by a computer and the programs *efficiently utilize memory*

- Disadvantages
  - Machine language is almost impossible for humans to use because it consists entirely of numbers.
  - Machine language programs are hard to maintain and debug.
  - Machine language has no mathematical functions available.
  - Memory locations are manipulated directly, requiring the programmer to keep track of every memory location.

2.2 Assembly Language

The Assembly Language is a human-readable notation for the machine language. Assembly language replaces the instructions represented by patterns of 0's and 1's with alphanumeric symbols also called as mnemonics in order to make it easier to remember and work with them including reducing the chances of making errors.

```
ADD 3, 5, result
SUB 1, 2, result
```

2.2 Assembly Language

- Advantages

Because of alphanumeric symbols, assembly language is also known as Symbolic Programming Language. The use of mnemonics is an advantage over machine language.

- Disadvantages
  - There are no symbolic names for memory locations.
  - It is difficult to read.
  - It is machine-dependent making it difficult for portability.

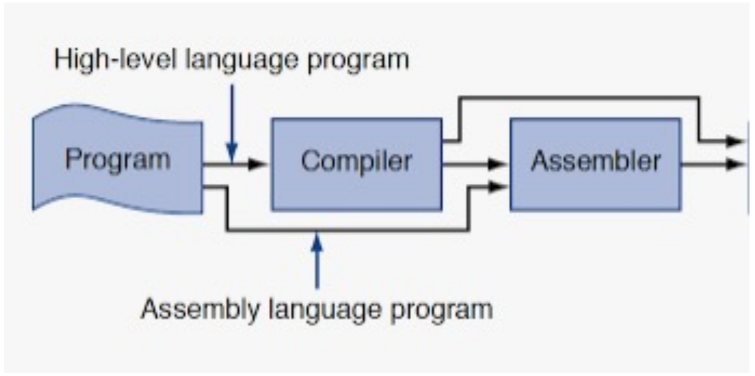
2.3 High-Level Language

High-level language is more like human language and less like machine language.

High-level languages are written in a form that is *close to our human language*, enabling programmers to just focus on the problem being solved.

High-level languages are *platform independent*

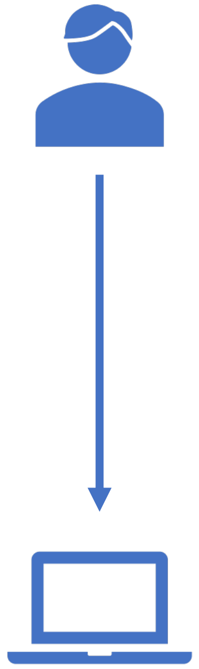
A program written in the high-level language is called *source program* or *source code* and is any collection of human-readable computer instructions.



2.3 High-Level Language

- Advantages
- Easier to modify, faster to write code and debug as it uses English like statements.
- Portable code, as it is designed to run on multiple machines.

Differences



• **Python Language**	a=b+c
• **Assembler Language**	LOAD b
	ADD c
	STRORE a
• **Machine Language**	1000001111
	1100111001
	0110001111

3. Programming Paradigm

A programming paradigm is a style, or "way" of programming.
Major programming paradigms are:

- **Imperative**
- **Logical**
- **Functional**
- **Object-Oriented**

3.1 Imperative Programming

- Imperative programming is a paradigm of computer programming in which the program describes a sequence of steps that change the state of the computer as each one is executed in turn.
- Imperative programming *explicitly tells the computer "how" to accomplish a certain goal.*
- *Structured* and *Procedural programming* are subset of Imperative programming.

3.2 Logical Programming

- Logical Programming uses basic rules expressed as *logical clauses* with a head and a body;

  "Y is true if X1, X2, and X3 are true."

- Facts are expressed similar to rules, but without a body;

  "Y is true."

- The idea in logical programming is that instead of telling the computer how to calculate things, *you tell it what things are*.

3.3 Functional Programming

- Functional Programming treat functions as first-class objects. In other words, you can pass a function as an argument to another function, or a function may return another function.

3.4 Object-Oriented Programming

- Object-oriented Programming approach is based on *objects* and *classes*.
- The object-oriented paradigm allows us to organize software as a collection of objects that consist of both data and behavior.
- This lets you have nice things like *encapsulation*, *inheritance*, and *polymorphism*.
- These properties are very important when programs become larger and larger.
- It provides key benefits of <u>reusable code</u> and <u>code extensibility</u>.

Program Language Execution Model

- *Compiling* and *interpreting* are two ways to run a computer program.
- The compilation translates all the instructions of a program into machine language, creating an executable file from the computer. The compilation is performed by compiler software.
- The interpretation translates and executes every single instruction of the program. Reads and executes the program's source code without creating an executable object file. It's slower than compiling

History of Python Programming Language

Python was conceived in the late 1980s and its implementation was started in December 1989 by Guido van Rossum (at CWI in the Netherlands as a successor to the ABC programming language capable of exception handling and interfacing with the Amoeba operating system.

Python language’s core philosophy

Its design philosophy emphasizes code readability

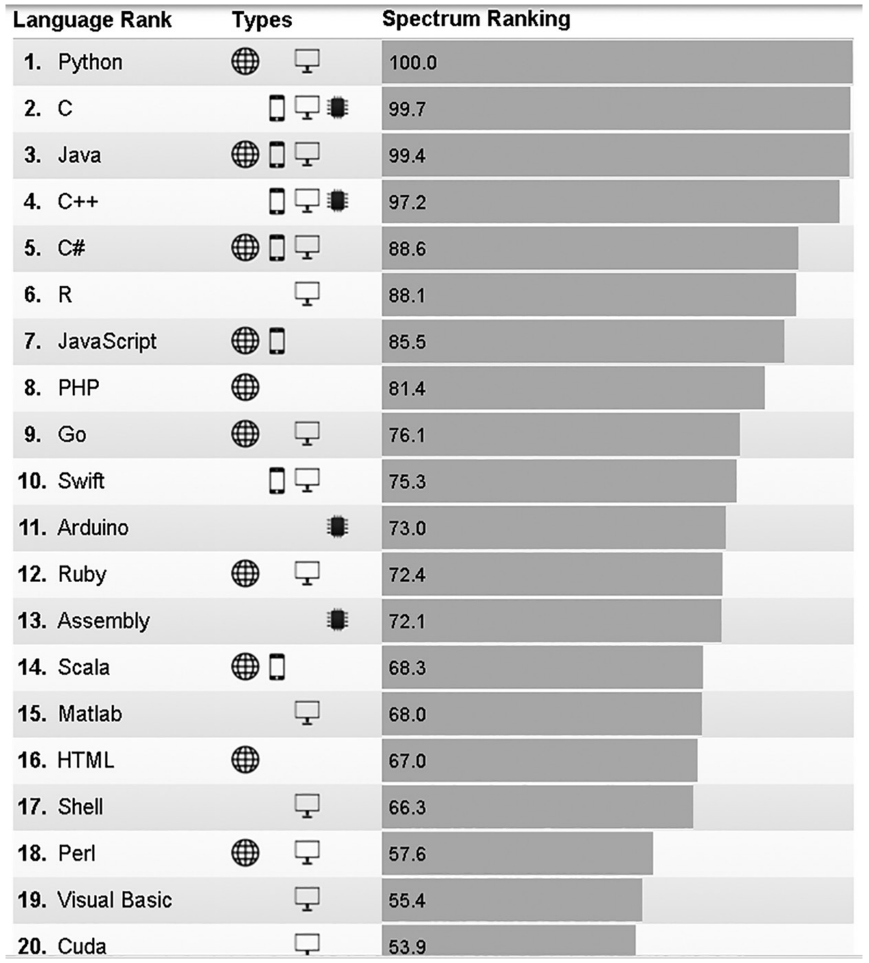
its syntax allows programmers to express concepts in fewer lines of code that would not be possible in languages such as C++ or Java.

The language provides constructs intended to enable clear programs on both a small and large scale.

Python is a multi-paradigm programming language having full support for Object-oriented programming and Structured programming and there are a number of language features which support Functional programming.

TOP 10

- Python has a solid claim to being the fastest-growing major programming language.
- Since 2003, Python has been consistently ranked in the top ten most popular programming



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥	100.0
2. C	📱 🖥 ▣	99.7
3. Java	🌐 📱 🖥	99.4
4. C++	📱 🖥 ▣	97.2
5. C#	🌐 📱 🖥	88.6
6. R	🖥	88.1
7. JavaScript	🌐 📱	85.5
8. PHP	🌐	81.4
9. Go	🌐 🖥	76.1
10. Swift	📱 🖥	75.3
11. Arduino	▣	73.0
12. Ruby	🌐 🖥	72.4
13. Assembly	▣	72.1
14. Scala	🌐 📱	68.3
15. Matlab	🖥	68.0
16. HTML	🌐	67.0
17. Shell	🖥	66.3
18. Perl	🌐 🖥	57.6
19. Visual Basic	🖥	55.4
20. Cuda	🖥	53.9

Scientific Tools

- Scientific tools are essential for simulating and analyzing complex systems.
- The Python ecosystem consists of these core scientific packages: *SciPy library, NumPy, Pandas, Matplotlib, etc.*.
- Most of these tools are available under Berkeley Software Distribution (BSD) license and can be used without any restrictions.

Python Scientific Libraries

1. **SciPy** library is mainly used for numerical integration and optimization.
2. **NumPy** provides N-dimensional array objects which can be used to perform linear algebra, Fourier transform and other mathematical operations.
3. **Jupyter** provides an interactive web-based interface which can be invoked from a browser. Jupyter is used to write Python programs and create embeddable plots to visualize data.
4. **SymPy** library is used to generate symbolic mathematics.
5. **Matplotlib** is the oldest and most popular plotting library available for Python

Machine Learning

- Machine Learning is an effective and adaptive tool to learn from experience and by a dataset.
- Many machine-learning algorithms and techniques have been developed that allow computers to learn.
- Scikit-Learn is a well-known Machine Learning tool built on top of other Python scientific tools like *NumPy*, *SciPy* and *Matplotlib*.
- Scikit-Learn supports various models for *Classification, Regression, Clustering, Model Selection, Dimensionality Reduction and Preprocessing*

Summarizing: What is Python?

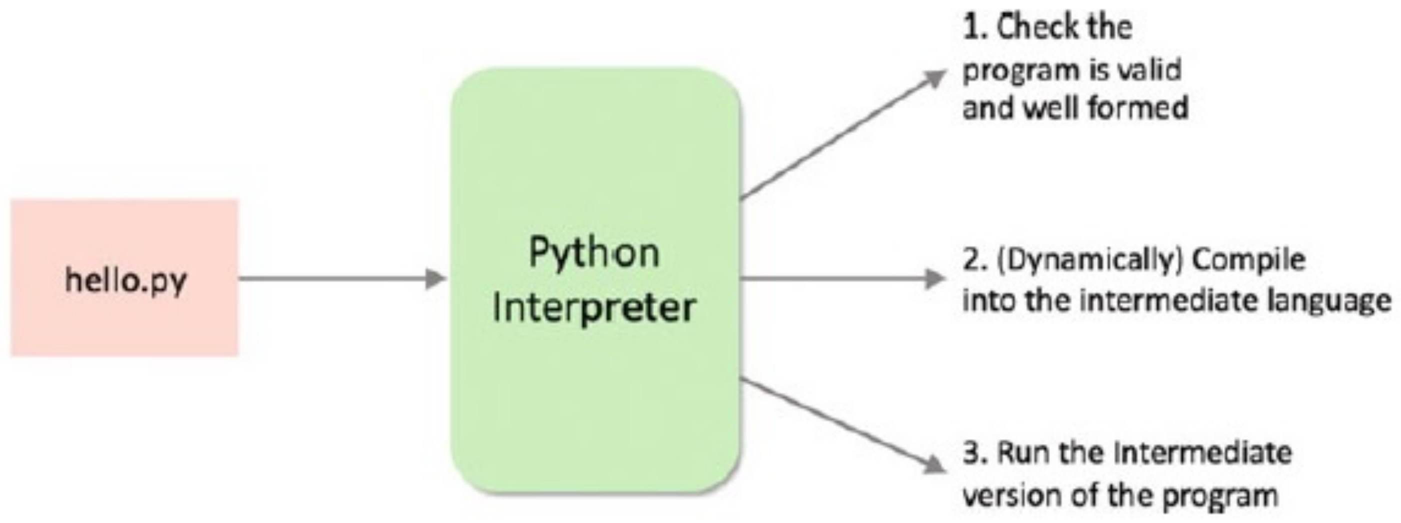
- Python is a free general-purpose programming language whose the history dates back to the late 1980s.
- It is *available across many platforms* including Windows, Linux and Mac OS.
- *Python versions*: Python 2 vs Python 3
- Python is a *multi-programming* language
- Several Python *modules* extend the functionality of the language and make it much easier to develop applications.

Why we should use Python?

- Python is readable
- Python is cross-platform language
- Python has the availability of a wide range of Libraries (modules)
- Python is free

Python Execution Model

- Python is not a precompiled language in the way that some other languages you may have come across are (such as C++). Instead it is what is known as an interpreted language (although even this is not quite accurate).


hello.py
Python Interpreter
1. Check the program is valid and well formed
2. (Dynamically) Compile into the intermediate language
3. Run the Intermediate version of the program

Running Python Programs

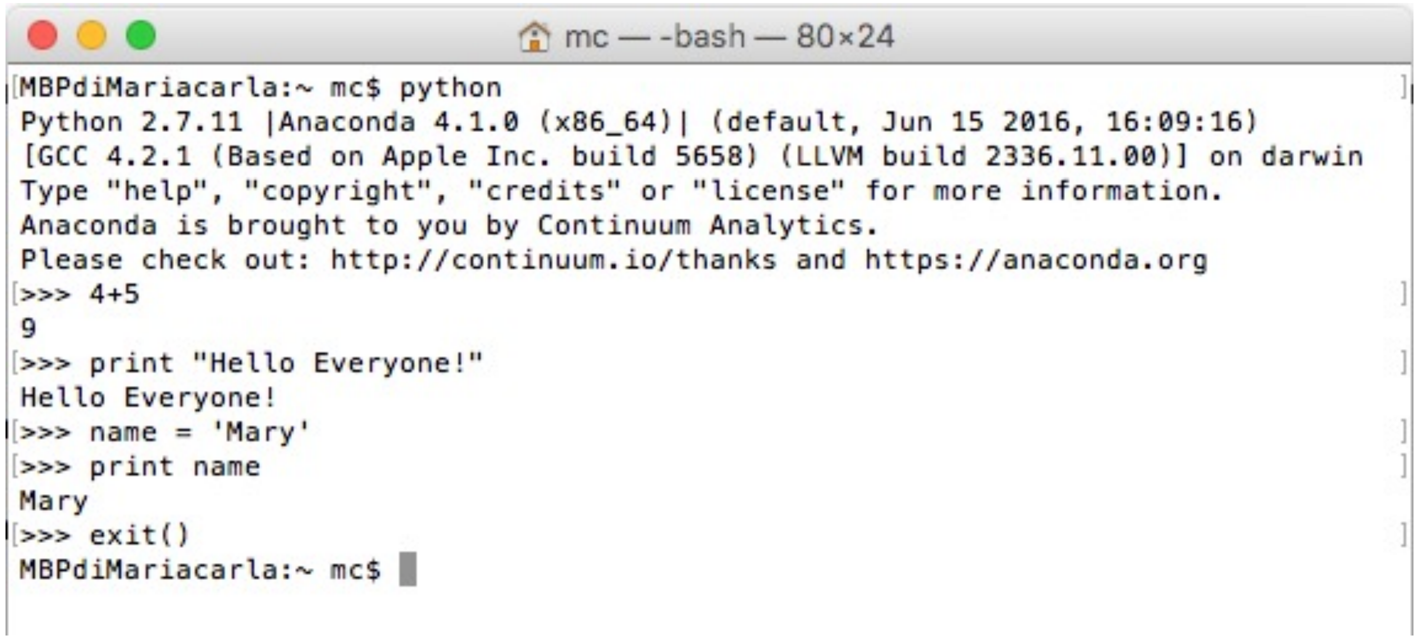
Interactively using the Python interpreter

Stored in a file and run using the *Python* command

Run as a script file specifying the Python interpreter to use within the script file

From within a Python IDE (Integrated Development Environment) such as PyCharm.

Interactively Using the Python Interpreter



```
MBPdiMariacarla:~ mc$ python
Python 2.7.11 |Anaconda 4.1.0 (x86_64)| (default, Jun 15 2016, 16:09:16)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> 4+5
9
>>> print "Hello Everyone!"
Hello Everyone!
>>> name = 'Mary'
>>> print name
Mary
>>> exit()
MBPdiMariacarla:~ mc$
```

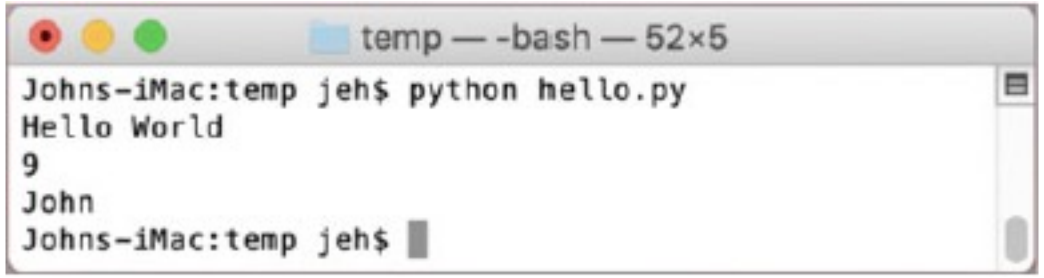
The Python in interactive mode also known as the Python REPL (Read Evaluate Print Loop style of operation) permits

Python statements and expressions

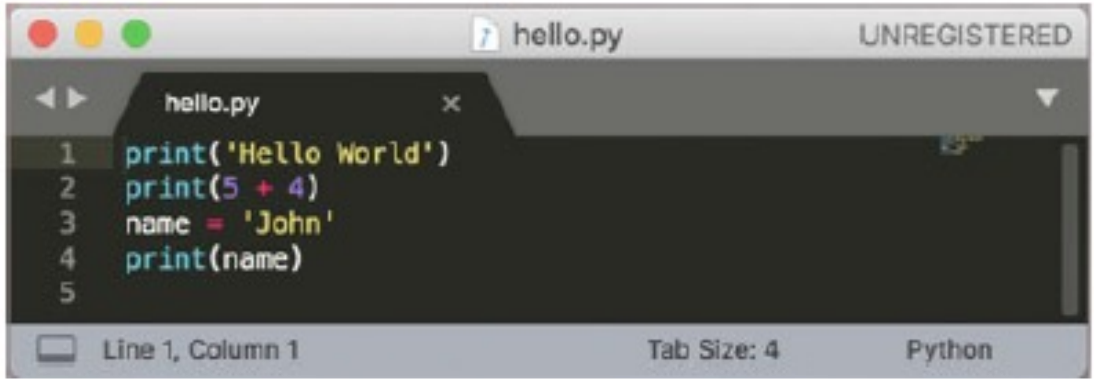
can be typed into the Python prompt and will then be executed directly.

