

## Exercise 5

# Introduction to Python

---

Part 1: Python Overview . . . . .	67
Part 2: Installing Python . . . . .	67
Part 3: Installing Pycharm IDE . . . . .	68
Part 4: Writing your first Python program . . . . .	68
Part 5: Python syntax . . . . .	69
Part 5.1: Variables & Constants . . . . .	70
Part 5.2: Input & Output . . . . .	70
Part 5.3: Selection Structures . . . . .	70
Part 5.4: Loop Structures . . . . .	71
Part 5.5: Functions . . . . .	71
Part 5.6: Classes & Objects . . . . .	71
Part 6: Practice questions . . . . .	72
References . . . . .	72

---

## Estimated time

03:00 Hours

## Overview

It is assumed that you have basic programming skills and, you are familiar with high level programming languages (i.e. C++, Java). The exercise leverages on this familiarity in order to introduce you to Python programming language.

In this exercise, you write simple Python programs.

## Objectives

After completing this exercise, you should be:

- Install Python software on a PC.
- Install Pycharm IDE on a PC.
- Write simple Python programs.

## Prerequisites

Basic understanding of high-level programming OR completed Exercise 1: *Introduction to programming* or Exercise 2: *Object-oriented programming*.

## Requirements

This exercise requires a PC that has Internet access and, has a Web browser installed (i.e. Chrome).

## Exercise instructions

In this exercise, you complete the following tasks:

- Write and compile code excerpts as answers to exercise questions.
- Add a comment with the following information: **Name**, **Student Number**, and **Date**.
- Paste code excerpts in a Word document and upload it the link provided in the e-learning.

## Part 1: Python Overview

Python language was conceived and designed by Guido van Rossum in 1980-90s and first released in 1991. The current official developer is Python Software Foundation.

Python is an interpreted high-level programming language whose design philosophy emphasizes *code readability*. It supports multiple programming paradigms: structured (or procedural), object-oriented and functional programming.

Python language has many uses, some popular ones include:

- web development (server-side) and software development;
- interactive computing (**Jupyter**);
- scientific (mathematics) computing (**NumPy**, **SciPy**, **Matplotlib**);
- artificial intelligence (**TensorFlow**, **Scikit-learn**);
- data manipulation and analysis (**Pandas**).

Python software runs on different platforms (Windows, Mac, Linux, Raspberry Pi, etc) and it has a simple syntax similar to the English language.

## Part 2: Installing Python

Before you begin writing Python code, you need Python software in your system. There are numerous ways for installing Python on any operating system. Follow this link <https://wiki.python.org/moin/BeginnersGuide/Download> to learn and choose your preferred installation method. Here are a few installation options:

- **Anaconda distribution\*\***: <https://www.anaconda.com/products/individual>.
- Python standalone: <https://www.python.org/downloads/>.
- Homebrew Package Manager: <https://brew.sh> ('brew install python').

