INTRODUCTION TO PYTHON

BMEG 591M

RAVNEET KAUR

AGENDA

- Introduction
- Installation
- Input / output
- Basic Data Types
- Containers
- Operators
- Assignment 1
- Control Statements
- Methods and Functions
- Assignment 2
- Packages and Modules
- Libraries

OUR SHARED SPACE

- Slides
- Code
- Assignments

https://github.com/ravubc/BMEG591MPython

INTRODUCTION

WHAT IS PYTHON?

- General purpose programming language often applied in scripting roles.

- So, Python is programming language as well as scripting language.

- Python is also called as Interpreted Language.

Who uses Python?

- Google
- NASA
- PBS
- The ONION

- And the list goes on…………

WHY PYTHON?

- Open source General-purpose language.
- Object-oriented, procedural, functional.
- The community provides many introductory resources.

Downloads: www.python.org

Documentation: www.python.org/doc/

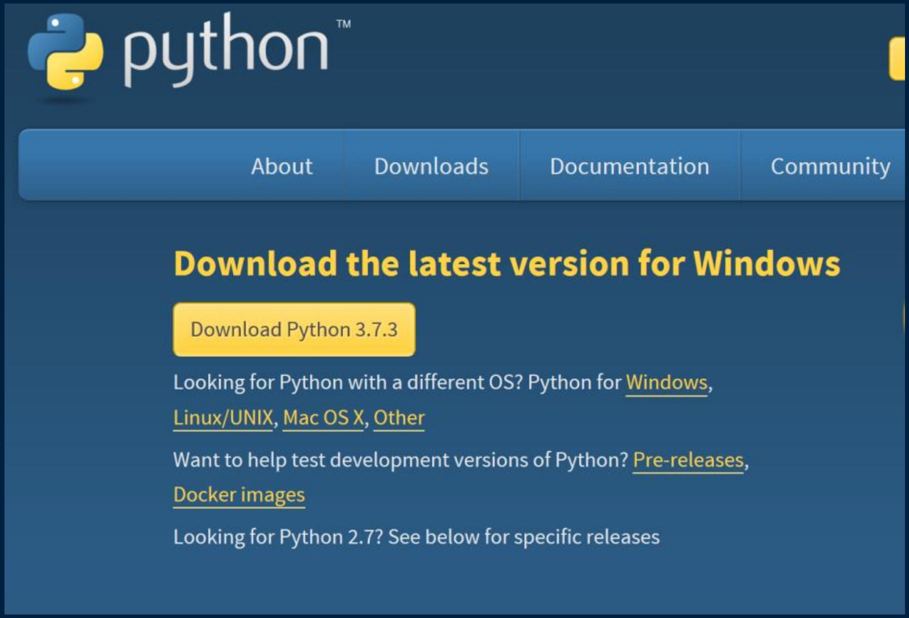
Community: www.python.org/community/

INSTALLATION

INSTALLATION

- Python comes pre-installed with Mac OS X and Linux.

- www.python.org/downloads/



Download the latest version for Windows

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

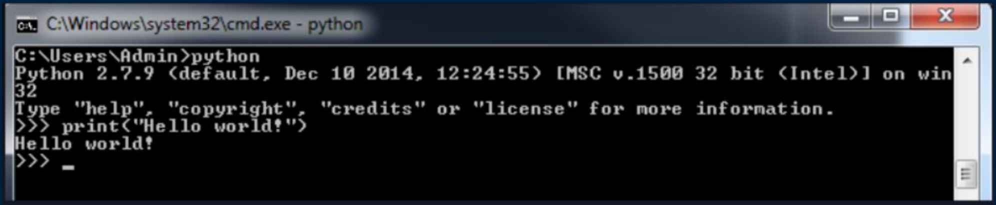
Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