Running a Python File

- We can store the Python commands into a file. This creates a program file that can then be run as an argument to the python command.
- To run the hello.py program we can use the python command followed by the name of the fi le:

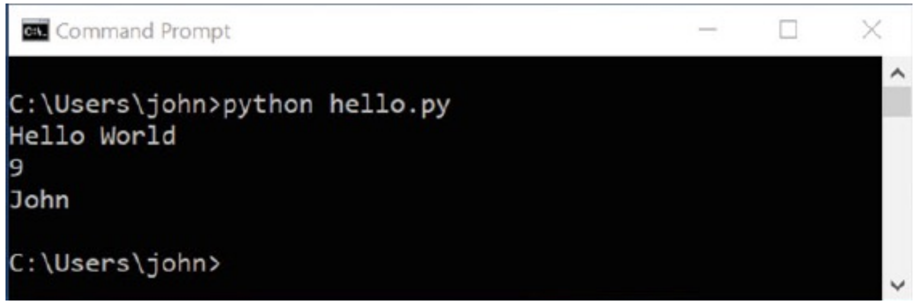

temp — -bash — 52×5

```
Johns-iMac:temp jeh$ python hello.py
Hello World
9
John
Johns-iMac:temp jeh$
```


hello.py UNREGISTERED

```python
print('Hello World')
print(5 + 4)
name = 'John'
print(name)
```

Line 1, Column 1 Tab Size: 4 Python


Command Prompt

```
C:\Users\john>python hello.py
Hello World
9
John

C:\Users\john>
```

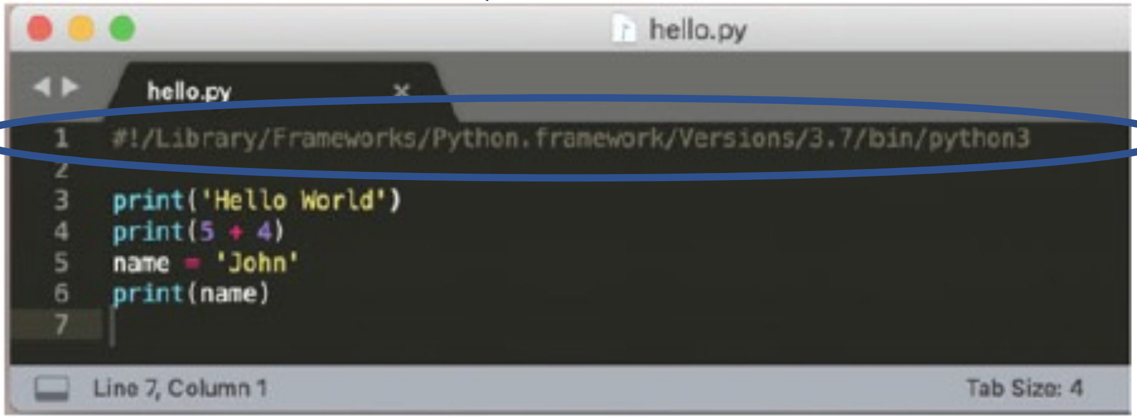
Executing a Python Script

- It is also possible to transform a file containing a stored Python program into a Script.
- A script is a stand-alone file that can be run directly without the need to (explicitly) use the python command.
- This is done by adding a special line to the start of the Python file that indicates the Python command (or interpreter) to use with the rest of the file.
- *This line must start with '#!' and must come at the start of the file*

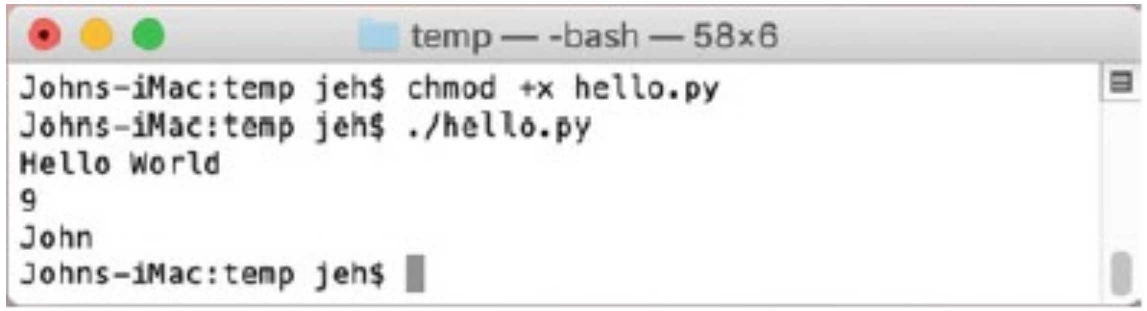
Executing a Python Script (Mac/Linux)

- To convert the previous section' s file into a Script we would need to add the path to the python interpreter.
- However, we cannot just run the file as it stands. If we tried to run the file without any changes then we will get an error indicating that the permission to execute the fi le has been denied:

```
$ ./hello.py
-bash: ./hello.py: Permission denied
$ chmod +x hello.py
```



```
#!/Library/Frameworks/Python.framework/Versions/3.7/bin/python3

print('Hello World')
print(5 + 4)
name = 'John'
print(name)
```

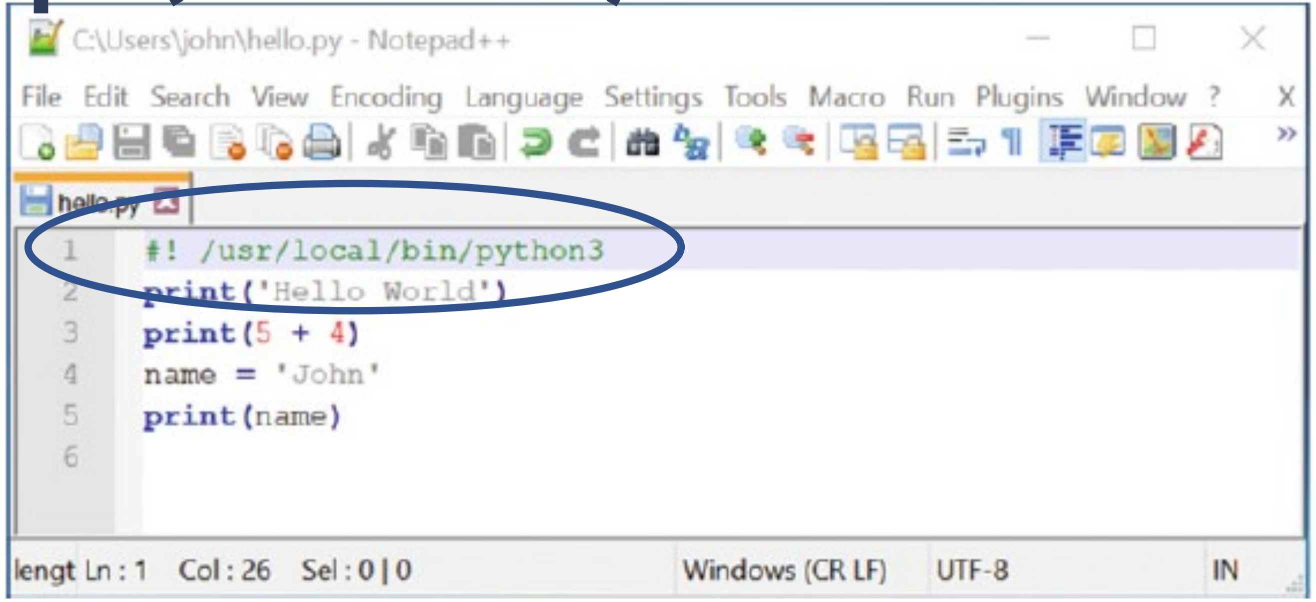


```
Johns-iMac:temp jeh$ chmod +x hello.py
Johns-iMac:temp jeh$ ./hello.py
Hello World
9
John
Johns-iMac:temp jeh$
```

Note the use of the './' preceding the file name

Executing a Python Script (Windows)

- By default Windows does not have the same concept. However, to promote cross platform portability, the Python Launcher for Windows can also support this style of operation.
- It allows scripts to indicate a preference for a specific Python version using the same


C:\Users\john\hello.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
hello.py

```
#! /usr/local/bin/python3
print('Hello World')
print(5 + 4)
name = 'John'
print(name)

```

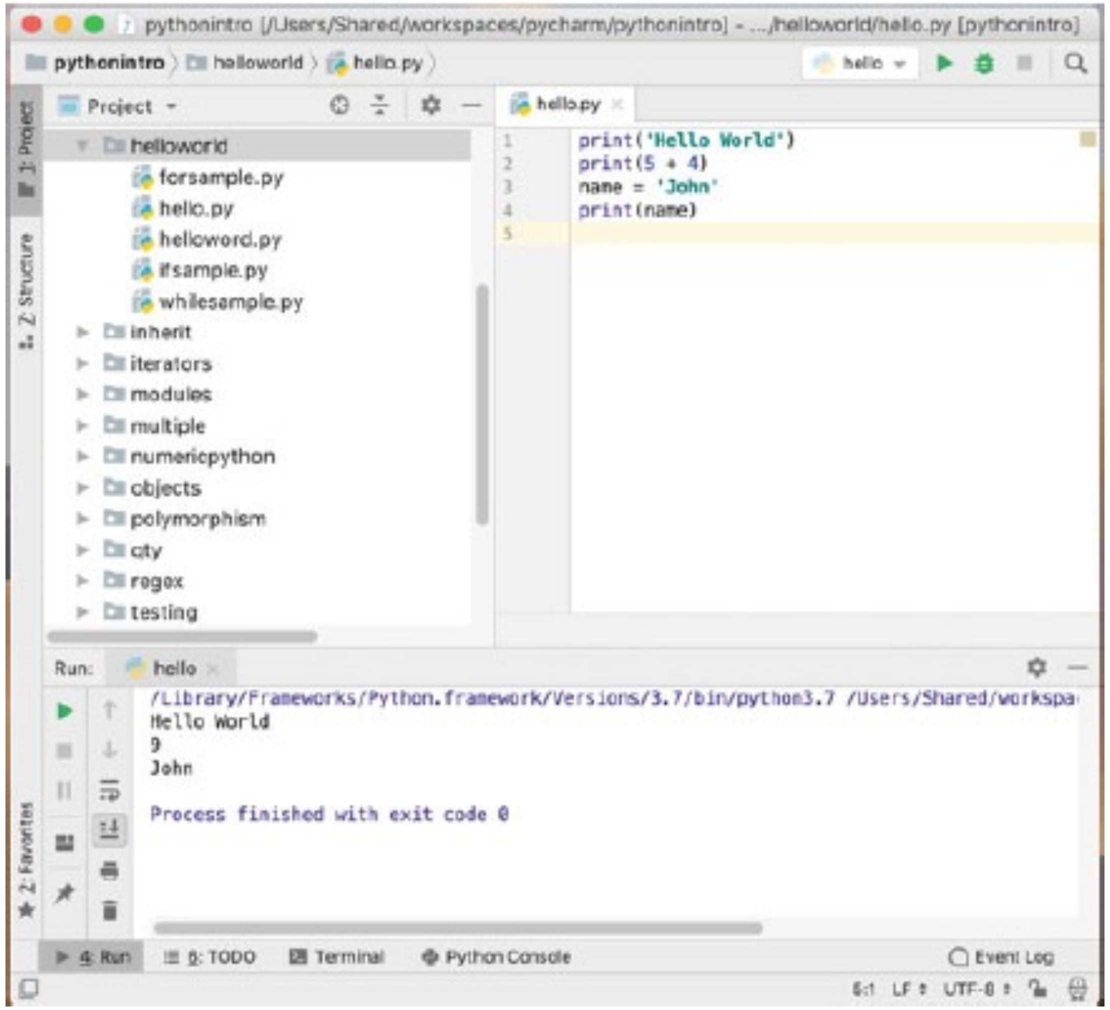
lengt Ln : 1 Col : 26 Sel : 0 | 0 Windows (CR LF) UTF-8 IN

#!user/local/bin/python3

will be interpreted as indicating that python3 is required

Using Python in an IDE

- We can also use an **IDE** such as PyCharm to writing and execute our Python program.
- The same program is shown using PyCharm below:



```
print('Hello World')
print(5 + 4)
name = 'John'
print(name)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/Shared/workspa
Hello World
9
John

Process finished with exit code 0
```

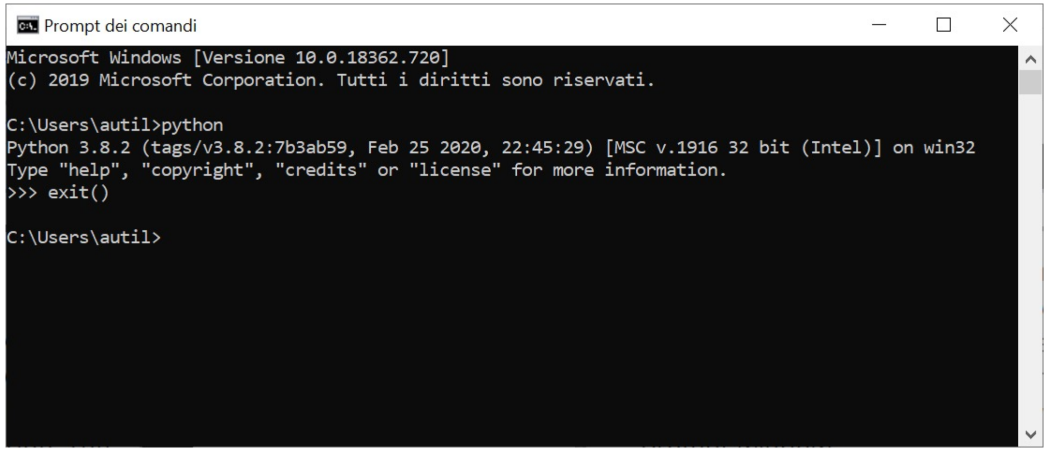
Setting Up the Python Environment

Check to See If Python Is Installed

<u>On a Windows</u> machine you can check the version installed by opening a Command Prompt window (this can be done by searching for Cmd in the 'Type here to search' box in Windows 10).

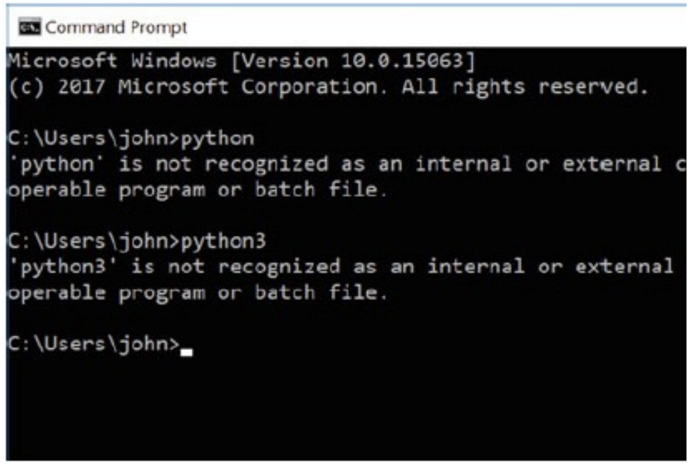
Once the Command window is open try typing in python .

Installed



```
Microsoft Windows [Versione 10.0.18362.720]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\autil>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\autil>
```

Not Installed

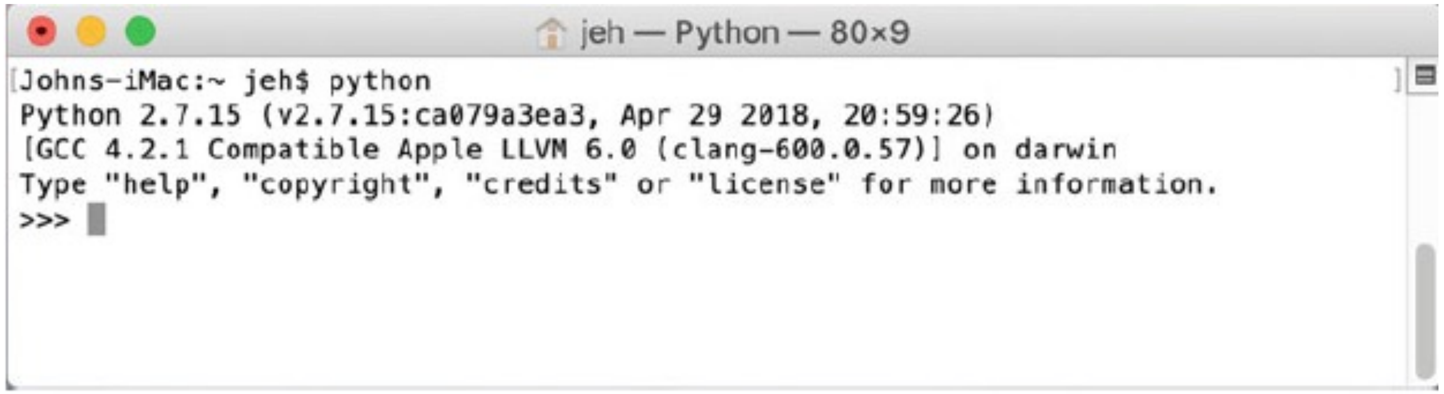


```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\john>python
'python' is not recognized as an internal or external c
operable program or batch file.

C:\Users\john>python3
'python3' is not recognized as an internal or external
operable program or batch file.

C:\Users\john>_
```

Check to See If Python Is Installed

On a Mac you can use the Terminal and do the same thing. You will probably find that at least python (2) is pre-installed for you. For example, if you type in python on a Mac you will get something like this

Use quit() or exit() to exit the Python interpreter; exit() is an alias for quit() and is provided to make Python easier to use.