## Part 3: Installing Pycharm IDE

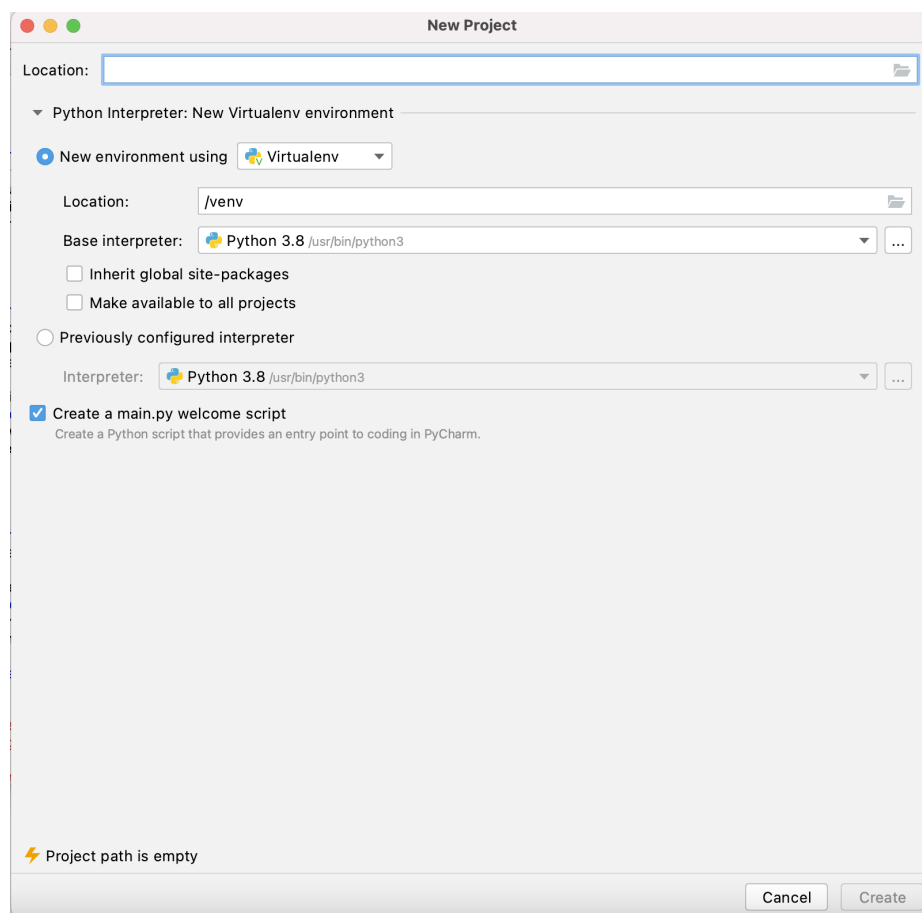
Again before getting started, you may need an IDE or a text editor that has been tailored to make editing, interpreting and executing Python code easy. This course recommends Pycharm IDE for these tasks since it is specifically designed for Python language.

- Installing Pycharm IDE ‘Community Edition’ (CE) via Download, follow this link <https://www.jetbrains.com/pycharm/download/>.

## Part 4: Writing your first Python program

In this exercise, you write and execute your first Python program. Follow the steps that follow:

1. Launch Pycharm IDE, create a ‘New Project’ and edit the configuration details. Select the ‘Project path’ and select your preferred Python interpreter (already installed in Part 2).

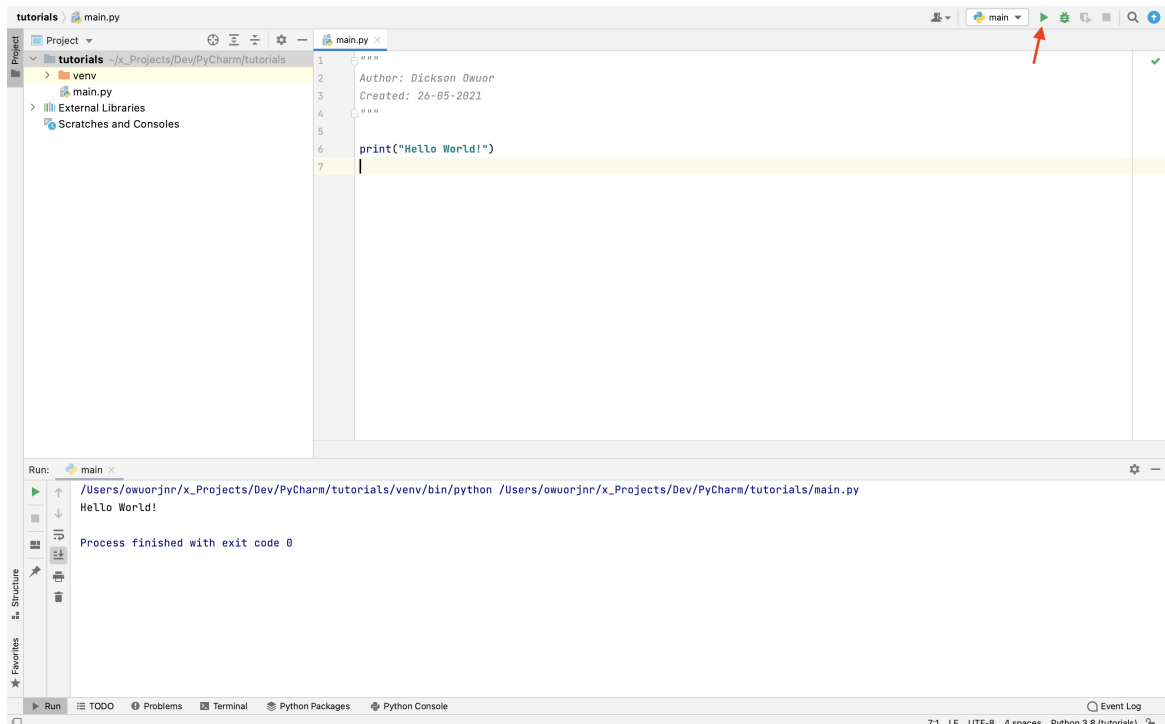


2. Create a new Python file from the 'File' menu (you may name it 'main.py').
3. Write the code shown in the figure below.



```
1 """
2 Author: Dickson Owuor
3 Created: 26-05-2021
4 """
5
6 print("Hello World!")
7
```