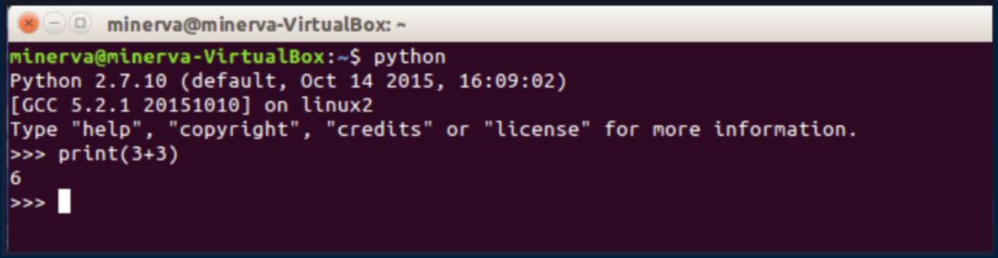
USING PYTHON

- Terminal or Command Prompt
- Jupyter/ Ipython
- Python IDE: PyCharm

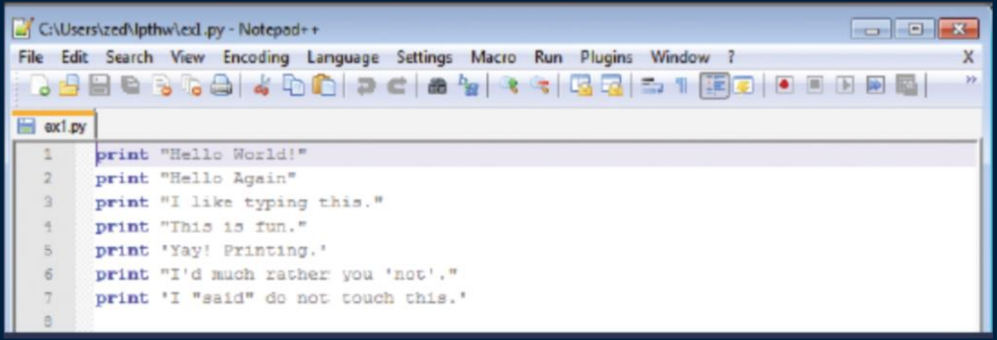
USING PYTHON: TERMINAL/COMMAND



```
C:\Users\Admin>python
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world!")
Hello world!
>>> _
```



```
minerva@minerva-VirtualBox:~$ python
Python 2.7.10 (default, Oct 14 2015, 16:09:02)
[GCC 5.2.1 20151010] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print(3+3)
6
>>> 
```