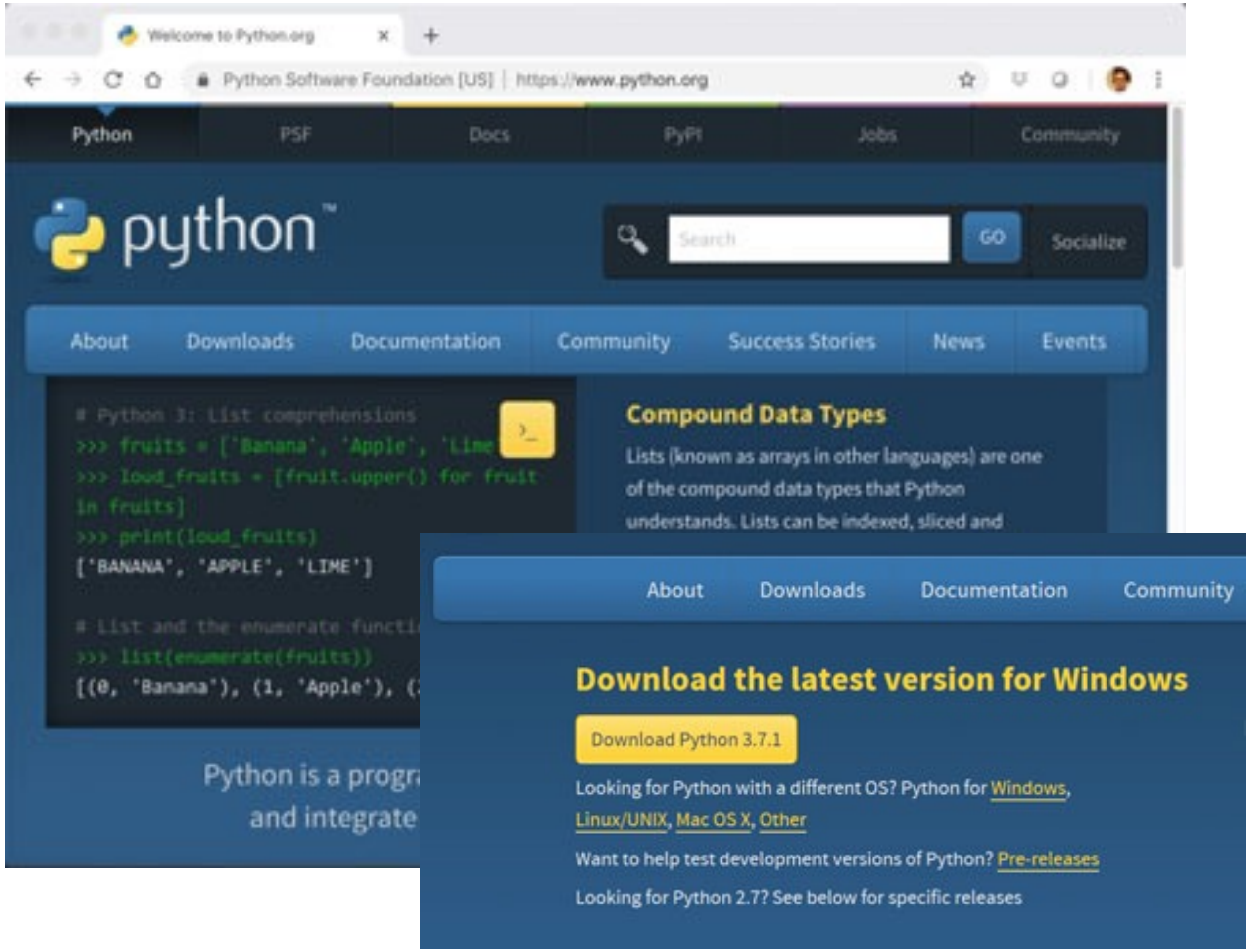


```
Johns-iMac:~ jeh$ python
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 29 2018, 20:59:26)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Installing Python on a Windows PC

Step 1: Downloading Python

- Python is available for a wide range of platforms from Windows, to Mac OS and Linux; you will need to ensure that you download the version for your operating system.
- Python can be downloaded from the main Python web site which can be found at http://www.python.org



```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime
>>> loud_fruits = [fruit.upper() for fruit
in fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate functi
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (
```

Compound Data Types

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and

Python is a progr
and integrate

Download the latest version for Windows

Download Python 3.7.1

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

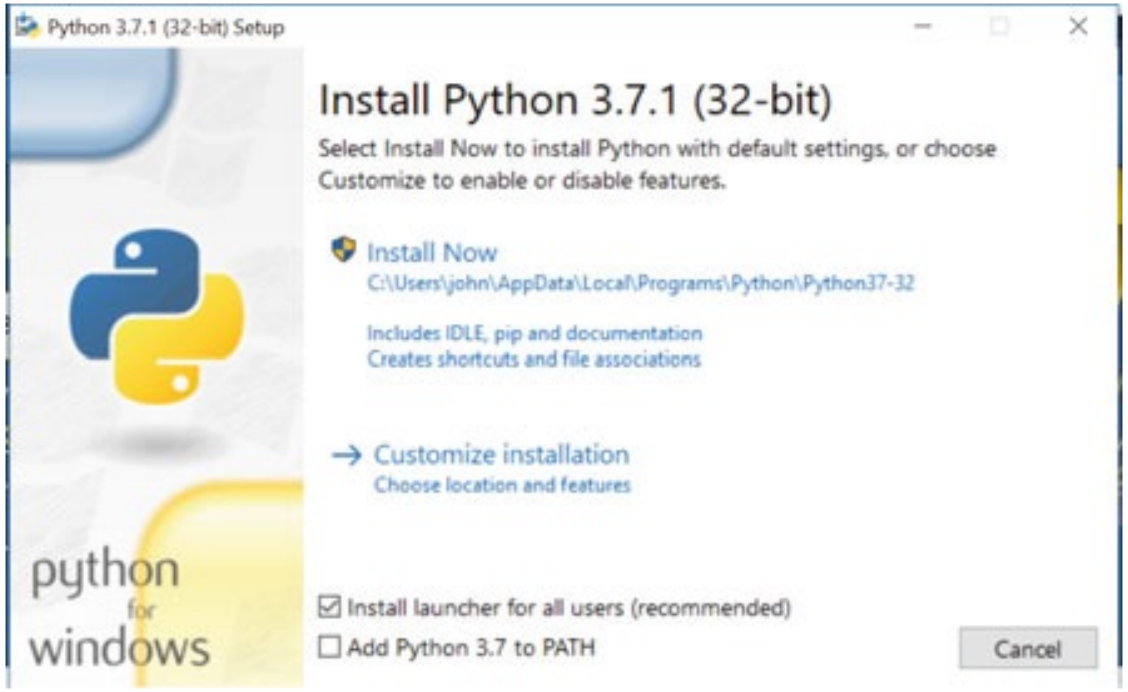
Want to help test development versions of Python? Pre-releases

Looking for Python 2.7? See below for specific releases

Installing Python on a Windows PC

Step 2: Running the Installer

- You will now be prompted for the location to install Python, for example:



Python 3.7.1 (32-bit) Setup

Install Python 3.7.1 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\john\AppData\Local\Programs\Python\Python37-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

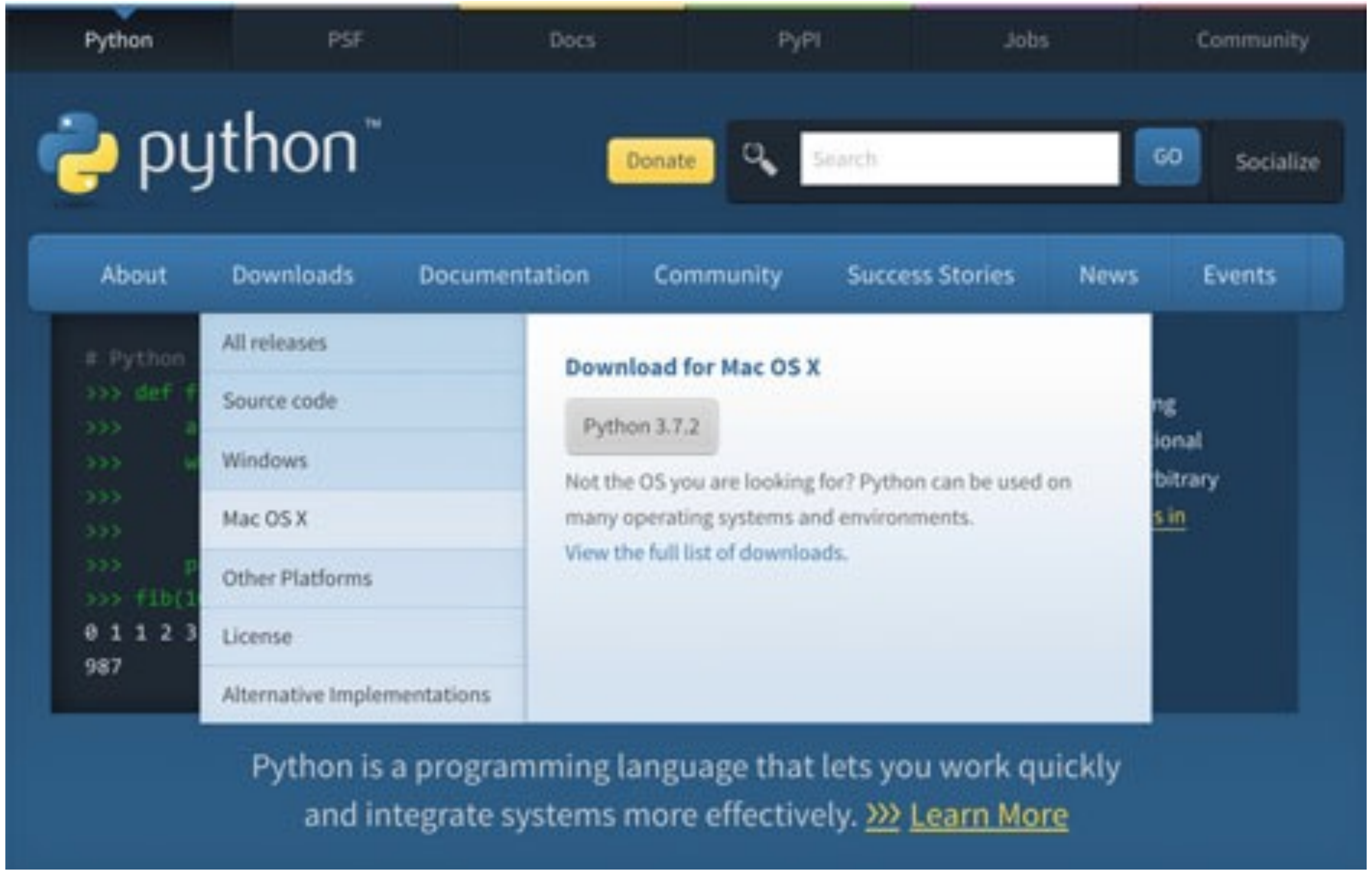
Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.7 to PATH

Installing Python on a Mac

Step 1: Downloading Python

- Installing Python on a Mac is similar to installing it on a Windows machine in that you can download the Python installer for the Apple Mac from the Python Software
- Foundation web site (https://www.python.org).



Download for Mac OS X

Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of downloads.

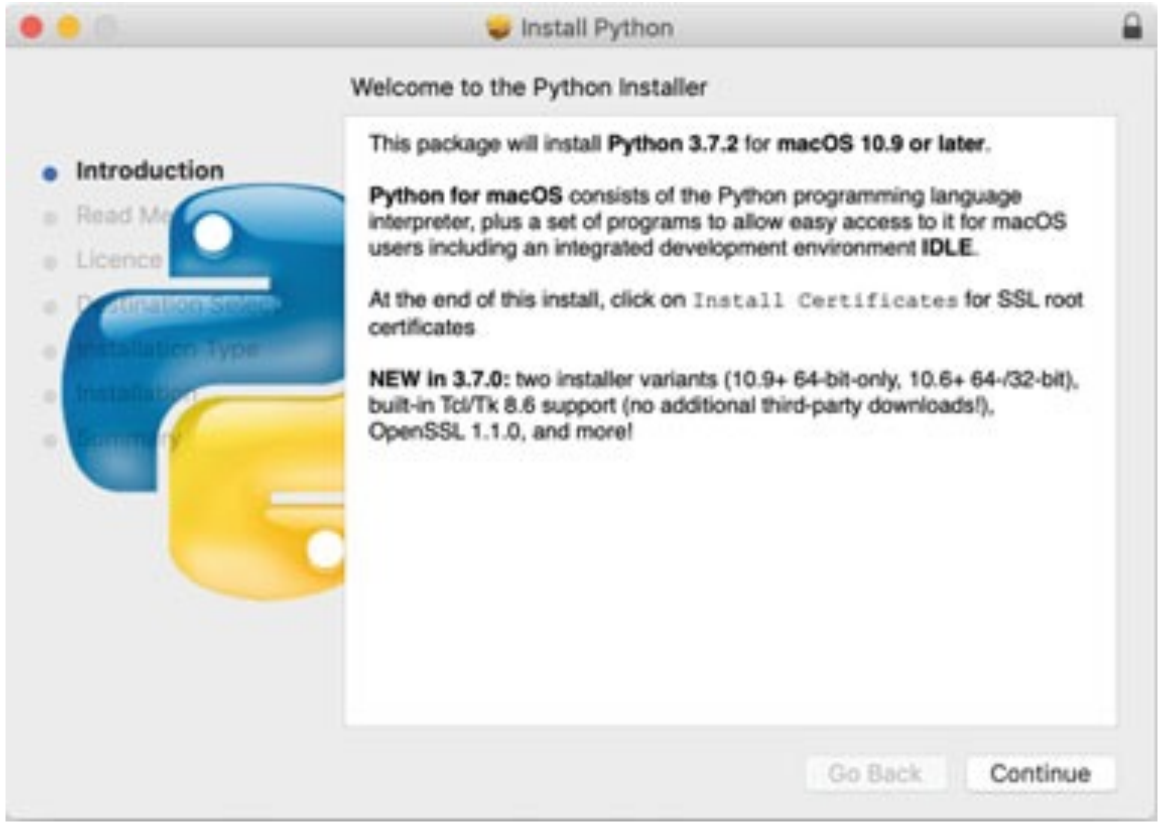
```
# Python
>>> def f
>>>     a
>>>     w
>>>
>>>
>>>     p
>>> fib(1
0 1 1 2 3
987
```

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Installing Python on a Mac

Step 2: Running the Installer

- You will now be prompted for the locatio From here you can select the macOS 64-bit installer (make sure you select the appropriate one for your version of the operating system). This will download an
- Apple package that can be installed (it will have a name similar to python-3.7.2-macos10.9.pkg although the version number of Python and the Mac OS operating system may be different).
- You will need to run this installer.



Welcome to the Python Installer

This package will install **Python 3.7.2** for **macOS 10.9 or later**.

Python for macOS consists of the Python programming language interpreter, plus a set of programs to allow easy access to it for macOS users including an integrated development environment **IDLE**.

At the end of this install, click on `Install Certificates` for SSL root certificates

NEW in 3.7.0: two installer variants (10.9+ 64-bit-only, 10.6+ 64-/32-bit), built-in Tcl/Tk 8.6 support (no additional third-party downloads!), OpenSSL 1.1.0, and more!

Installing Python on a Linux

Python version 2.x is already installed in most Linux distributions but as we're going to be using Python 3.x, there's a little work we need to do first to get hold of it.

1) Open a Linux Terminal

2) If you have never built any software on your system before, you must install the build essentials, SQLite, and bzip2 or the Python installation will fail.. Therefore, you need to type these commands into the Terminal (press keyboard button Enter after every command)

```
sudo apt-get install build-essential
sudo apt-get install libsqlite3-dev
sudo apt-get install libbz2-dev
```

Installing Python on a Linux

3) open your web browser to www.python.org/downloads/ and look for the button detailing the download link for Python 3.x.x (the current version is Python 3.8.2) to download the source Python-3.8.2.tar.xz file.

4) In the Terminal, go the Downloads folder by entering: cd Downloads/. Then unzip the contents of the downloaded Python source code with: tar -xvf Python-3.6.2.tar.xz.

5) Now enter the newly unzipped folder with cd Python-3.6.2/.

6) Within the Python folder, enter:

```
./configure

sudo make altinstall
```

7) check if the installation was really successful. Open the Terminal and enter the command python3 and press keyboard button Enter

Python Distribution

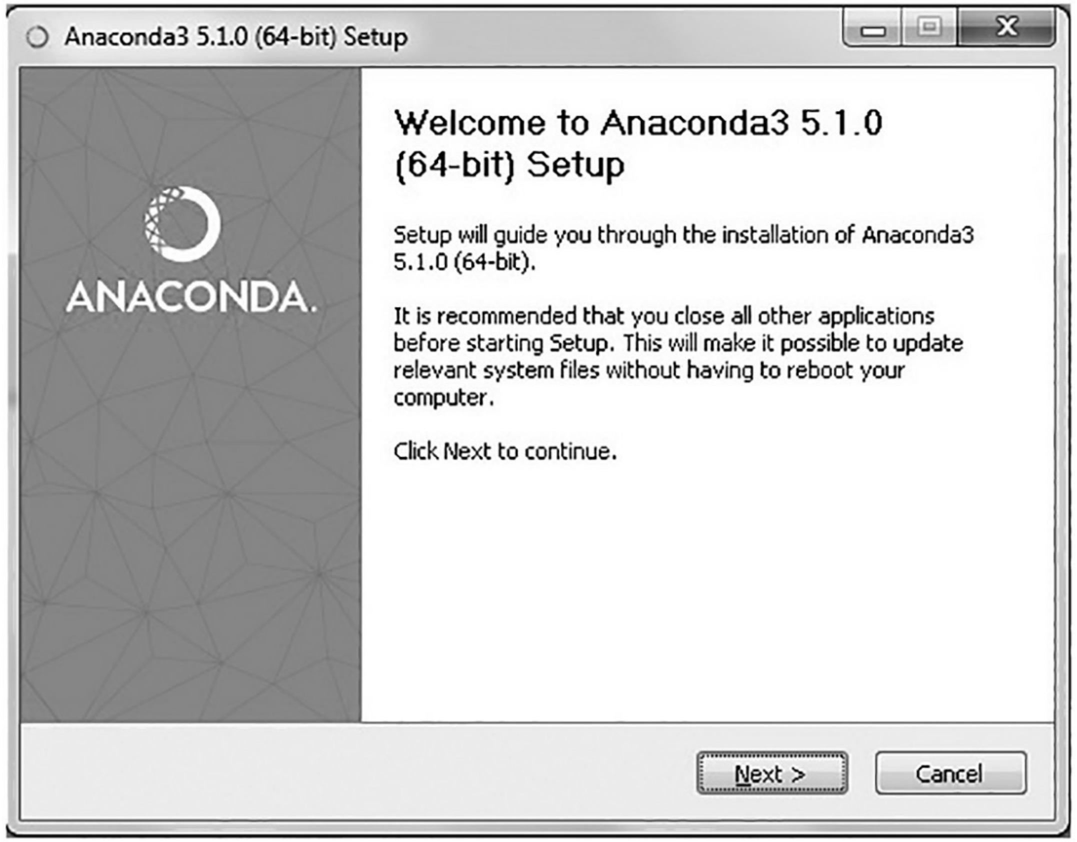
- Python Software Foundation releases Python interpreter with standard libraries. But in order to use Python in a scientific or enterprise environment other packages needs to be installed. Having these packages tested for compatibility with the latest release of Python is cumbersome and time-consuming.
- Anaconda and Enthought Canopy Express are two popular distributions that come with core Python interpreter and popular scientific tools to help us start working out of the box.

Installing Anaconda Python Distribution

- Anaconda is a free and open source distribution of the Python programming language for data science and machine-learning related applications such as large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment.
- Package versions are managed by the package management system conda, which makes it quite simple to install, run, and update complex data science and machine learning software libraries like Scikit-learn, PyTorch, TensorFlow, and SciPy.

Installing: step by step

- **Step 1:** Go to the link https://www.anaconda.com/distribution/#macos You have the option to download the 64-bit version of Python 3 supported Anaconda distribution.
- **Step 2:** Click on the executable file of Anaconda Python distribution which you have downloaded and the setup screen will start loading.
- **Step 3:** You will get a welcome screen as shown in FIGURE 1.5 . Click on Next button


Anaconda3 5.1.0 (64-bit) Setup

Welcome to Anaconda3 5.1.0 (64-bit) Setup

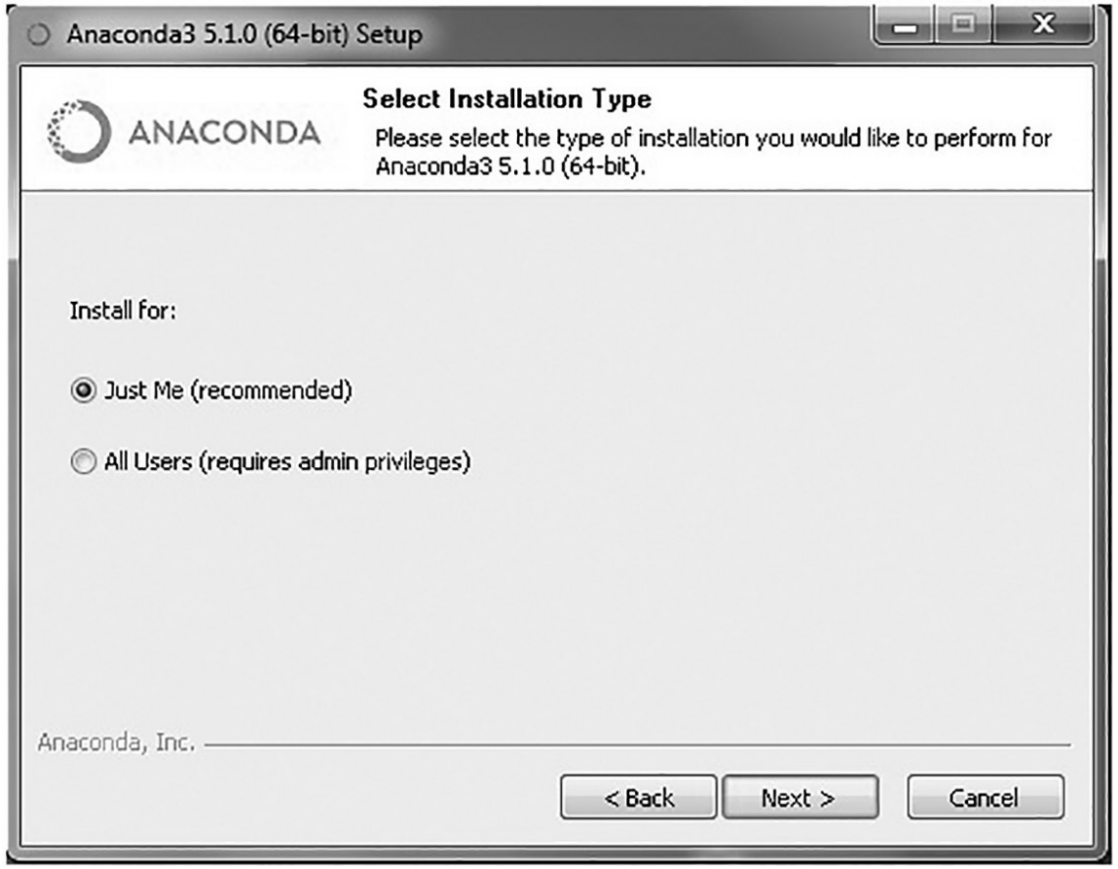
Setup will guide you through the installation of Anaconda3 5.1.0 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Installing: step by step

- Step 4: You will get a License Agreement Screen, read the licensing terms and click on I Agree button.
- Step 5: Assuming that you are the only user on your system, select Just Me radio button. Click on Next button.



Anaconda3 5.1.0 (64-bit) Setup

ANACONDA

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 5.1.0 (64-bit).

Install for:

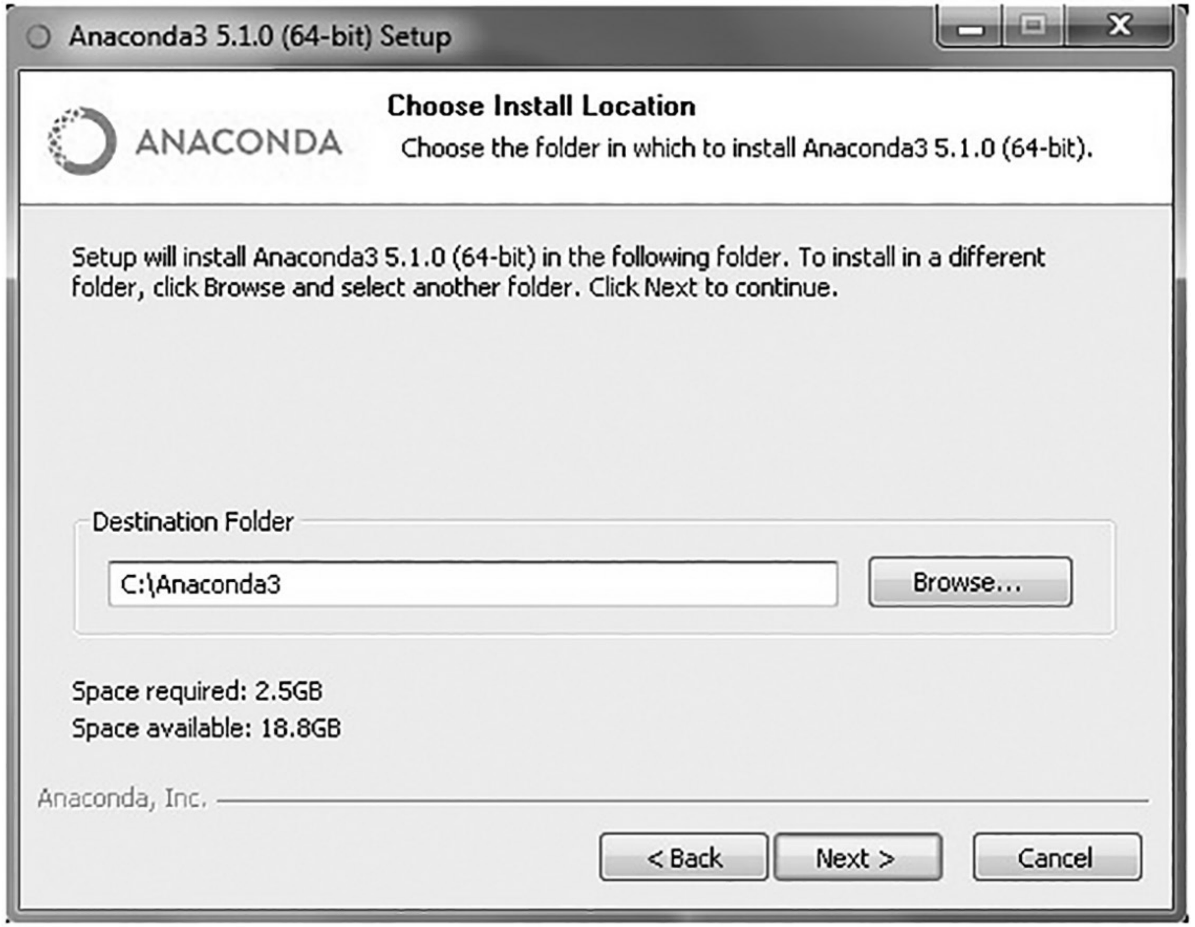
[✓ Just Me (recommended)]

All Users (requires admin privileges)

Anaconda, Inc.

Installing: step by step

- **Step 6:** You need to choose a location to install Anaconda Python distribution. The default installation will be under Users folder. Change the destination folder to C:\Anaconda3 to install Anaconda and click on Next button


Anaconda3 5.1.0 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 5.1.0 (64-bit).

Setup will install Anaconda3 5.1.0 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

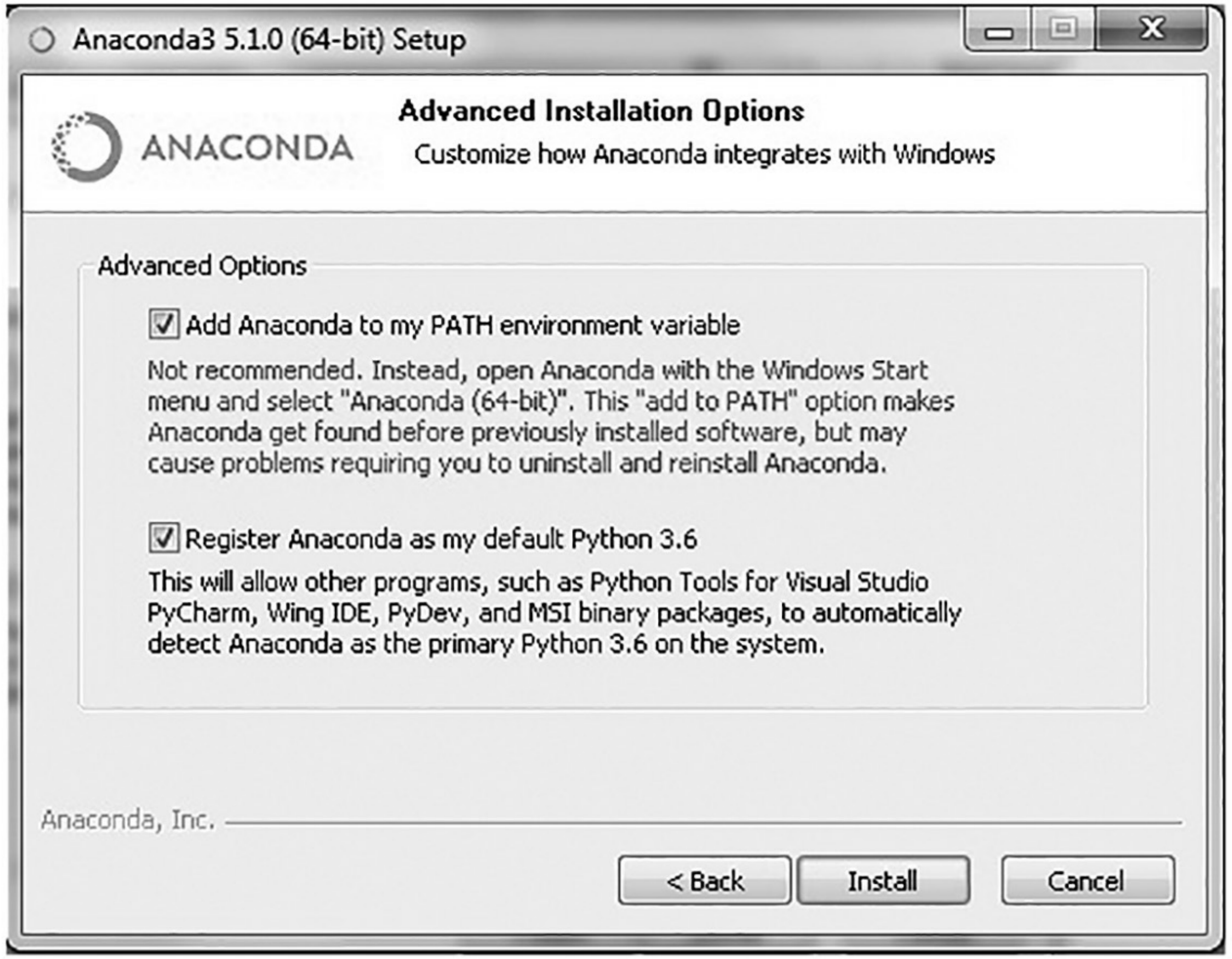
Destination Folder: C:\Anaconda3

Space required: 2.5GB
Space available: 18.8GB

Anaconda, Inc.

Installing: step by step

- **Step 7:** In the Advanced Installation Options screen, select all the check boxes. Ignore the warnings and click on Install button



Anaconda3 5.1.0 (64-bit) Setup

Advanced Installation Options
Customize how Anaconda integrates with Windows

Advanced Options

[✓] Add Anaconda to my PATH environment variable

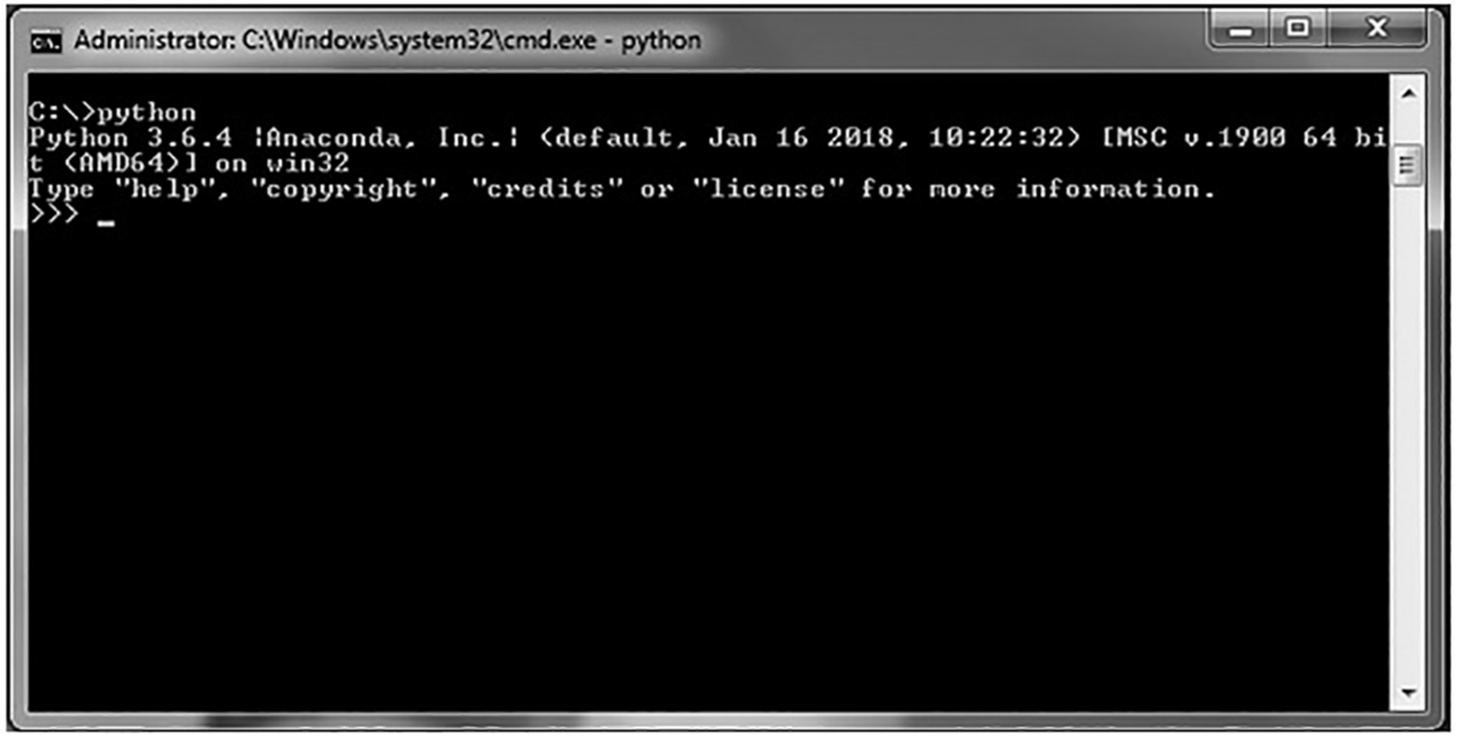
Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

[✓] Register Anaconda as my default Python 3.6

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.6 on the system.

Installing: step by step

- Step 8: This starts the installation of Anaconda Python Distribution and once the installation is complete, click on Next button.
- Step 9: Finish the setup by clicking on Finish button.
- Step 10: To check whether the installation is working properly or not, go to the command prompt and type python



```
C:\>python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Integrated Development Environments (IDEs)

Integrated Development Environments (IDEs) help in the rapid development of the software and increase in productivity.

PyCharm is the most popular IDE for Python

It has advanced features like auto code completion, code highlighting, refactoring, remote development capabilities etc.

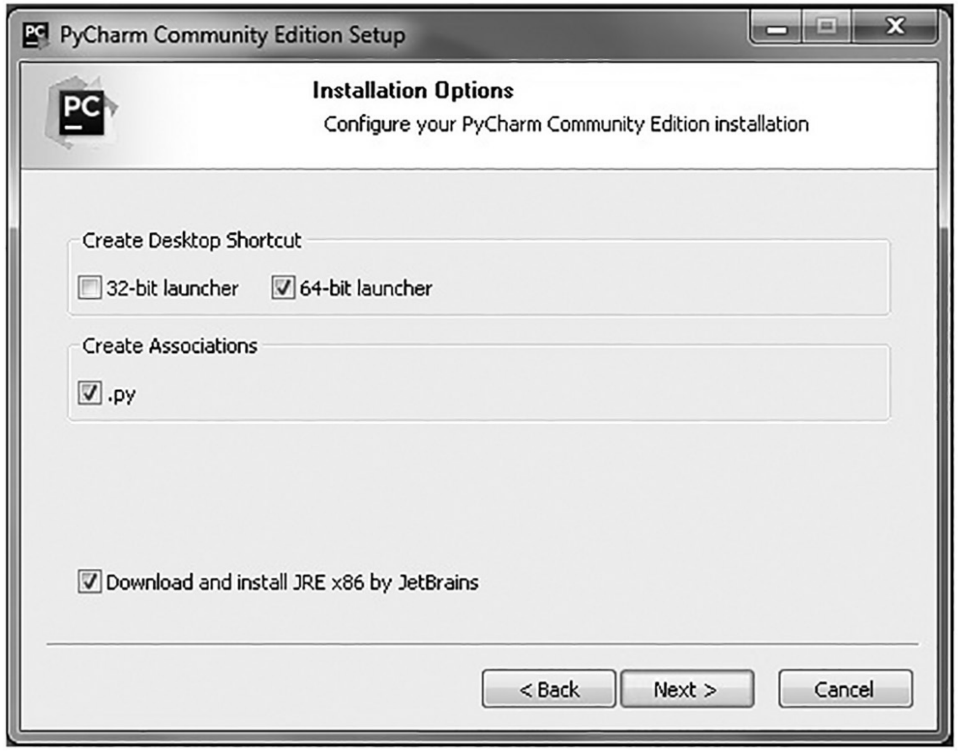
It is available for various platforms like Windows, Linux and OS X.

Installing PyCharm IDE to Set Up a Python Development Environment

- **PyCharm** is an Integrated Development Environment (IDE) used for Python programming language. It is developed by the Czech company JetBrains. I
- t provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems and supports web development with Django web framework.
- PyCharm is cross-platform, with the availability of Windows, MacOS and Linux versions.

Installing: step by step

- Step 1: Go to the link https://www.jetbrains.com/pycharm/download/#section=mac and download the Community edition
- Step 2: Click on the executable file which you have downloaded. You will be presented with the PyCharm Community Edition Setup screen. Click on Next button.
- Step 3: Now you will be presented with Choose Install Location screen. Go with the default destination folder to install PyCharm Community Edition. Click on Next button.
- Step 4: Select all the check boxes in the Installation Options screen except for the 32-bit launcher check box (since Windows 10 is a 64-bit OS, you don't need 32-bit launcher) and click on Next


PyCharm Community Edition Setup

Installation Options

Configure your PyCharm Community Edition installation

Create Desktop Shortcut

☐ 32-bit launcher ☑ 64-bit launcher

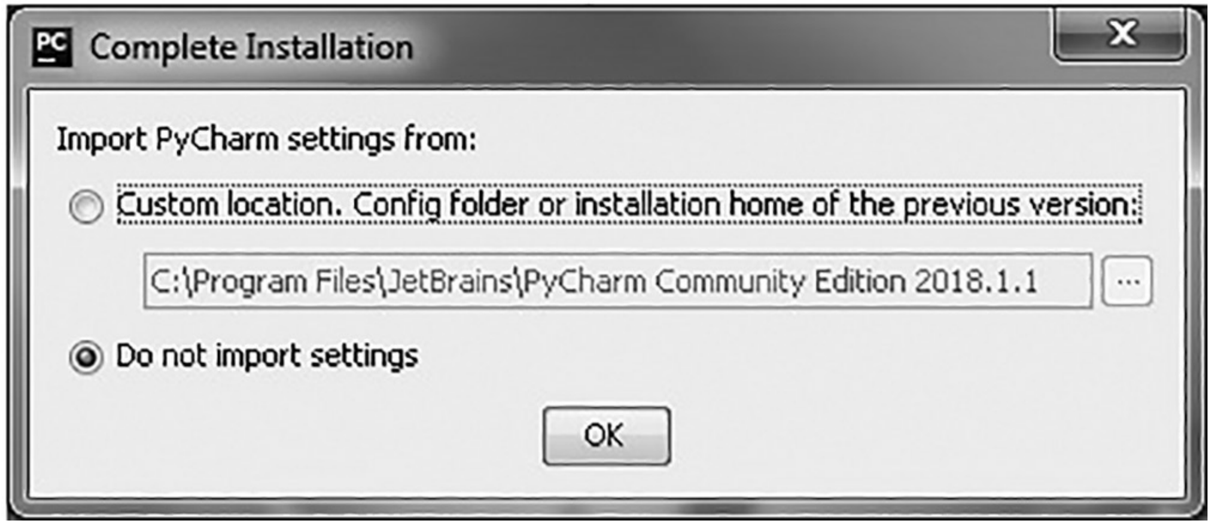
Create Associations

☑ .py

☑ Download and install JRE x86 by JetBrains

Installing: step by step

- Step 5: Go with the default Start Menu Folder as shown on the screen and click on Install button. It will take some time for the installation to finish. Once the installation is done click on the Finish button.
- Step 6: You will be asked whether you want to import previous PyCharm settings. Since we are starting on a clean slate, let's select the second radio button as shown in FIGURE 1.11 and click on OK button.