4. To interpret and execute your code, click on the 'run' button shown by the red arrow in the figure below. The results should be displayed in the Output window.



## Part 5: Python syntax

Python strives for a simpler, less-cluttered syntax and grammar. Below are the guidelines to adhere to when writing Python code:

- Designed for readability, and has some similarities to the English language with influence from mathematics.

- Uses new lines to complete a command (NOT semicolons or parentheses).
- Relies on indentation, using full-colon, using whitespace, to define scope (i.e. scope of loops, functions, classes).

### Part 5.1: Variables & Constants

```
"""
Author: Dickson Owuor
Created: 26-05-2021
"""

x = 5    # x is an Integer variable
y = "hello"  # y is a String variable
z = 3.0   # z is a Float variable

k = x + 12
print(k)
```

### Part 5.2: Input & Output

```
"""
Author: Dickson Owuor
Created: 26-05-2021
"""

# Input assignment
name = input("What is your name?\n")
print("Hi, %s" %name)
```

### Part 5.3: Selection Structures

```
"""
Author: Dickson Owuor
Created: 26-05-2021
"""

a = 33
b = 33
if b > a:
    print(str(b) + " is greater than " + str(a))
elif b < a:
    print(str(b) + " is lesser than " + str(a))
else:
    print(str(b) + " and " + str(a) + " are equal")
```

## Part 5.4: Loop Structures

```
"""
Author: Dickson Owuor
Created: 26-05-2021
"""
```

```
# while loop
while i < 6:
    print(i)
    i += 1
```

```
# for loop
for x in range(6):
    print(x)
```

## Part 5.5: Functions

```
"""
Author: Dickson Owuor
Created: 26-05-2021
"""
```

```
def my_function(name):
    print("Welcome to Python, " + name)
```

```
my_function("Dickson")
```

## Part 5.6: Classes & Objects

```
"""
Author: Dickson Owuor
Created: 26-05-2021
"""
```

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def print_details(self):
        print(str(self.name) + " is " + str(self.age) + " yrs old")
```

```
p1 = Person("Dickson", 58)
print(p1.name)
p1.print_details()
```

## Part 6: Practice questions

Attempt all the questions that follow:

1. Write a program that displays all the *odd numbers* between 0 and 100. The program should also display the sum of the odd numbers.
2. Write a program that will calculate and display the *Factorial* of any number the user enters.
3. Write a program that will display the *Fibonacci sequence* of any number a user enters.

## References

- [ 1 ] <https://www.python.org/>
- [ 2 ] [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
- [ 3 ] <https://www.w3schools.com/python/>

## End of exercise



## Exercise 6

# Jupyter interactive computing

---

Part 1: <a href="#">Jupyter</a> overview . . . . .	75
Part 2: <a href="#">Jupyter Notebook</a> web application . . . . .	75
Part 3: <a href="#">JupyterLab</a> web IDE . . . . .	76
Part 4: Running interactive code . . . . .	76
Part 5: Practice questions . . . . .	77
References . . . . .	77

---

## Estimated time

01:00 Hour

## Overview

In this exercise, you install **Jupyter** software library and run interactive code in **Python** language.

## Objectives

After completing this exercise, you should be:

- Install **Jupyter** software library.
- Run interactive code excerpts in **Python**.

## Prerequisites

Completed Exercise [5](#): *Introduction to Python*.

## Requirements

This exercise requires a PC that has Internet access and, has a Web browser installed (i.e. **Chrome**).

## Exercise instructions

In this exercise, you complete the following tasks:

- Write and compile code excerpts as answers to exercise questions.
- Add a comment with the following information: **Name**, **Student Number**, and **Date**.
- Save your *notebook* as a PDF and upload it the link provided in the e-learning.

## Part 1: Jupyter overview

Project **Jupyter** is a project and community whose goal is to “develop open-source software, open-standards, and services for interactive computing across dozens of programming languages”. Project **Jupyter** has developed and supported the interactive computing products **Jupyter Notebook**, **JupyterHub**, and **JupyterLab**.

**Jupyter** applications (i.e. **Jupyter Notebook**, **JupyterLab** etc) are an open document format based on JSON. They contain a complete record of the user’s sessions and include code, narrative text, equations and rich output. Each application communicates with computational Kernels using the Interactive Computing Protocol, an open network protocol based on JSON data over ZMQ and WebSockets.

Kernels are processes that run interactive code in a particular programming language and return output to the user. Kernels also respond to tab completion and introspection requests.

## Part 2: Jupyter Notebook web application

The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.


### Installing and launching Jupyter Notebook

**Jupyter Notebook** can be installed using a Package installer such as **conda** or **pip**. In this exercise, we use the **pip** package installer (already installed together with **Python** in Exercise 5).

1. **Jupyter Notebook** can be installed via **pip** from PyPI:

A dark blue rectangular button with the text "pip install notebook" in white, followed by a white icon of a document with a plus sign.

2. Once installed, launch **Jupyter Notebook** from the Terminal/Command Prompt with:

A light gray rectangular box with the text "jupyter notebook" in a monospaced font.

## Part 3: JupyterLab web IDE

JupyterLab is a web-based interactive development environment for **Jupyter** notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

### Installing and launching JupyterLab

1. JupyterLab can be installed via pip from PyPI:

```
pip install jupyterlab
```

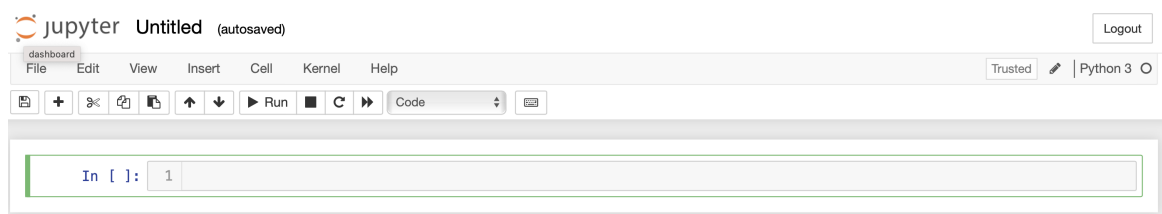
2. Once installed, launch JupyterLab the Terminal/Command Prompt with:

```
jupyter-lab
```

## Part 4: Running interactive code

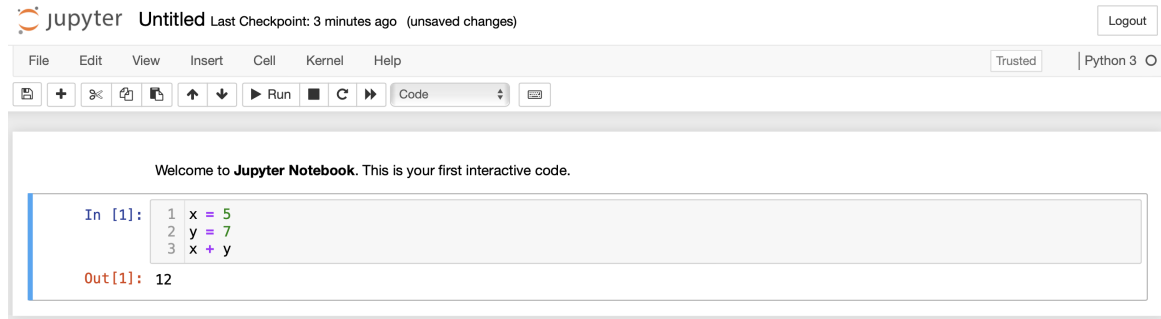
In this exercise, you write and run your first interactive code (using either **Jupyter Notebook** or **JupyterLab**). Follow the steps that follow:

1. Launch **Jupyter Notebook/JupyterLab**;
2. Create a new **Python** notebook. This opens a notebook on a new tab (if you are using **Jupyter Notebook**) as shown in the image below.
3. Save your notebook.



4. The Menu bar allows you to manage files, edit cells, restart or shutdown Kernel.
5. The Toolbar allows you to add, copy or cut cells; run code; switch to Markdown or Code.

6. Write and run the Markdown and Code as shown in the figure below.



## Part 5: Practice questions

Attempt all the questions that follow:

1. Write a program that displays all the *odd numbers* between 0 and 100. The program should also display the sum of the odd numbers.
2. Write a program that will calculate and display the *Factorial* of 10.
3. Write a program that will display the *Fibonacci sequence* of 10.

## References

- [ 1 ] <https://jupyter.org/index.html>
- [ 2 ] [https://en.wikipedia.org/wiki/Project\\_Jupyter](https://en.wikipedia.org/wiki/Project_Jupyter)
- [ 3 ] <https://www.markdownguide.org/>

## End of exercise



## Exercise 7

# Data manipulation and analysis using Python

---

Part 1: NumPy overview . . . . .	81
Part 2: Pandas overview . . . . .	81
Part 3: Data extraction . . . . .	82
Part 4: Data preprocessing . . . . .	82
Part 4.1: Data cleansing . . . . .	82
Part 4.2: Feature selection . . . . .	82
Part 4.3: Feature engineering . . . . .	82
Part 6: Practice questions . . . . .	82
References . . . . .	83

---

## Estimated time

20:00 Hours

## Overview

In this exercise, you import `NumPy` and `Pandas` software libraries into a `Python` program and use them to perform numerical computations on data, manipulate and analyze data.

## Objectives

After completing this exercise, you should be:

- Install `Pandas` and `NumPy` software libraries.
- Use the `Pandas` and `NumPy` libraries to manipulate and analyze data.

## Prerequisites

Completed Exercise 6: *Jupyter interactive computing*.

## Requirements

This exercise requires a PC that has Internet access and, has a Web browser installed (i.e. `Chrome`).

## Exercise instructions

In this exercise, you complete the following tasks:

- Write and compile code excerpts as answers to exercise questions.
- Add a comment with the following information: **Name**, **Student Number**, and **Date**.