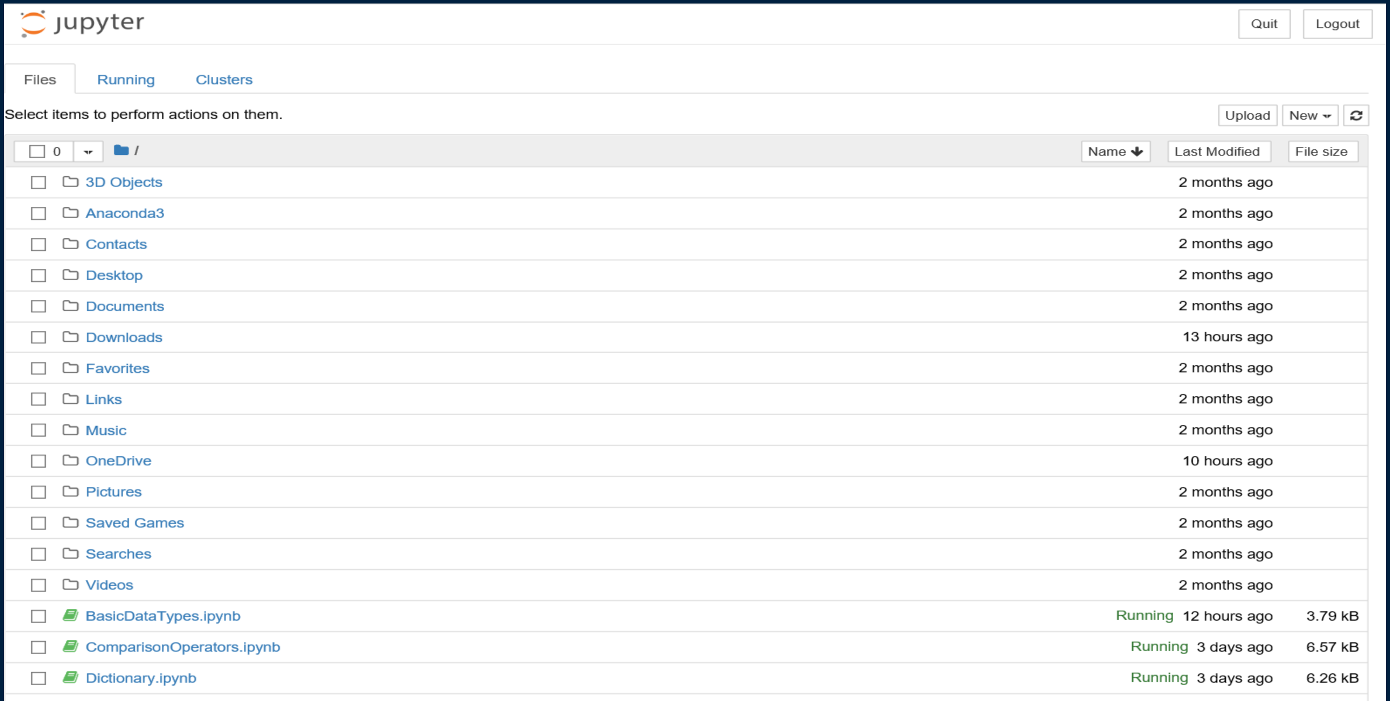

C:\Users\zed\lpthw\ex1.py - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
ex1.py

```
print "Hello World!"
print "Hello Again"
print "I like typing this."
print "This is fun."
print 'Yay! Printing.'
print "I'd much rather you 'not'."
print 'I "said" do not touch this.'
```

- Python prompts with '>>>'
- To exit Python():
- CTRL-D
- exit()

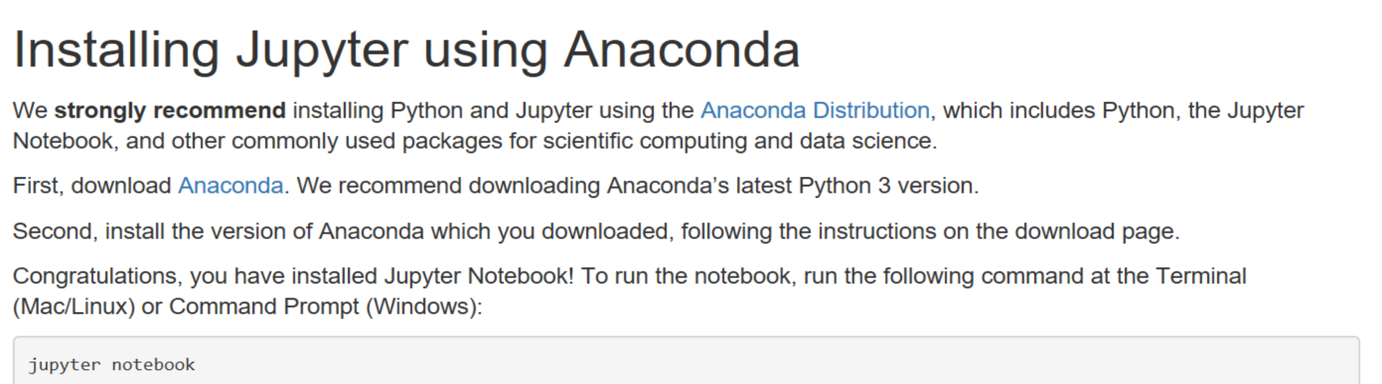
USING PYTHON/JUPYTER

https://try.jupyter.org/



Select items to perform actions on them.

Name		Last Modified	File size
3D Objects		2 months ago	
Anaconda3		2 months ago	
Contacts		2 months ago	
Desktop		2 months ago	
Documents		2 months ago	
Downloads		13 hours ago	
Favorites		2 months ago	
Links		2 months ago	
Music		2 months ago	
OneDrive		10 hours ago	
Pictures		2 months ago	
Saved Games		2 months ago	
Searches		2 months ago	
Videos		2 months ago	
BasicDataTypes.ipynb	Running	12 hours ago	3.79 kB
ComparisonOperators.ipynb	Running	3 days ago	6.57 kB
Dictionary.ipynb	Running	3 days ago	6.26 kB