Complete Installation

Import PyCharm settings from:

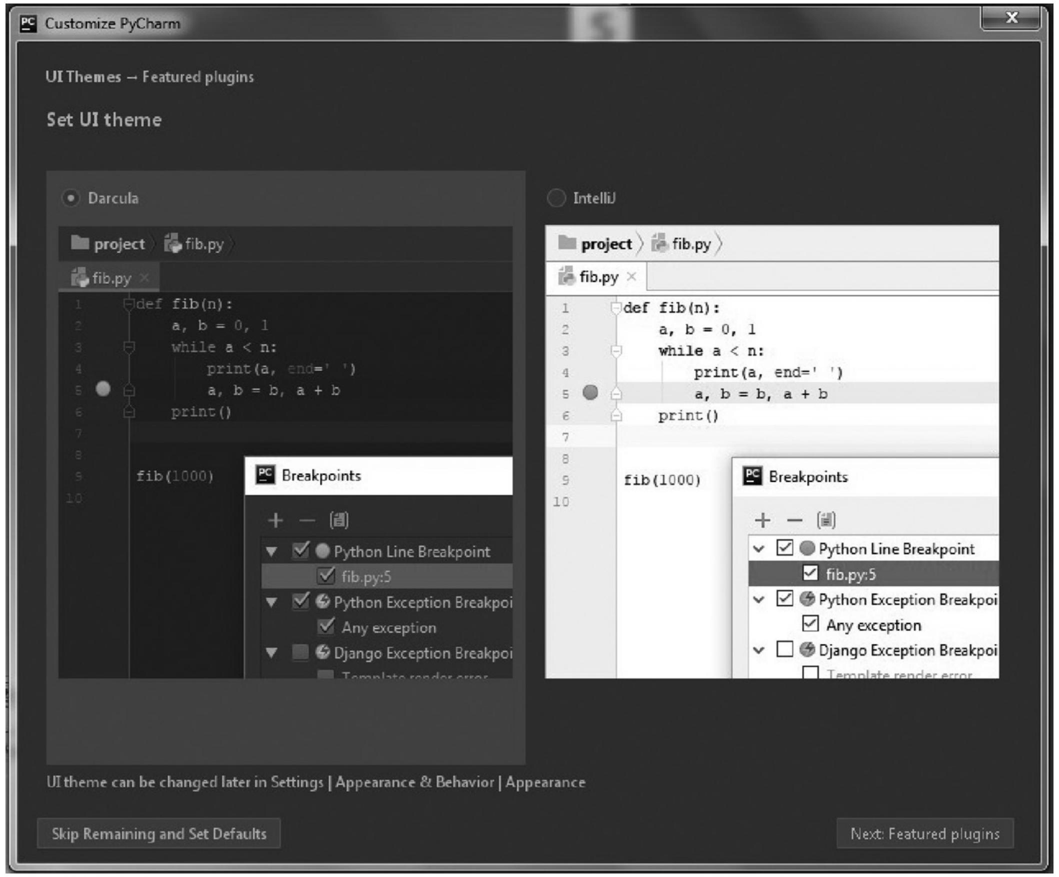
[ ] Custom location. Config folder or installation home of the previous version:

C:\Program Files\JetBrains\PyCharm Community Edition 2018.1.1

[✓ Do not import settings]

Installing: step by step

- **Step 7:** You will be prompted with a Windows Security Alert. Do not worry about it. Click on Allow Access Button. Next screen will be Python community edition initial configuration. Let the default settings remain as it is and click on OK button. After you have completed initial PyCharm configuration, a customization screen will be displayed; click on Skip Remaining and Set Defaults button



Customize PyCharm

UI Themes → Featured plugins

Set UI theme

● Darcula ○ IntelliJ

project 〉 fib.py 〉

fib.py ×

```
def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a + b
    print()

fib(1000)
```

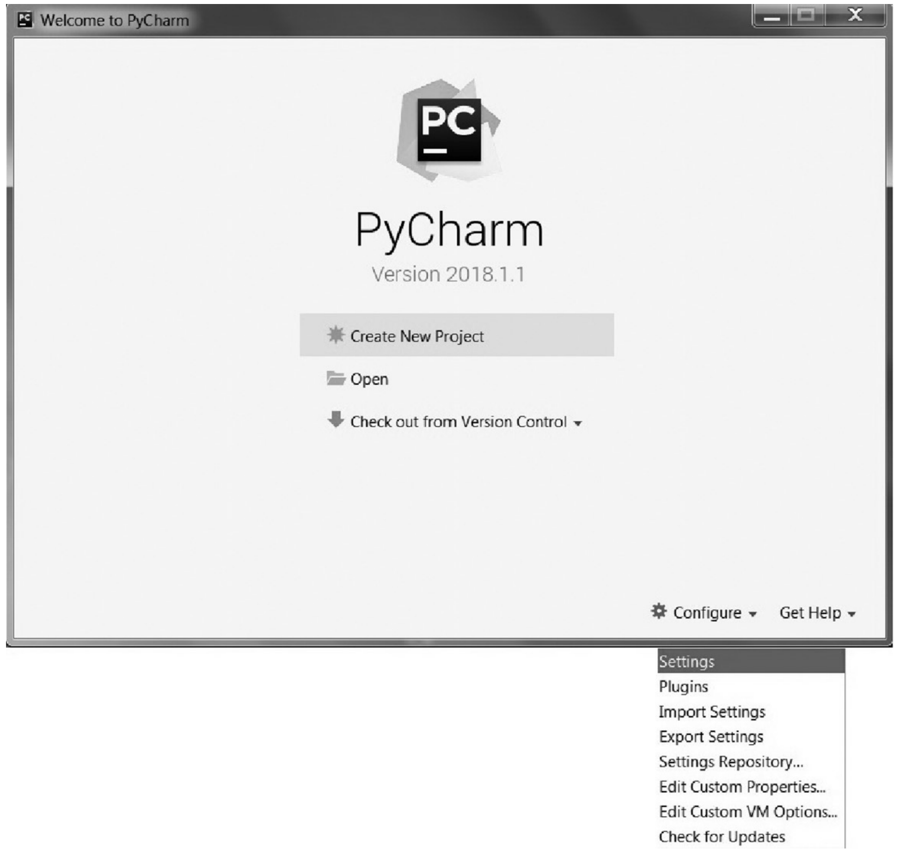
Breakpoints

- ☑ Python Line Breakpoint
  - ☑ fib.py:5
- ☑ Python Exception Breakpoint
  - ☑ Any exception
- ☐ Django Exception Breakpoint
  - ☐ Template render error

UI theme can be changed later in Settings | Appearance & Behavior | Appearance

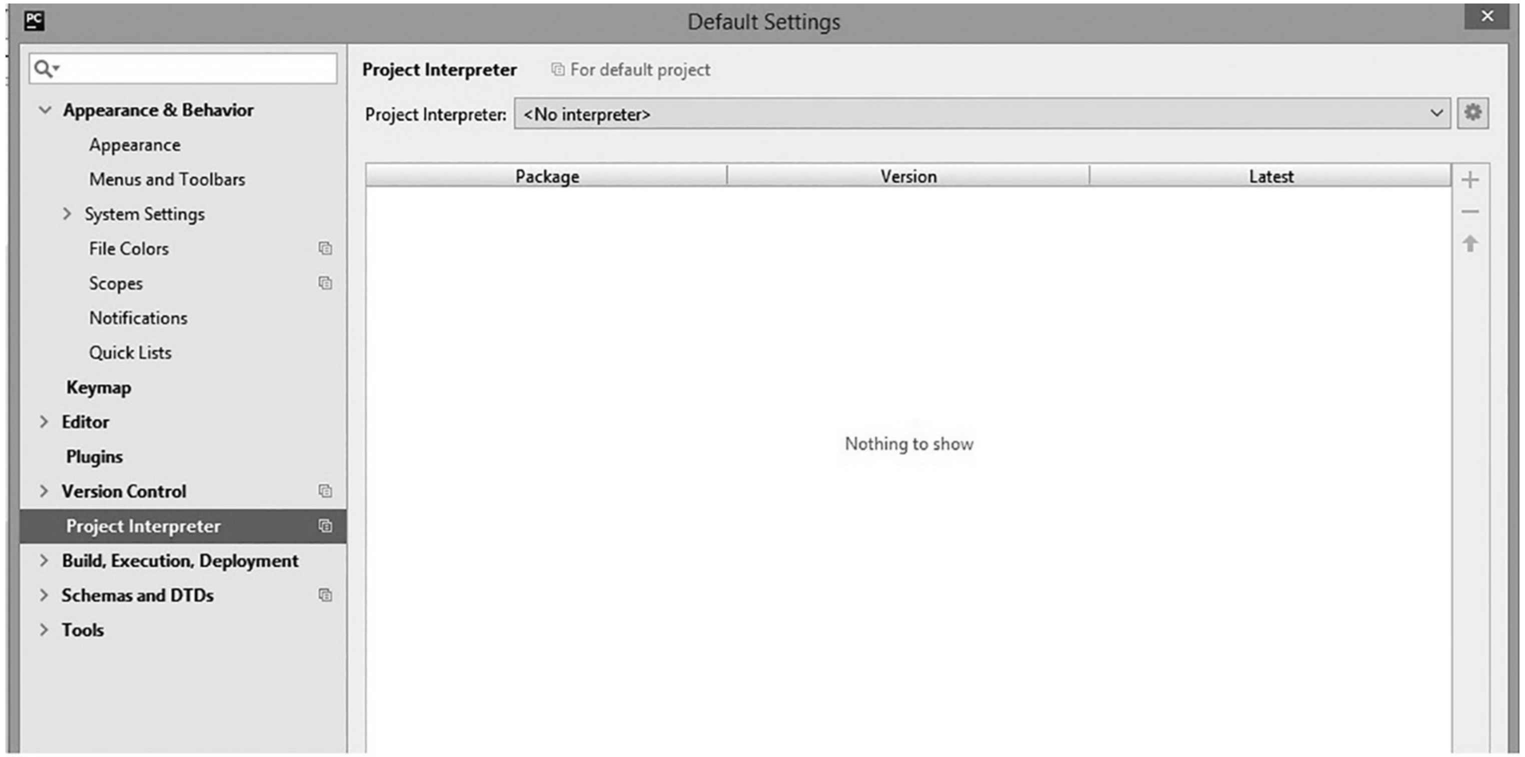
Installing: step by step

- **Step 8:** In the next screen, click on Configure pull down list and select Settings option as shown in FIGURE



Installing: step by step

- **Step 9:** In the Default Settings screen, on the left pane, click on Project Interpreter as shown in FIGURE



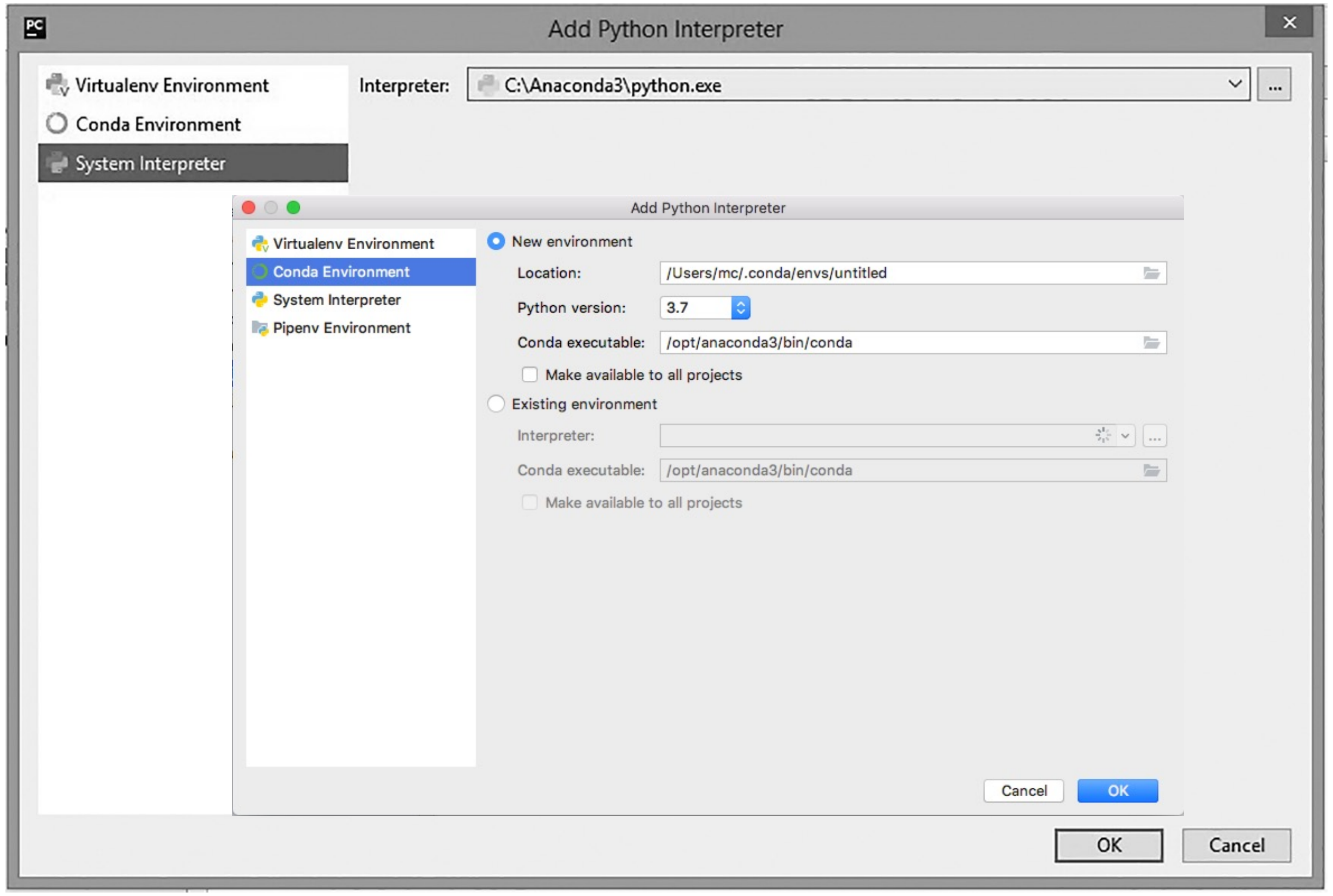
Project Interpreter   For default project

Project Interpreter: <No interpreter>

Nothing to show

Installing: step by step

- **Step 9:** On the right pane, in the Project Interpreter option, click on the button having toothed wheel icon and select Add. In the Add Python Interpreter screen, on the left pane, click on System Interpreter and select the Python interpreter path from the Interpreter pull down list as shown in FIGURE Click on OK button



Add Python Interpreter

- Virtualenv Environment
- Conda Environment
- System Interpreter

Interpreter: C:\Anaconda3\python.exe

Add Python Interpreter

- Virtualenv Environment
- Conda Environment
- System Interpreter
- Pipenv Environment

New environment

Location: /Users/mc/.conda/envs/untitled

Python version: 3.7

Conda executable: /opt/anaconda3/bin/conda

Make available to all projects

Existing environment

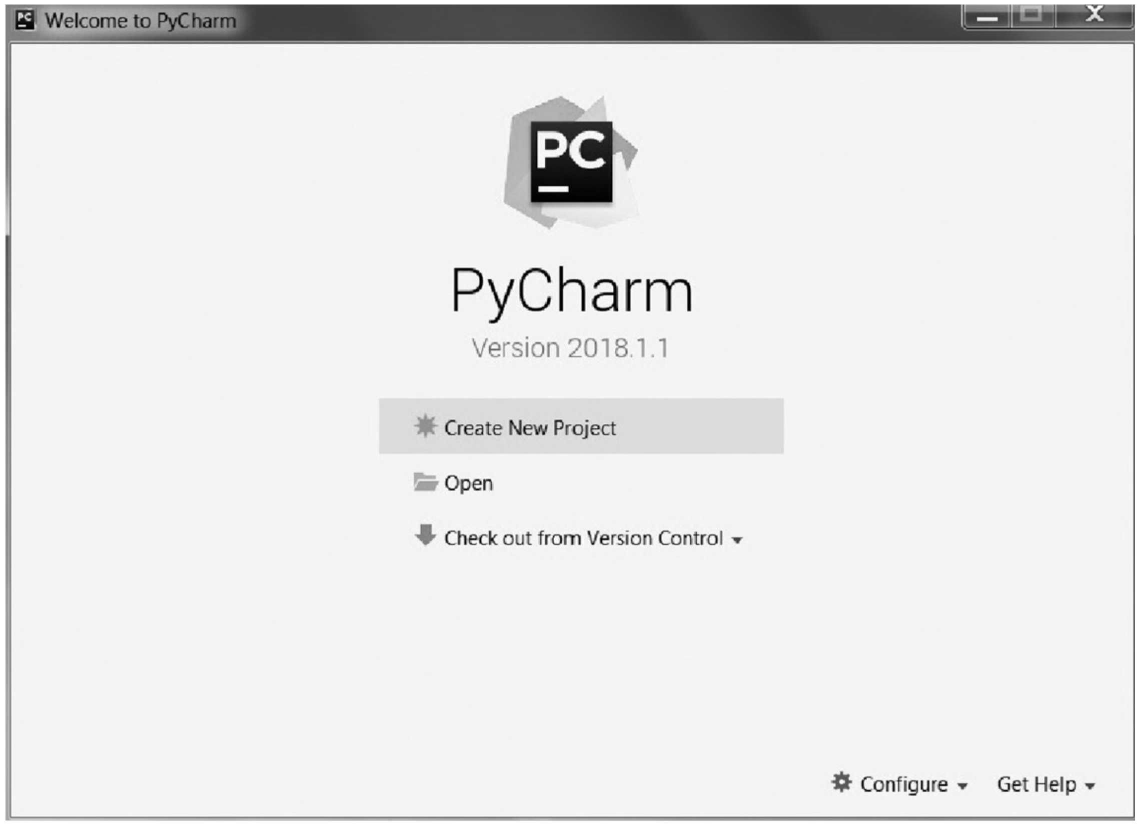
Interpreter:

Conda executable: /opt/anaconda3/bin/conda

Make available to all projects

Installing: step by step

- Step 10: It will take some time to list all the packages. Once done click on OK button.
- Step 11: You will be again presented with the Welcome screen. Now, to work with PyCharm IDE, click on Create New Project option.