- Save your *notebook* as a PDF and upload it the link provided in the e-learning.



## Part 1: NumPy overview

NumPy is an open-source software package for scientific numerical computing using Python. NumPy offers powerful tools for N-dimensional array computing; a large collection of comprehensive mathematical functions (i.e. random number generators, linear algebra routines, Fourier transforms etc) to operate on these arrays. NumPy was originally created by Travis Oliphant in 2005.

### Installing NumPy

**Prerequisite:** Python software (installed in Exercise 5).

NumPy can be installed using a Package installer such as **conda** or **pip**. In this exercise, we use the **pip** package installer (usually installed together with Python) to install NumPy with:

```
pip install numpy
```

## Part 2: Pandas overview

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. The name is derived from the term “panel data”. Panel data is a subset of *longitudinal* data where observations belong to the same subject every time. Pandas was originally developed by Wes McKinney in 2008.

Python with Pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, etc.

Pandas library offers the following library features:

- A fast and efficient DataFrame object for data manipulation with integrated indexing;
- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible reshaping and pivoting of data sets;
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets;
- Columns can be inserted and deleted from data structures for size mutability;
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets;

- High performance merging and joining of data sets;
- Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- Time series-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data.

## Installing Pandas

**Prerequisite:** Python software (installed in Exercise 5).

Pandas can be installed via pip from PyPI (Python Package Index - the official third-party software repository for Python):

```
pip install pandas
```

## Part 3: Data extraction

## Part 4: Data preprocessing

### Part 4.1: Data cleansing

### Part 4.2: Feature selection

### Part 4.3: Feature engineering

## Part 6: Practice questions

## References

- [ 1 ] <https://Pandas.pydata.org/>
- [ 2 ] <https://NumPy.org/>
- [ 3 ] <https://en.wikipedia.org/wiki/NumPy>
- [ 4 ] [https://en.wikipedia.org/wiki/Pandas\\_\(software\)](https://en.wikipedia.org/wiki/Pandas_(software))
- [ 5 ] <https://pypi.org/project/numpy/>
- [ 6 ] <https://pypi.org/project/pandas/>
- [ 7 ] <https://jupyter.org/index.html>
- [ 8 ] [https://en.wikipedia.org/wiki/Project\\_Jupyter](https://en.wikipedia.org/wiki/Project_Jupyter)

## End of exercise