Installing Jupyter using Anaconda

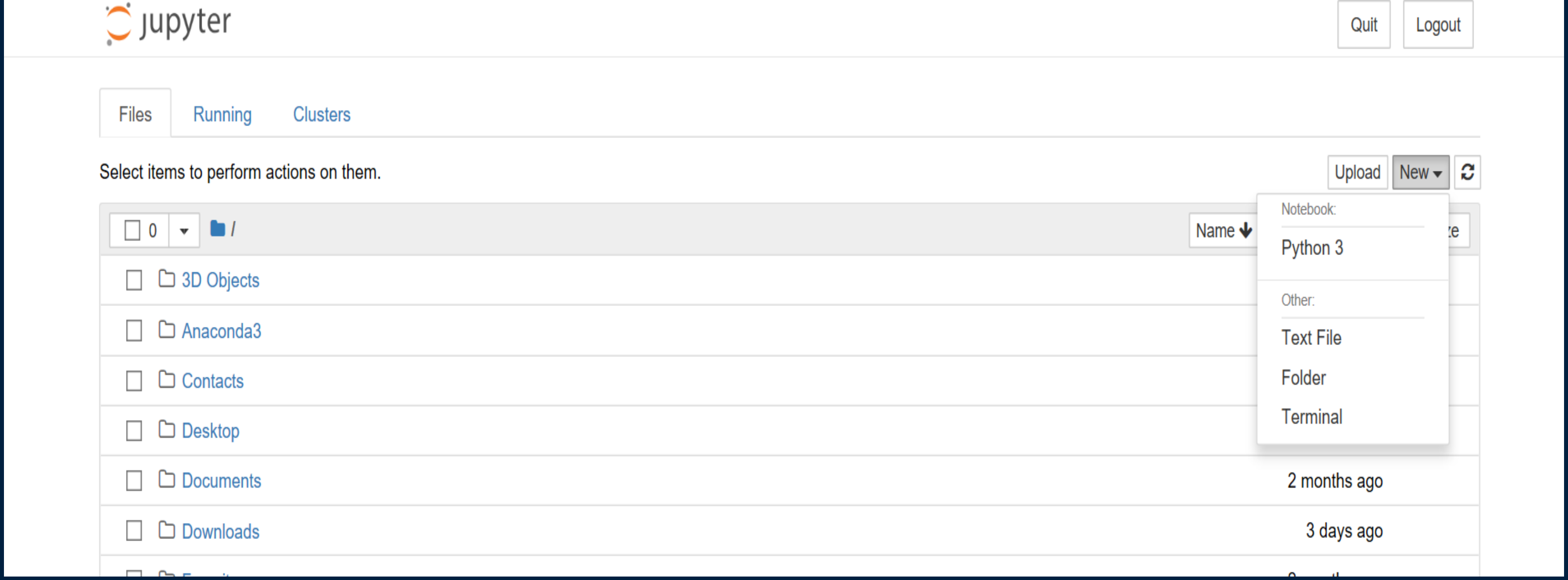
We **strongly recommend** installing Python and Jupyter using the Anaconda Distribution, which includes Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

First, download Anaconda. We recommend downloading Anaconda's latest Python 3 version.

Second, install the version of Anaconda which you downloaded, following the instructions on the download page.

Congratulations, you have installed Jupyter Notebook! To run the notebook, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows):

```
jupyter notebook
```



jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them.

Upload New ▾

□ 0 ▾ /

Name ↓

Notebook:

Python 3

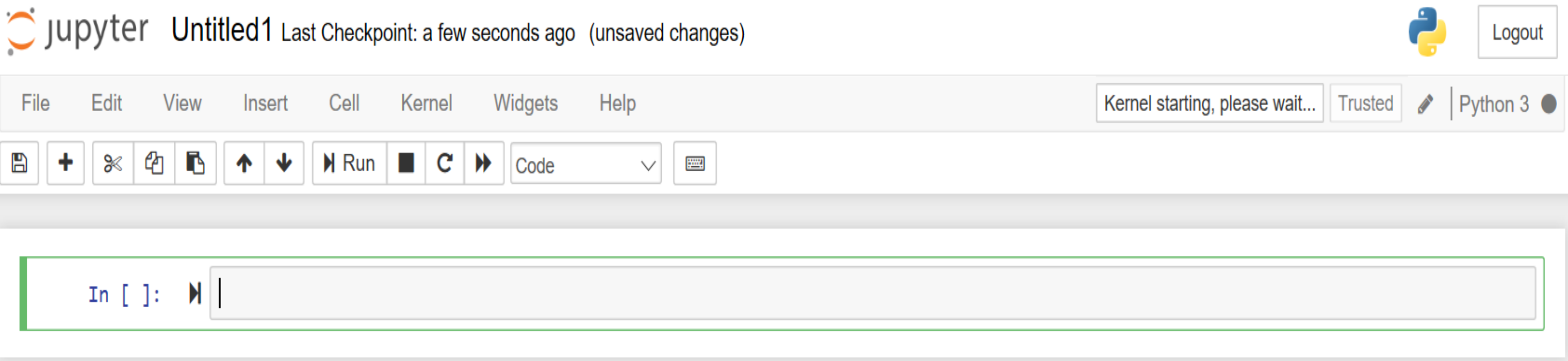
Other:

Text File

Folder

Terminal

□	3D Objects	
□	Anaconda3	
□	Contacts	
□	Desktop	
□	Documents	2 months ago
□	Downloads	3 days ago



jupyter Untitled1 Last Checkpoint: a few seconds ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

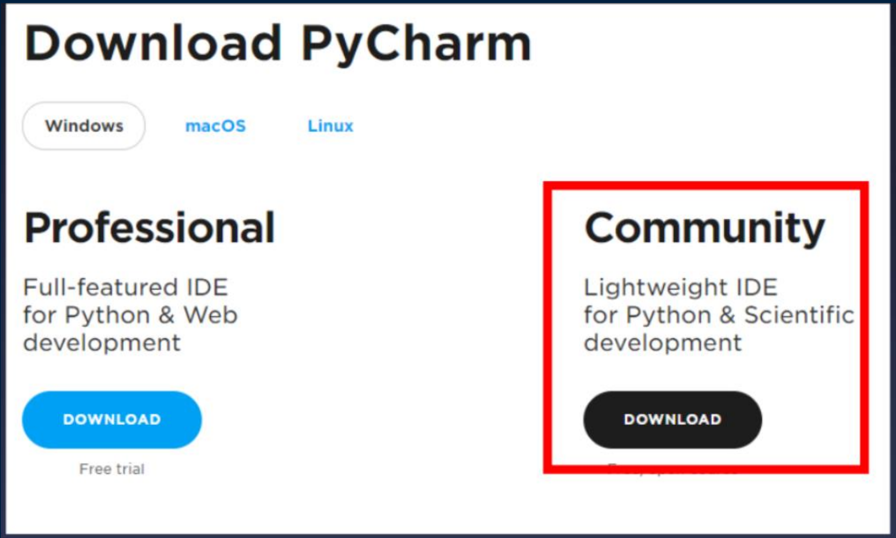
Kernel starting, please wait... Trusted Python 3 ●