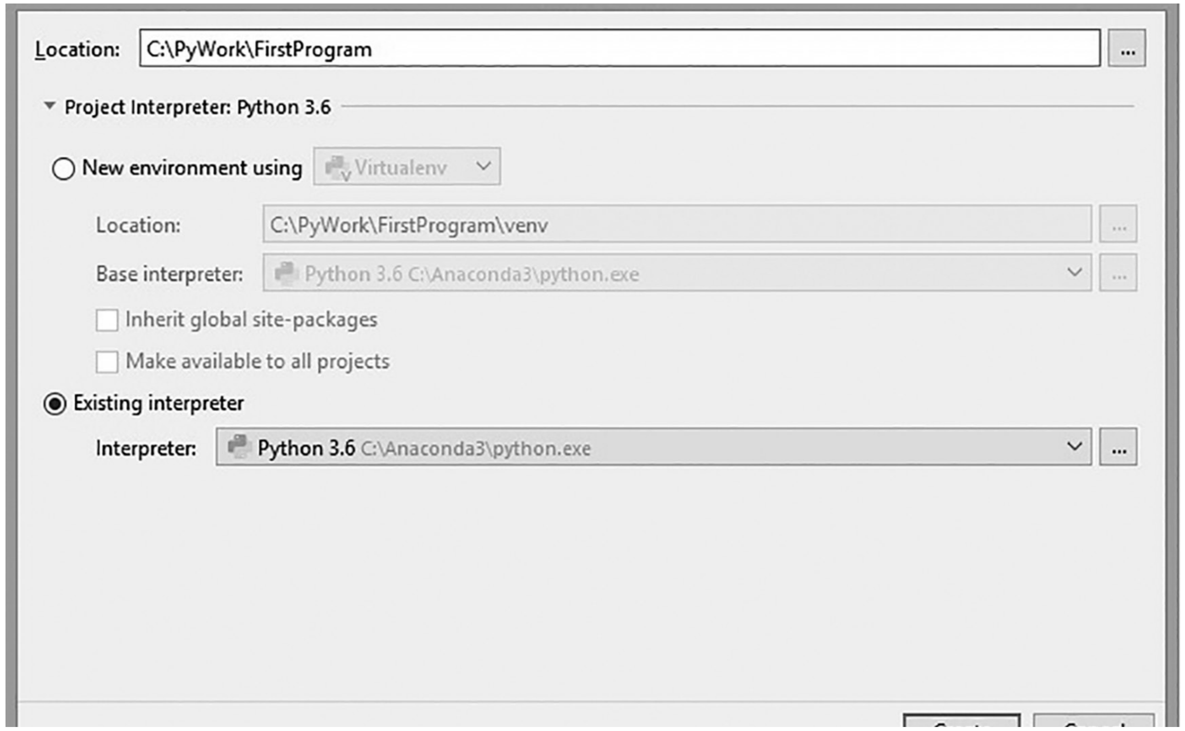


Creating and Running Your First Python Project

Before you start make sure that the following prerequisites are met:

- You are working with PyCharm Community Edition
- You have installed Python 3.7/8 supported Anaconda distribution.

Creating and running a Python project



Location: C:\PyWork\FirstProgram

Project Interpreter: Python 3.6

( ) New environment using Virtualenv

Location: C:\PyWork\FirstProgram\venv

Base interpreter: Python 3.6 C:\Anaconda3\python.exe

☐ Inherit global site-packages

☐ Make available to all projects

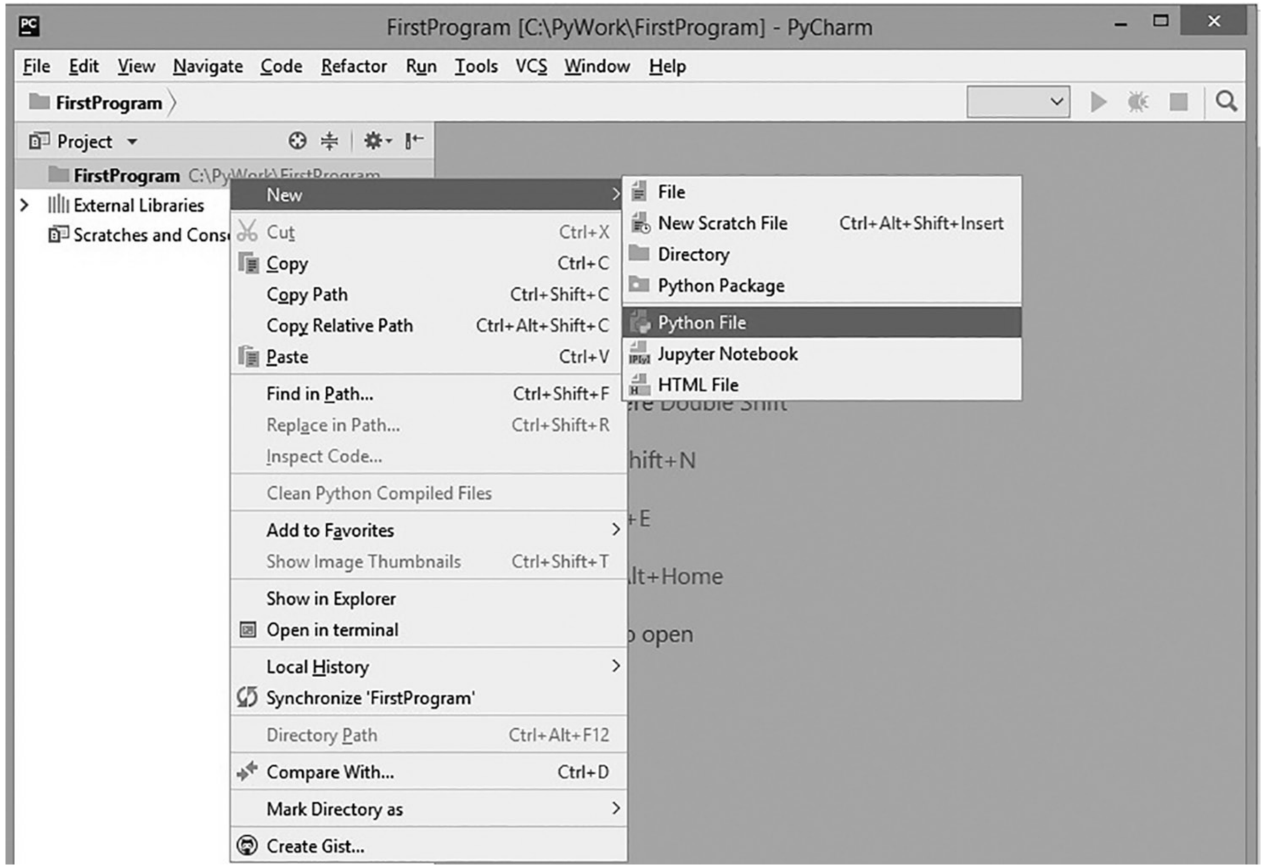
(●) Existing interpreter

Interpreter: Python 3.6 C:\Anaconda3\python.exe

Step 1: Click on the JetBrains Community Edition shortcut icon and you will be presented with a welcome screen and click on Create New Project. This screen is presented initially when you are creating a project for the first time. For the subsequent project creation, Go to File → New Project and go to Step 2. Create a folder called PyWork in C:\ drive.

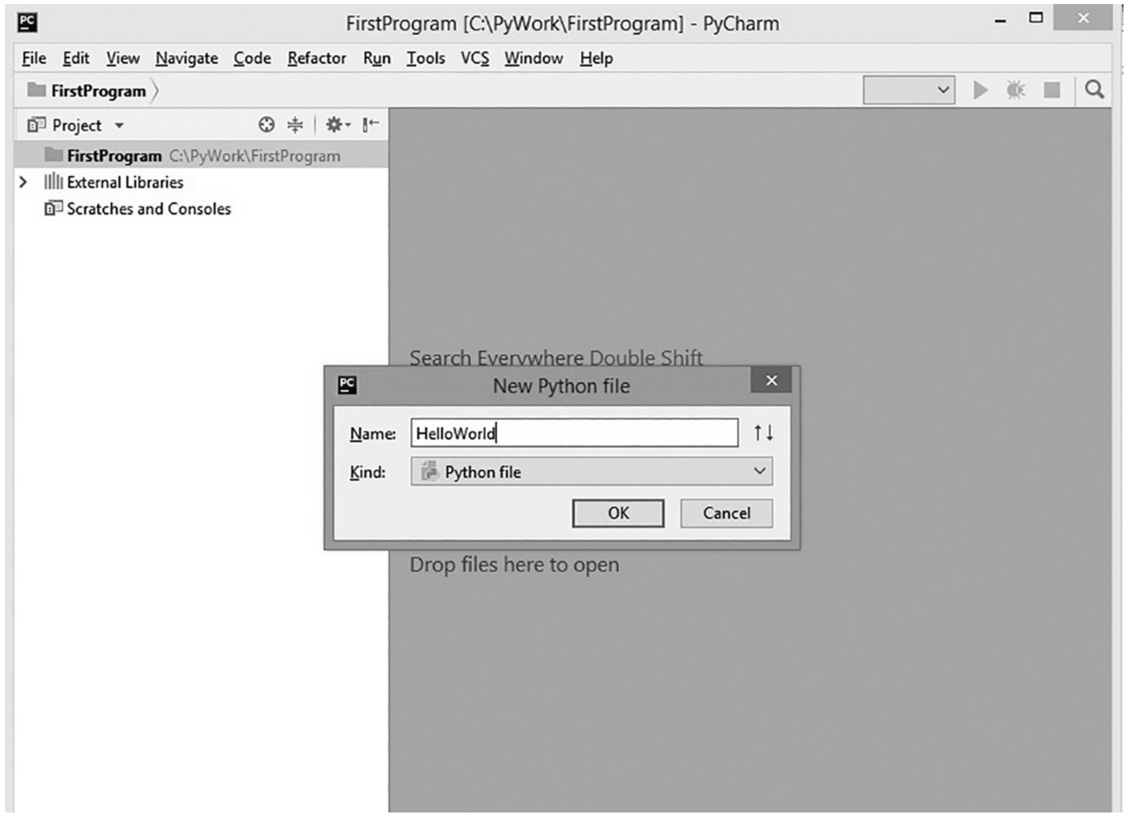
Step 2: For the Location option, browse to the PyWork folder which you had created in C:\ drive in Step 1. Now give a name to your Python Project. For the purpose of the demo, the project name is given as FirstProgram. Expand the project Interpreter and click on Existing interpreter radio button. Select Python Interpreter path from Interpreter pull down list if it is not already selected. Then, click on the Create button.

Creating and running a Python project


FirstProgram [C:\PyWork\FirstProgram] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
FirstProgram
Project
FirstProgram C:\PyWork\FirstProgram
External Libraries
Scratches and Consoles
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+Alt+Shift+C
Paste Ctrl+V
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Inspect Code...
Clean Python Compiled Files
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Show in Explorer
Open in terminal
Local History
Synchronize 'FirstProgram'
Directory Path Ctrl+Alt+F12
Compare With... Ctrl+D
Mark Directory as
Create Gist...
File
New Scratch File Ctrl+Alt+Shift+Insert
Directory
Python Package
Python File
Jupyter Notebook
HTML File

Step 3: On the left pane of the presented screen you can see the Project Name, which in our case is FirstProgram. Right-click on FirstProgram → Select New → Select Python File

Creating and running a Python project


FirstProgram [C:\PyWork\FirstProgram] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
FirstProgram
Project
FirstProgram C:\PyWork\FirstProgram
External Libraries
Scratches and Consoles
Search Everywhere Double Shift
Drop files here to open
OK
Cancel

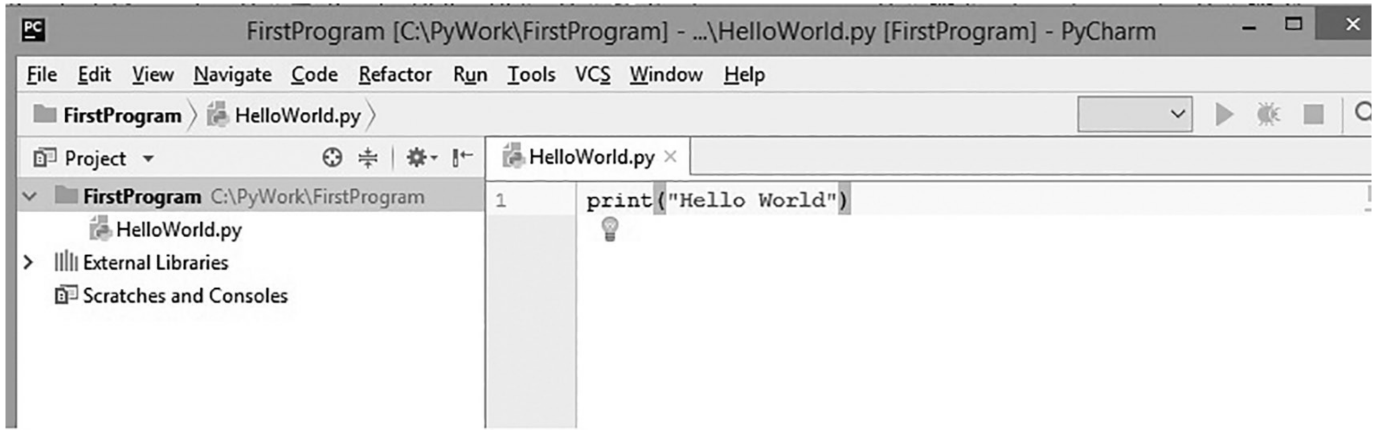
New Python file

Name: HelloWorld

Kind: Python file

Step 4: Give a name to the Python File. For the purpose of this demo, let's name it as HelloWorld as shown in FIGURE. No need to specify any extension. The PyCharm IDE itself will attach the .py extension to the file name which you have specified. Click on OK button.

Creating and running a Python project


FirstProgram [C:\PyWork\FirstProgram] - ...\HelloWorld.py [FirstProgram] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help
FirstProgram
HelloWorld.py
Project
FirstProgram C:\PyWork\FirstProgram
HelloWorld.py
External Libraries
Scratches and Consoles

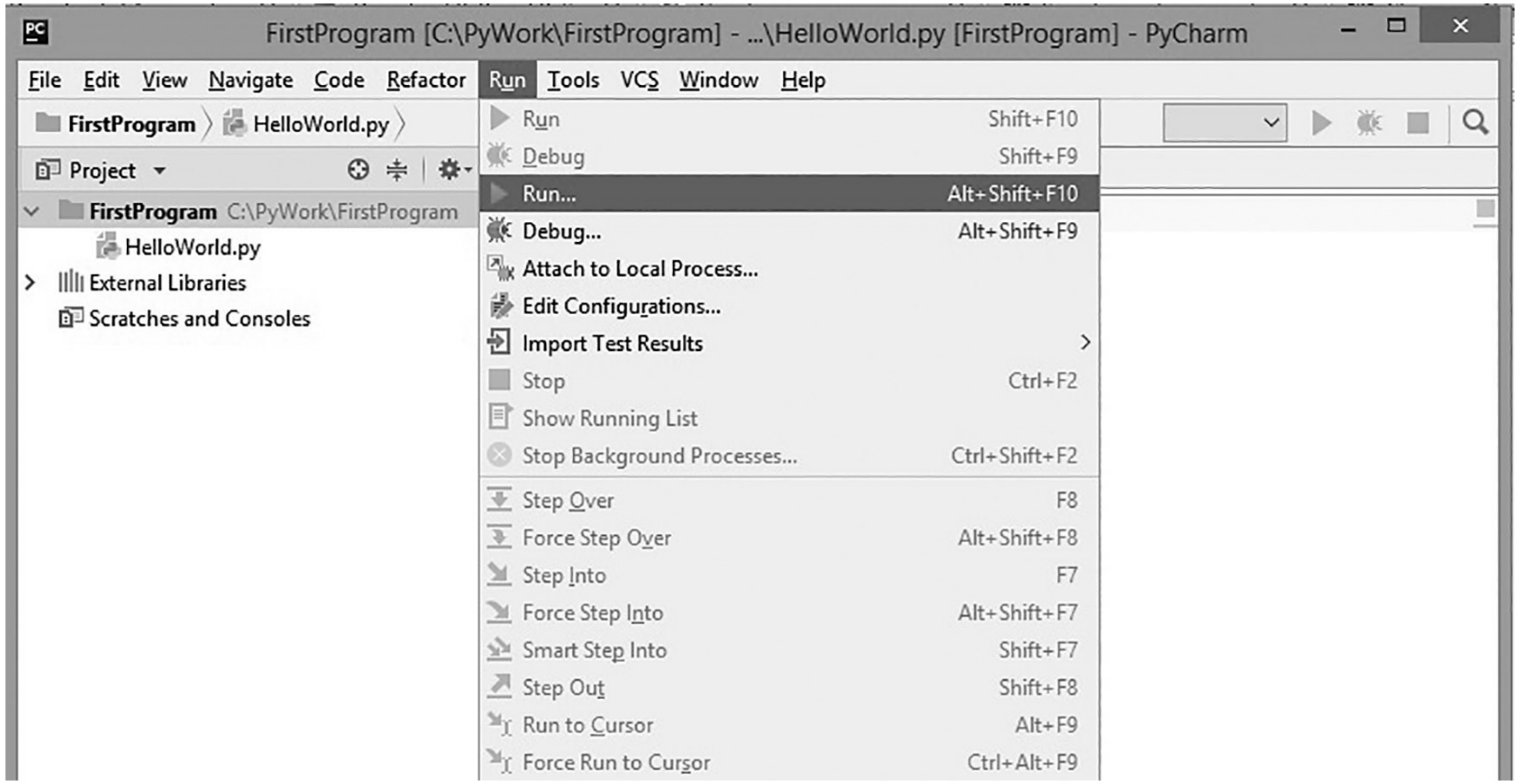
```
print("Hello World")
```

Step 5: On the left pane double click on the File name which you have created (in our case it is HelloWorld.py). This should open an empty file on the right side of the editor in which you type the following statement

print("Hello World")

Creating and running a Python project

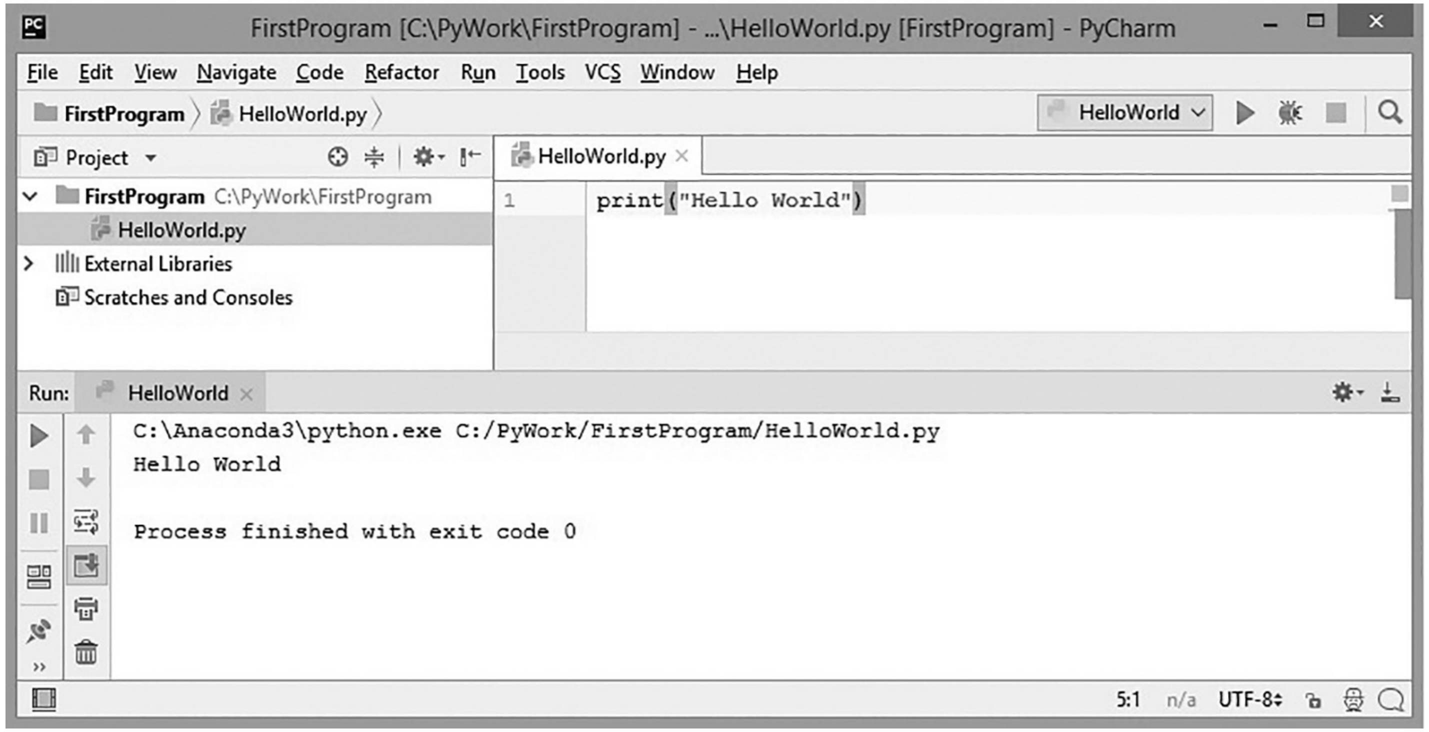
Step 6: To execute the above code, go to Run menu → and click on Run. Another way of executing the program is to right-click on the Python File Name (in our case it is HelloWorld.py) and select Run "HelloWorld".


FirstProgram [C:\PyWork\FirstProgram] - ...\HelloWorld.py [FirstProgram] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
FirstProgram HelloWorld.py
Project
FirstProgram C:\PyWork\FirstProgram
HelloWorld.py
External Libraries
Scratches and Consoles
Run Shift+F10
Debug Shift+F9
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Local Process...
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Stop Background Processes... Ctrl+Shift+F2
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Force Step Into Alt+Shift+F7
Smart Step Into Shift+F7
Step Out Shift+F8
Run to Cursor Alt+F9
Force Run to Cursor Ctrl+Alt+F9

Creating and running a Python project

Step 7: You can see the output in the output window of the PyCharm IDE.

Step 8: You can add multiple Python files to the project and execute them individually by following the steps 3, 4 and 5.


FirstProgram [C:\PyWork\FirstProgram] - ...\HelloWorld.py [FirstProgram] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
FirstProgram
HelloWorld.py
Project
FirstProgram C:\PyWork\FirstProgram
HelloWorld.py
External Libraries
Scratches and Consoles
Run:
HelloWorld
5:1 n/a UTF-8

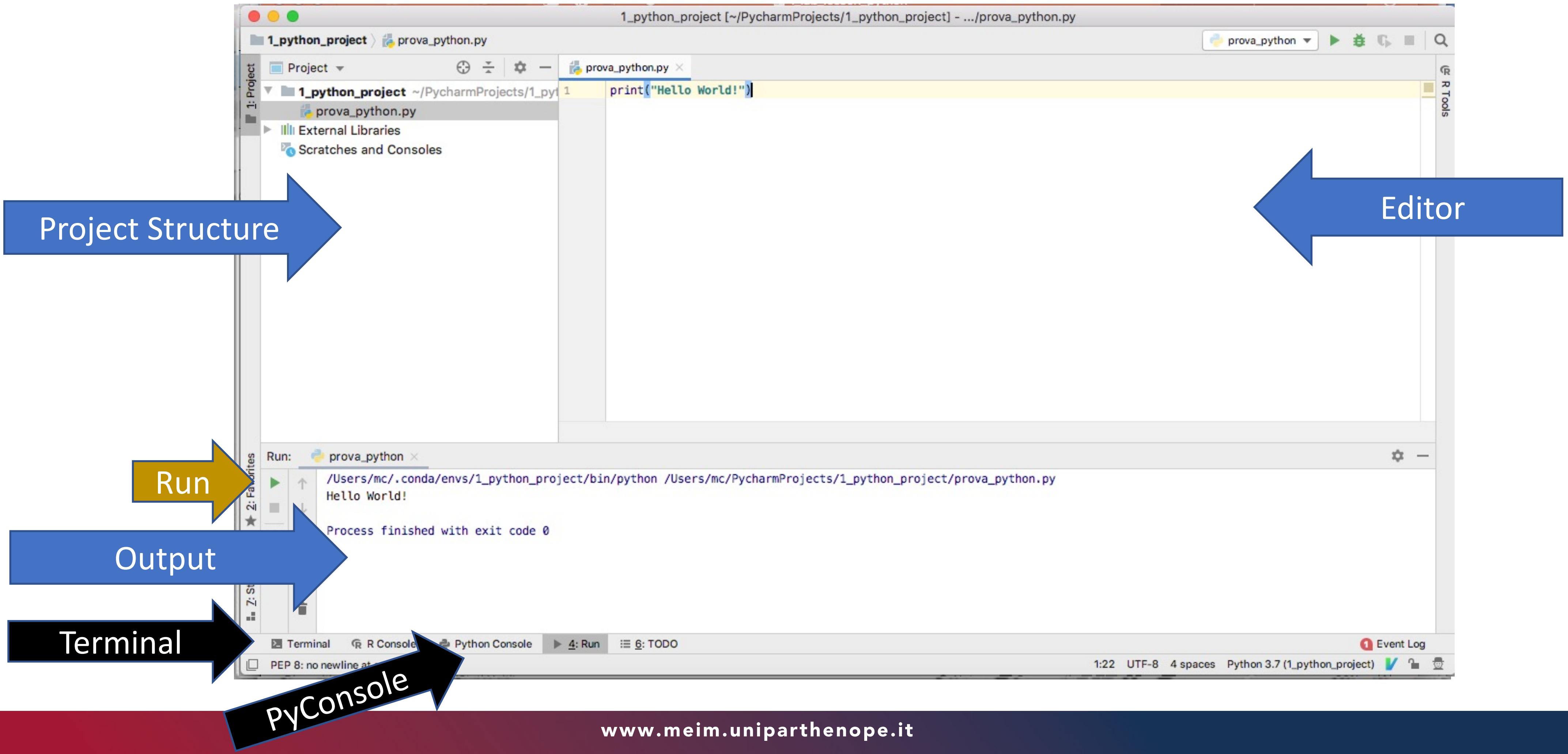
```
print("Hello World")
```

```
C:\Anaconda3\python.exe C:/PyWork/FirstProgram/HelloWorld.py
Hello World

Process finished with exit code 0
```

IDE components

- The source code editor can help programming by:
  - Listing line numbers of code
  - Color lines of code (comments, text…)
  - Auto-indent source code
- Output window
- Debugger
- Terminal

The Pycharm IDE



```
print("Hello World!")
```

```
/Users/mc/.conda/envs/1_python_project/bin/python /Users/mc/PycharmProjects/1_python_project/prova_python.py
Hello World!

Process finished with exit code 0
```

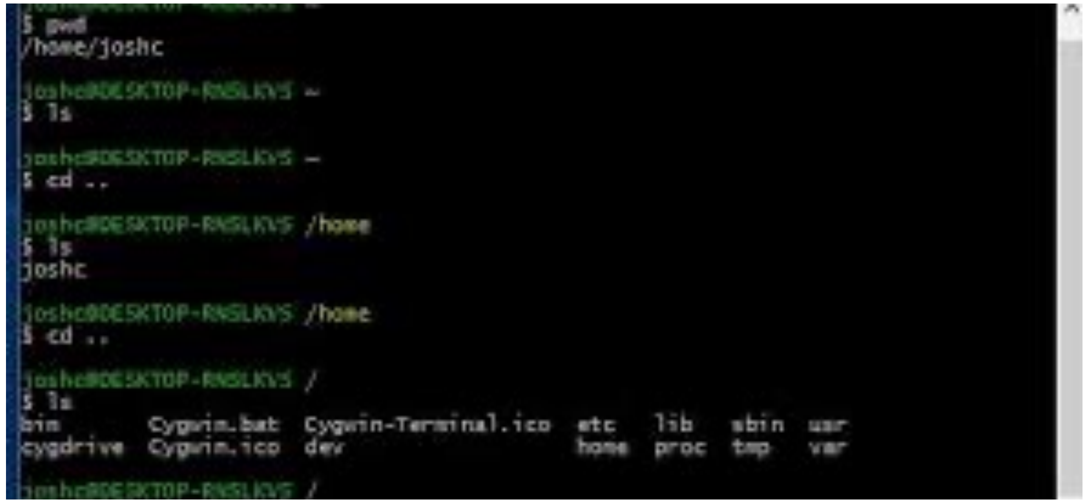
Terminal: bash shell?

- PyCharm Terminal includes a *textual shell*
- It supports different shells depending on the Operating System
- The shell supported by the PyCHarm IDE under Mac OS and Linux is the classical bash shell (the one we aim at using for our purposes)
- The shell supported by the PyCHarm IDE under Windows is the classical Command line interface cmd. → the following steps will allow you to substitute the *cmd* shell with the *bash* one.

How to include a Unix shell (bash) in Pycharm under Windows

1. Download Cygwin app from www.Cygwin.com . Both version at 32 and 64-bit exist. Make sure to select the right one
2. Follow the installation setup with default options.

   Cygwin will be installed in C:\



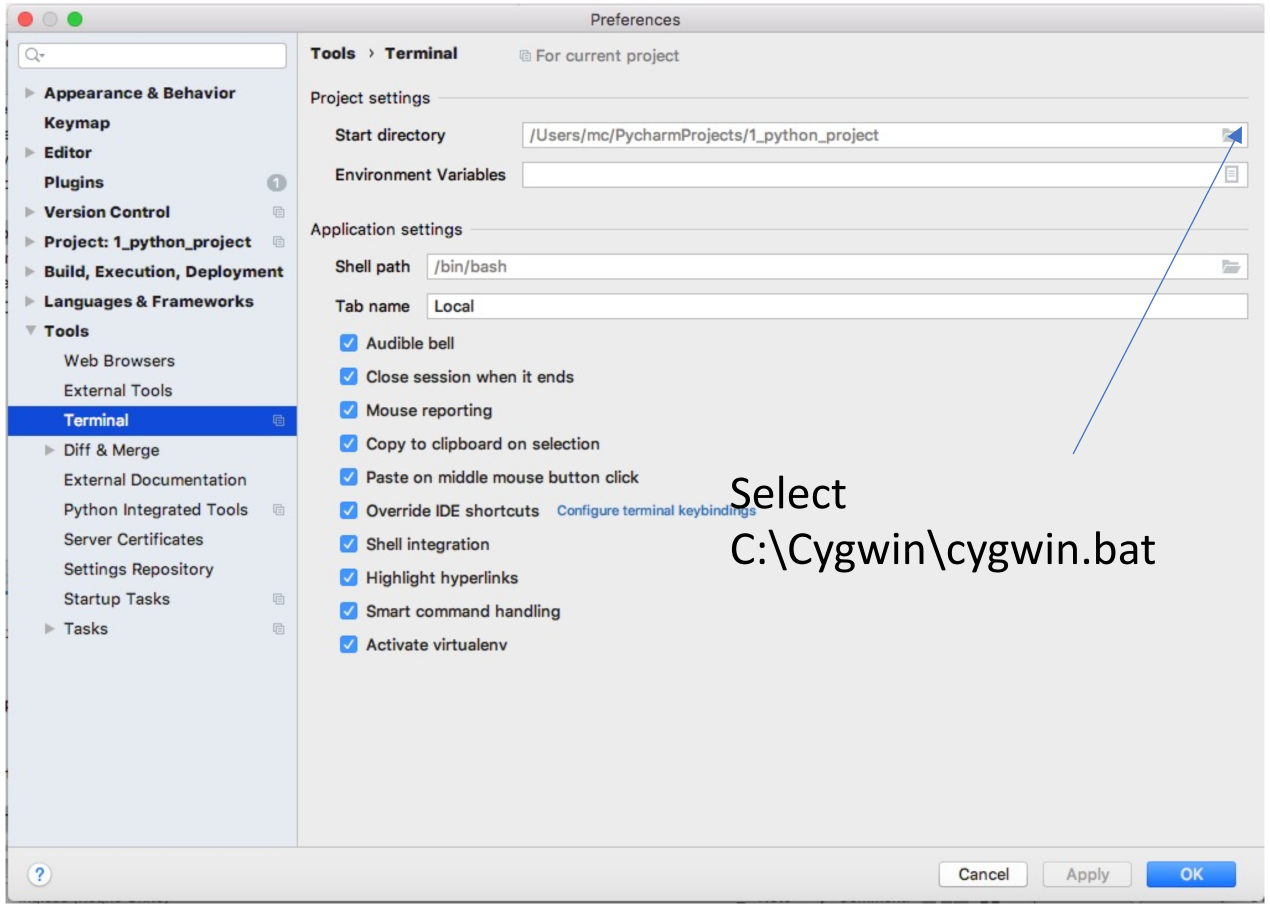
```
$ pwd
/home/joshc

joshc@DESKTOP-RNSLKVS ~
$ ls

joshc@DESKTOP-RNSLKVS ~
$ cd ..

joshc@DESKTOP-RNSLKVS /home
$ ls
joshc

joshc@DESKTOP-RNSLKVS /home
$ cd ..

joshc@DESKTOP-RNSLKVS /
$ ls
bin       Cygwin.bat  Cygwin-Terminal.ico  etc   lib   sbin  usr
cygdrive  Cygwin.ico  dev                  home  proc  tmp   var

joshc@DESKTOP-RNSLKVS /
```

Cygwin

Get that Linux feeling - on Windows

How to include a Unix shell (bash) in Pycharm under Windows

3. Open PyCharm IDE and Go to File/Settings/
6. Select Tools/Terminal
7. Click on the Browse button to select the "Shell Path" and select C:\Cygwin\cygwin.bat
8. Apply the changes and click on "ok"



Tools › Terminal   For current project

Project settings
Start directory: /Users/mc/PycharmProjects/1_python_project
Environment Variables:

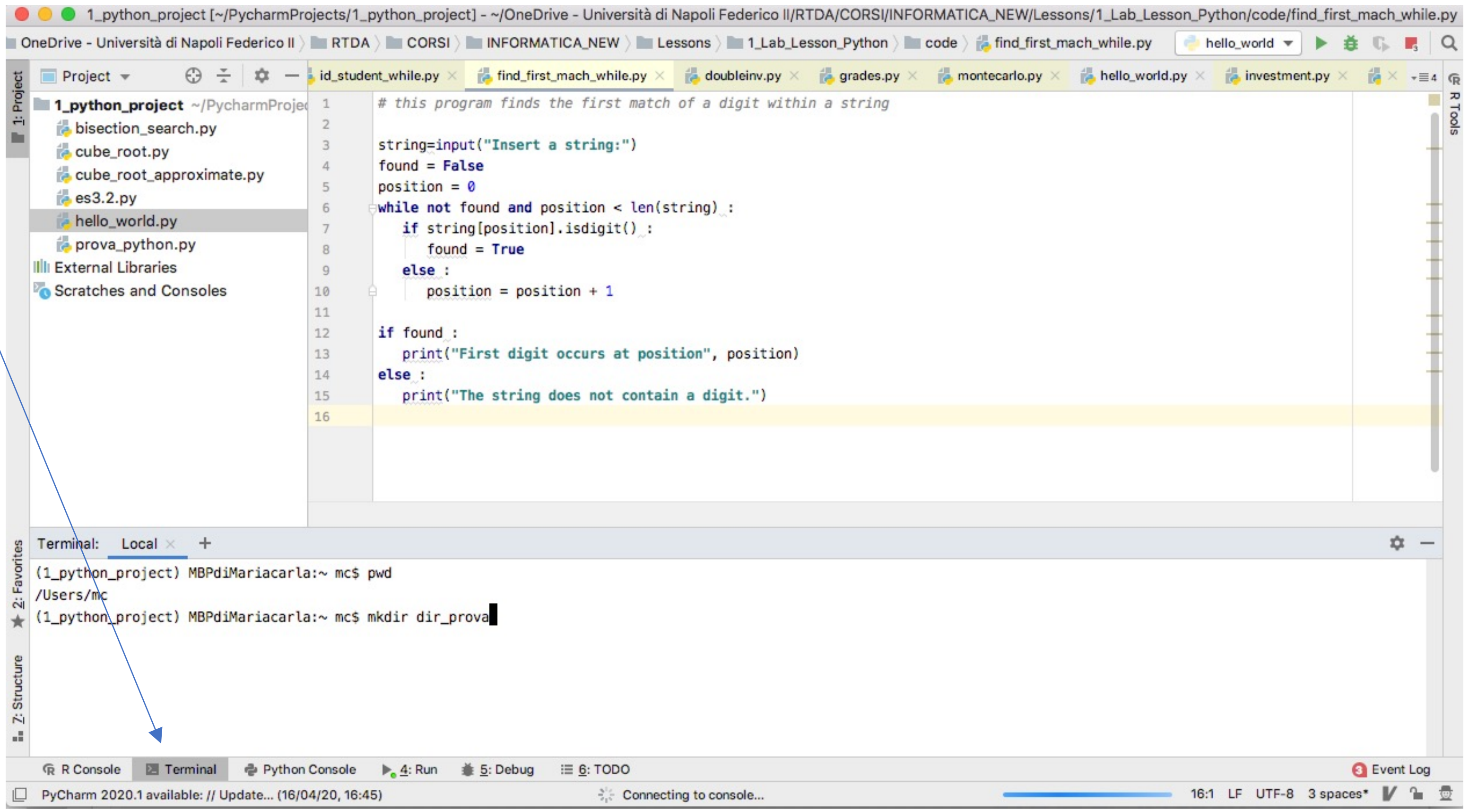
Application settings
Shell path: /bin/bash
Tab name: Local
- [✓] Audible bell
- [✓] Close session when it ends
- [✓] Mouse reporting
- [✓] Copy to clipboard on selection
- [✓] Paste on middle mouse button click
- [✓] Override IDE shortcuts   Configure terminal keybindings
- [✓] Shell integration
- [✓] Highlight hyperlinks
- [✓] Smart command handling
- [✓] Activate virtualenv