Run Code

```
In [ ]:
```

USING PYTHON/ PYCHARM

- https://www.jetbrains.com/pycharm/download/



Download PyCharm

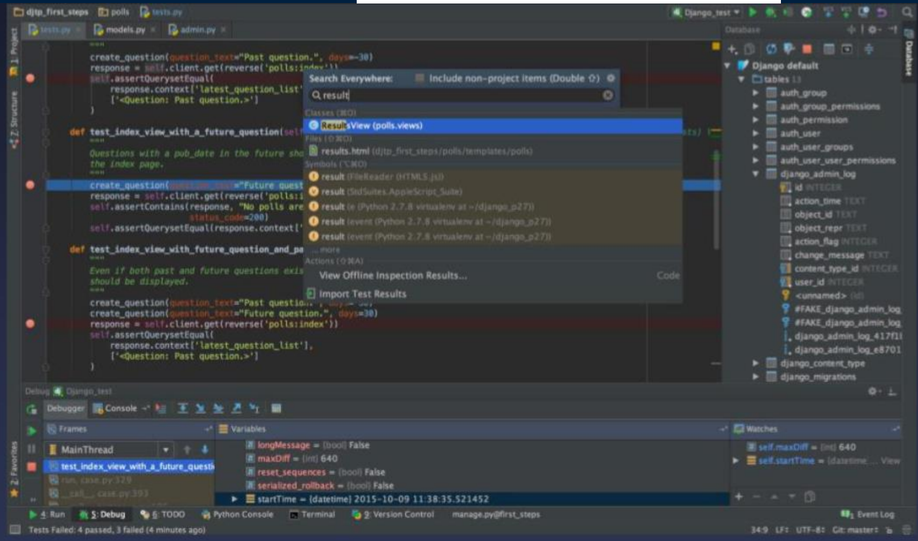
Professional

Full-featured IDE
for Python & Web
development

Free trial

Community

Lightweight IDE
for Python & Scientific
development



```
        """
        create_question(question_text="Past question.", days=-30)
        response = self.client.get(reverse('polls:index'))
        self.assertQuerysetEqual(
            response.context['latest_question_list'
            ['<Question: Past question.>']
        )

    def test_index_view_with_a_future_question(self
        """
        Questions with a pub_date in the future sho
        the index page.
        """
        create_question(question_text="Future quest
        response = self.client.get(reverse('polls:i
        self.assertContains(response, "No polls are
                            status_code=200)
        self.assertQuerysetEqual(response.context['

    def test_index_view_with_future_question_and_pa
        """
        Even if both past and future questions exis
        should be displayed.
        """
        create_question(question_text="Past questio
        create_question(question_text="Future question.", days=30)
        response = self.client.get(reverse('polls:index'))
        self.assertQuerysetEqual(
            response.context['latest_question_list'],
            ['<Question: Past question.>']
        )
```

Search Everywhere: Include non-project items (Double ⇧)

result

Classes (⌘O)
- ResultsView (polls.views)

Files (⇧⌘O)
- results.html (djtp_first_steps/polls/templates/polls)

Symbols (⌥⌘O)
- result (FileReader (HTML5.js))
- result (StdSuites.AppleScript_Suite)
- result (e (Python 2.7.8 virtualenv at ~/django_p27))
- result (event (Python 2.7.8 virtualenv at ~/django_p27))
- result (event (Python 2.7.8 virtualenv at ~/django_p27))
- ... more

Actions (⇧⌘A)
- View Offline Inspection Results... Code
- Import Test Results

Django default
- tables 13
  - auth_group
  - auth_group_permissions
  - auth_permission
  - auth_user
  - auth_user_groups
  - auth_user_user_permissions
  - django_admin_log
    - id INTEGER
    - action_time TEXT
    - object_id TEXT
    - object_repr TEXT
    - action_flag INTEGER
    - change_message TEXT
    - content_type_id INTEGER
    - user_id INTEGER
    - <unnamed> (id)
    - #FAKE_django_admin_log
    - #FAKE_django_admin_log
    - django_admin_log_417f1
    - django_admin_log_e8701
  - django_content_type
  - django_migrations

Frames:
- MainThread
- test_index_view_with_a_future_questi
- run, case.py:329
- __call__, case.py:393

Variables:
```
longMessage = {bool} False
maxDiff = {int} 640
reset_sequences = {bool} False
serialized_rollback = {bool} False
startTime = {datetime} 2015-10-09 11:38:35.521452
```

Watches:
```
self.maxDiff = {int} 640
self.startTime = {datetime ... View
```

Tests Failed: 4 passed, 3 failed (4 minutes ago)

34:9 LF: UTF-8: Git: master:

PYTHON PROGRAMS

- Python programs or modules are written as text files with traditionally a .py extension.

- Each python module has its own namespace.

- Python programs and modules are differentiated by the way they are called.

✓.py files executed directly are programs

✓.py files referenced via import statement are modules

DYNAMIC TYPING

Python uses *dynamic typing*, meaning you can reassign variables to different data types. This makes Python very flexible in assigning data types; it differs from other languages that are *statically typed*.

Pros of Dynamic Typing

- Very easy to work with
- Faster development time

Cons of Dynamic Typing

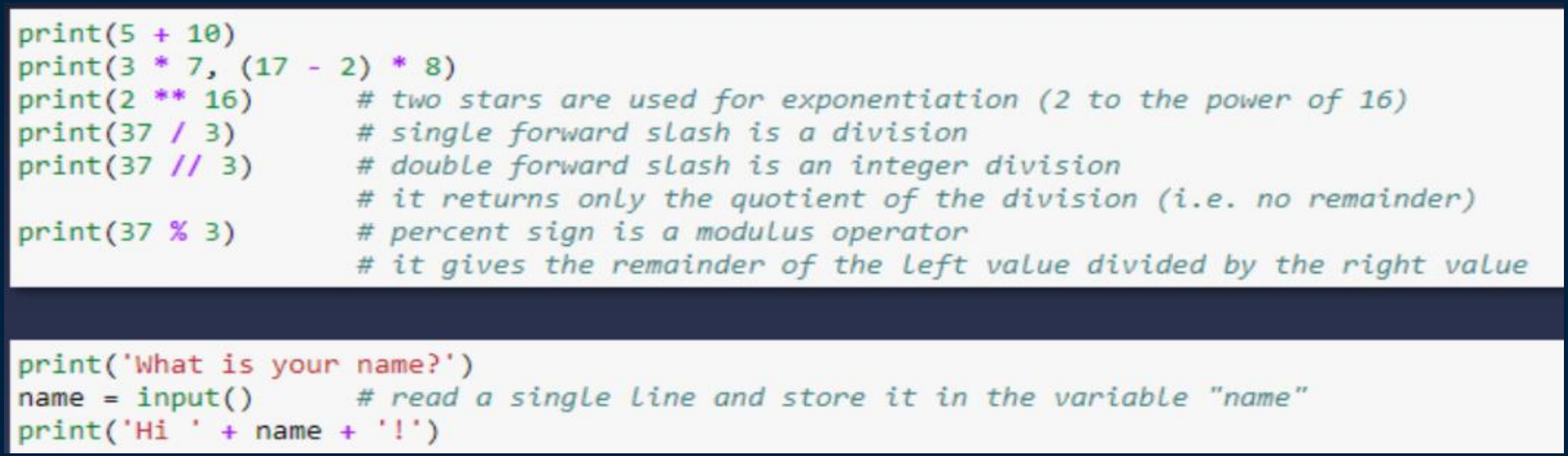
- May result in unexpected bugs!
- You need to be aware of type()

INPUT/ OUTPUT

INPUT / OUTPUT

Input: input()

Output: print()



```
print(5 + 10)
print(3 * 7, (17 - 2) * 8)
print(2 ** 16)      # two stars are used for exponentiation (2 to the power of 16)
print(37 / 3)       # single forward slash is a division
print(37 // 3)      # double forward slash is an integer division
                    # it returns only the quotient of the division (i.e. no remainder)
print(37 % 3)       # percent sign is a modulus operator
                    # it gives the remainder of the left value divided by the right value
```

```
print('What is your name?')
name = input()      # read a single line and store it in the variable "name"
print('Hi ' + name + '!')
```

InputOutput.ipynb

FILES

- Python uses file objects to interact with external files on your computer.

- These can be text files, emails, audio file, excel documents etc.

- Creating a file:

  ```
  %%writefile new_file.txt
  ```

BASIC DATA TYPES

BASIC DATA TYPES

- Numbers

  Integers and float numbers are similar as in other languages.

- Strings

  Strings have various in-built operations.

- Booleans

  Python implements all of the usual operators for Boolean logic, but uses English words rather than symbols (&&, ||, etc.)

DETERMINING VARIABLE TYPE WITH TYPE()

- **int** (for integer)
- **float**
- **str** (for string)
- **list**
- **tuple**
- **dict** (for dictionary)
- **set**
- **bool** (for Boolean True/False)

BasicDataTypes.ipynb

NUMBERS IN PYTHON

1. Types of Numbers
2. Basic Arithmetic
3. Difference between division and floor division
4. Object Assignment in Python

VARIABLE ASSIGNMENT

- Names cannot start with a number
- Names can not contain spaces, use _ instead
- Names can not contain any of these symbols: :'",<>/?|\!@#%^&*~-+
- It's considered best practice that names are lowercase with underscores
- Avoid using Python built-in keywords like list and str
- Avoid using the single characters l (lowercase letter el), O (uppercase letter oh) and I (uppercase letter eye) as they can be confused with 1 and 0

numbers.ipynb

STRINGS

- Record text information
- Strings in Python are actually a sequence.

- “Hello”----→ Sequence of letters
- H----- 1
- e----- 2

- And so on….

strings.ipynb

CONTAINERS

CONTAINERS

- Containers are any object that holds an arbitrary number of other objects.

- Containers provide a way to access the contained objects and to iterate over them.

✓Lists

✓Dictionaries

✓Tuples

LISTS

- Like Strings

- Except

  Lists are mutable, elements inside a list can be changed.