Select C:\Cygwin\cygwin.bat

How to include a Unix shell (bash) in Pycharm under Windows

9. Now go on the PyCharm IDE Terminal

10. digit some unix command to check the shell is changed (es. pwd, ls, etc.) . If the commands are executed the unix shell has been correctly integrated in PyCharm

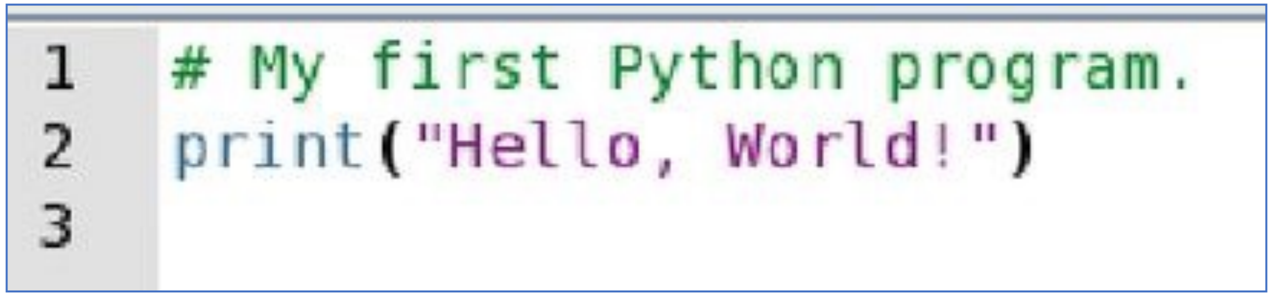


```python
# this program finds the first match of a digit within a string

string=input("Insert a string:")
found = False
position = 0
while not found and position < len(string) :
   if string[position].isdigit() :
      found = True
   else :
      position = position + 1

if found :
   print("First digit occurs at position", position)
else :
   print("The string does not contain a digit.")
```

```
(1_python_project) MBPdiMariacarla:~ mc$ pwd
/Users/mc
(1_python_project) MBPdiMariacarla:~ mc$ mkdir dir_prova
```

Your first program

- Traditional ‘Hello World’ program in Python



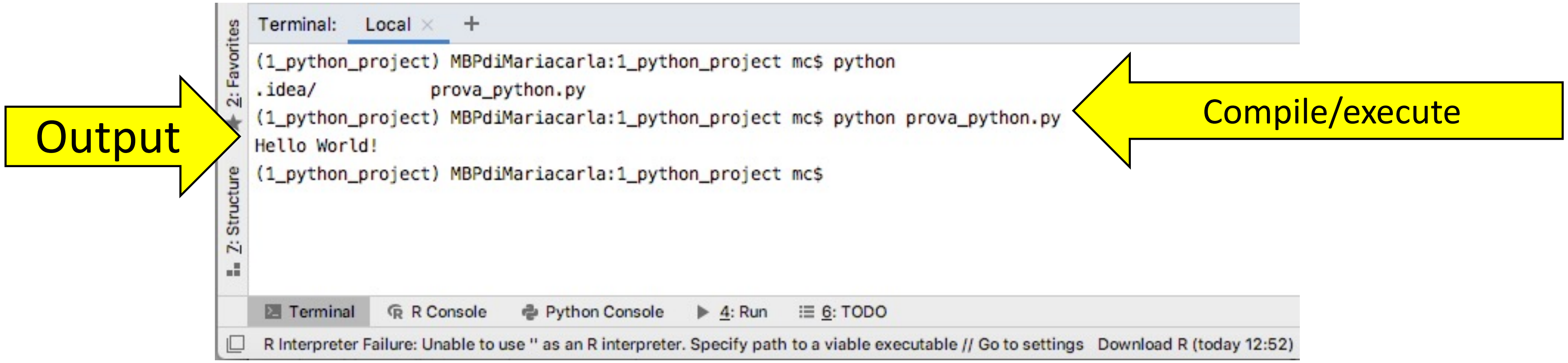
```python
# My first Python program.
print("Hello, World!")
```

- We will examine this program in the next section
  - Typing the program into your IDE would be good practice!
  - Be careful of spelling e.g., ‘print’ vs. ‘primt’

PyTHon iS CaSe SeNsItiVe.

Text editor programming

- Remember that you can also use a simple text editor to write your source code
- Once saved as Hello.py, you can use a console window to:
  - Compile the program
  - Run the program



```
(1_python_project) MBPdiMariacarla:1_python_project mc$ python
.idea/          prova_python.py
(1_python_project) MBPdiMariacarla:1_python_project mc$ python prova_python.py
Hello World!
(1_python_project) MBPdiMariacarla:1_python_project mc$
```

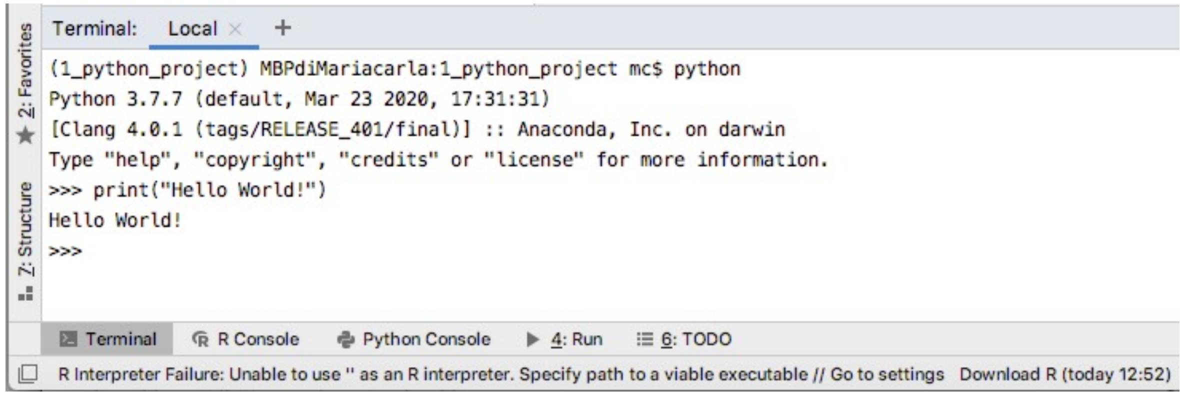
R Interpreter Failure: Unable to use '' as an R interpreter. Specify path to a viable executable // Go to settings   Download R (today 12:52)

Organize your work

- Your ‘source code’ is stored in .py files
- Create a folder for this course
- Create one folder/project per lesson for collecting related programs inside in
  - A project can consist of several .py files
- Be sure you know where your IDE stores your files
  - You need to be able to find you files

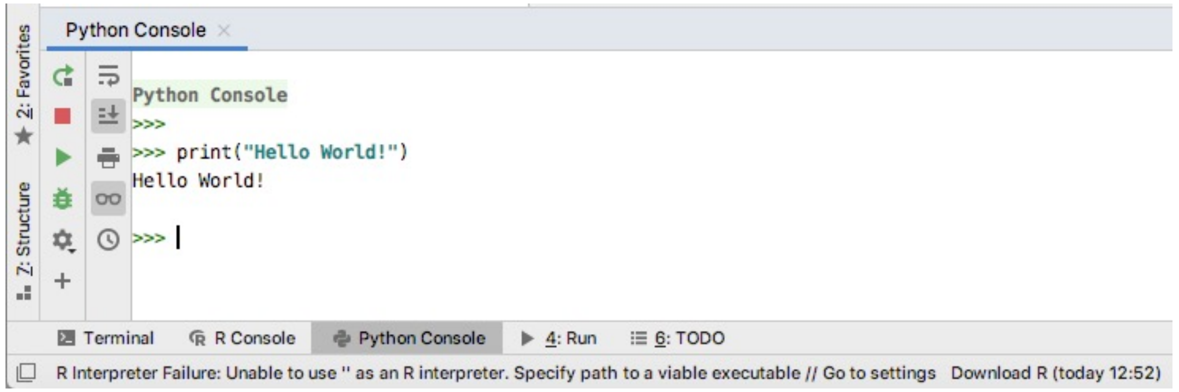
Python interactive mode

- Like other languages you can write/save a complete Python program in a file and let the interpreter execute the instructions all at once.
- Alternatively you can run instructions one at a time using interactive mode.
    - It allows quick 'test programs' to be written.
    - Interactive mode allows you to write python statements directly in the **console window** or in the **terminal** by typing "python" to start the interpreter.



```
(1_python_project) MBPdiMariacarla:1_python_project mc$ python
Python 3.7.7 (default, Mar 23 2020, 17:31:31)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>>
```

R Interpreter Failure: Unable to use '' as an R interpreter. Specify path to a viable executable // Go to settings Download R (today 12:52)



```
Python Console
>>>
>>> print("Hello World!")
Hello World!

>>> |
```

R Interpreter Failure: Unable to use '' as an R interpreter. Specify path to a viable executable // Go to settings Download R (today 12:52)

Let’s Get Started!

- Open the PyCharm IDE on you computer
- We are going to start simple, and as we learn more about Python, we’ll use additional features in PyCharm

The PyCharm Tool Bar

In the spirit of keeping things simple we will start with the four icons in the top left corner

- Create a new file
- Open a file from disk
- Save the active file
- Save all unsaved items

Writing Our First Program

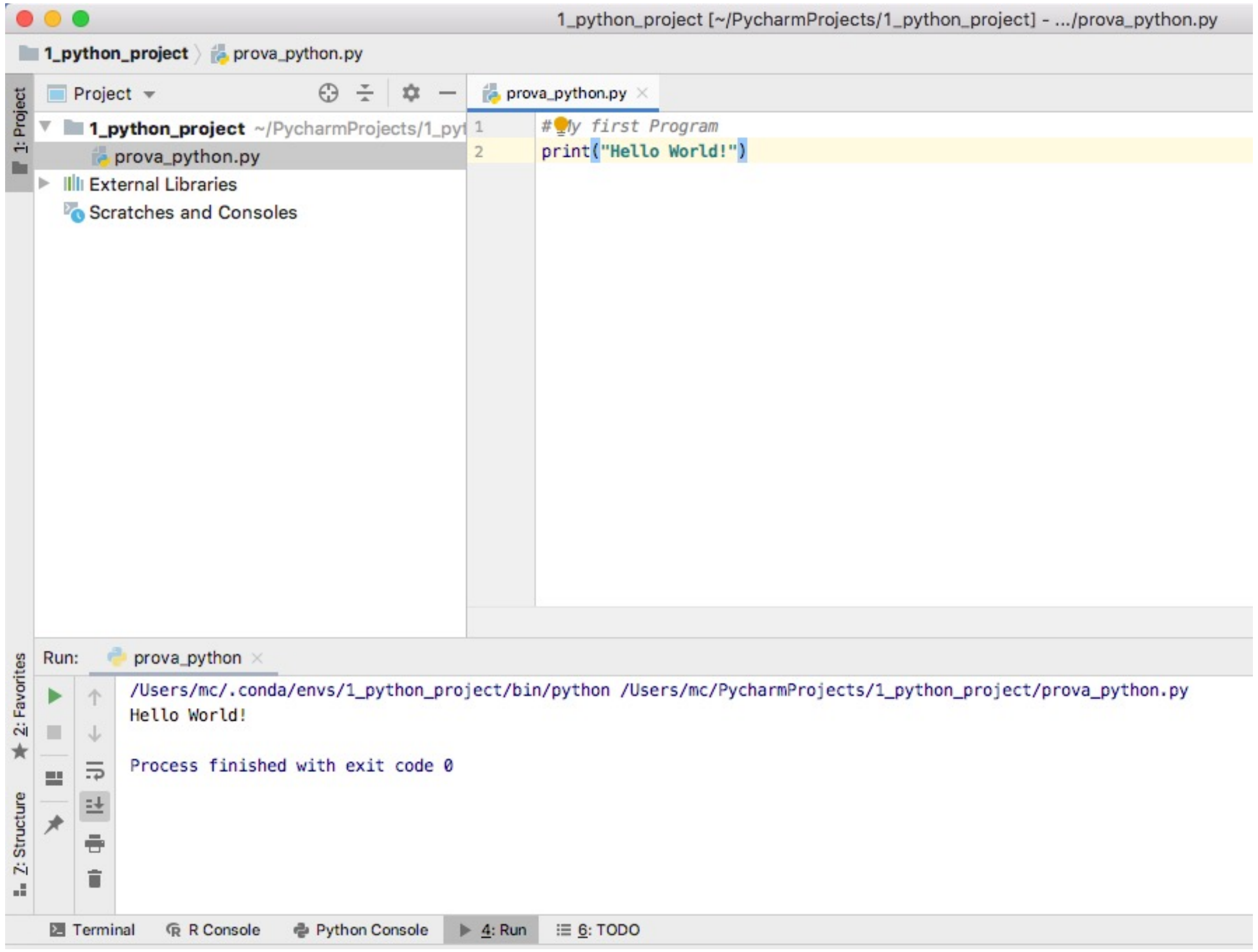
- Click on the icon to create a new file
- The File Editor window will open
  - Type the following into the Editor:

```
# My first Python program
  print(“Hello World!”)
```

  - Save your file as “hello.py”
  - This is “Step 2 Write a simple program”
  - Remember – Python is **case sensitive**
    - You have to enter the upper and lower case letters exactly as this appear above

Running your Program

Click the Green Arrow in the Tool Bar



```
#My first Program
print("Hello World!")
```

```
/Users/mc/.conda/envs/1_python_project/bin/python /Users/mc/PycharmProjects/1_python_project/prova_python.py
Hello World!

Process finished with exit code 0
```

Analyzing Your First Program

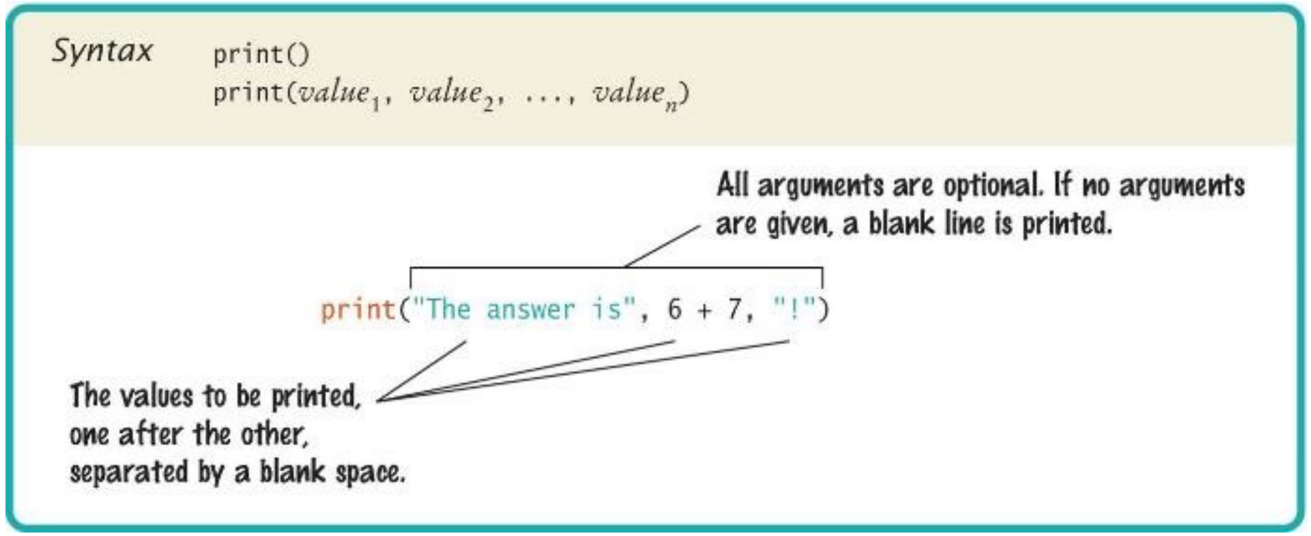
- A Python program contains one or more lines of instructions (statements) that will be translated and executed by the interpreter

```
# My first Python program
Print("Hello World!")
```

- The first line is a **comment** (a statement that provides descriptive information about the program to programmers).
- The second line contains a **statement** that prints a line of text onscreen "Hello, World!"

Basic Python Syntax: *Print*

- Using the Python 'print()' function.
  - A function is a collection of programming instructions that carry out a particular task (in this case to print a value onscreen).
  - It's code that somebody else wrote for you!



Syntax

```
print()
print(value1, value2, ..., valuen)
```

```
print("The answer is", 6 + 7, "!")
```

All arguments are optional. If no arguments are given, a blank line is printed.
The values to be printed, one after the other, separated by a blank space.

Syntax for Python Functions

- To use, or call, a function in Python you need to specify:
  - The name of the function that you want to use (in the previous example the name was print)
  - Any values (arguments) needed by the function to carry out its task (in this case, "Hello World!").
  - Arguments are enclosed in *parentheses* and multiple arguments are *separated with commas*.
  - A *sequence of characters* enclosed in quotations marks are called a string

More Examples of the print Function

- Printing numerical values
  ```
  print(3 + 4)
  ```
  - Evaluates the expression 3 + 4 and displays 7
- Passing multiple values to the function
  ```
  print("the answer is", 6 * 7)
  ```
  - Displays The answer is 42
  - Each value passed to the function is displayed, one after another, with a blank space after each value
- By default the print function starts a new line after its arguments are printed
  ```
  print("Hello")
  print("World!")
  ```
  - Prints two lines of text
  - Hello
  - World!

Our Second Program (Page 12, printtest.py)

```
##
#   Sample Program that demonstrates the print function
#
#   Prints 7

print(3 + 4)

#   Print Hello World! on two lines
print(“Hello”)
print(“World!”)

#   Print multiple values with a single print function call
print(“My favorite number are”, 3 + 4, “and” 3 + 10)

#   Print Hello World! on two lines
print(“Goodbye”)
print()
print(“Hope to see you again”)
```

Errors

- There are two Categories of Errors:
  - Compile-time Errors
    - aka Syntax Errors
      - Spelling, capitalization, punctuation
      - Ordering of statements, matching of parenthesis, quotes…
    - No executable program is created by the compiler
    - Correct first error listed, then compile again.
      - Repeat until all errors are fixed
  - Run-time Errors
    - aka Logic Errors
    - The program runs, but produces unintended results
    - The program may ‘crash’

Syntax Errors

- Syntax error are caught by the compiler
- What happens if you

• Miss-capitalize a word:	`Print("Hello World!")`
• Leave out quotes	`print(Hello World!)`
• Mismatch quotes	`print("Hello World!')`
• Don't match brackets	`print('Hello'`

- Type each example above in the Wing **Python Shell** window
  - What error messages are generated?

Logic Errors

- What happens if you
  - Divide by zero `print(1/0)`
  - Misspell output `print("Hello, Word!")`
- Programs will compile and run
  - The output may not be as expected
- Type each example above in the Wing **Python Shell** window
  - What error messages are generated?

Read from console: Input()

- You can read data from the console with the input() function and associate it to a variable:

```
name = input("Please enter your name")
```

- If you want to read numerical value you have to convert the string with the converting functions int() or float() :
- age = int(input("Please enter age: "))
- kg = float(input("Please enter your weight: "))

- The above is equivalent to doing it two steps (getting the input and then converting it to a number):

```
aString = input("Please enter age: ")      # String input
age = int(aString)                         # Converted int
```

MASTER MEIM 2021-2022

Thank you for your attention