Lists.ipynb

DICTIONARIES

- Like Hash Tables
- Objects are stored by the key instead of their relative position

- Example: dict = {'key1':'value1'}

dictionary.ipynb

TUPLES

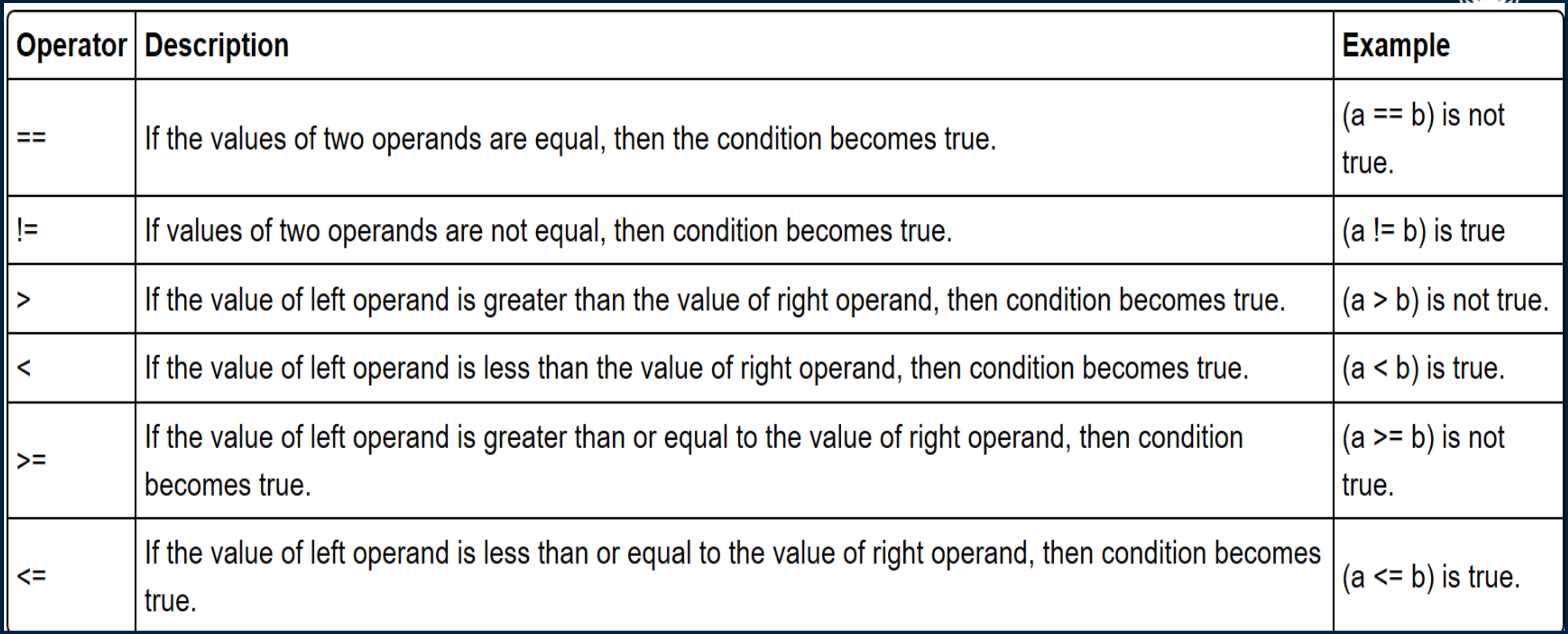
- Tuples are very similar to lists
- However, unlike lists they are *immutable* meaning they can not be changed.

- The construction of a tuples use () with elements separated by commas.
- Example: tup=(1,2,3)

Tuples.ipynb

OPERATORS

COMPARISON OPERATORS



Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

CHAINED COMPARISON OPERATORS

- Python has the ability to *chain* multiple comparisons to perform a more complex test.

- Example: 1 < 2 < 3

operators.ipynb

ASSIGNMENT 1

1. Create a dictionary with 4 key-value pairs (e.g., populate with "fruit":"price").

Print the keys of the dictionary.

Print the length of the dictionary.

Add one more key-value pair to the dictionary.

Delete one key-value pair from the dictionary.

Print the length of the keys of the dictionary.

2. Write a program to check if the program is palindrome or not.

CONTROL STATEMENTS

CONTROL STATEMENTS

- Control Flow statements makes use of colons and indentation(Whitespace)
- Indentation sets Python apart from other languages.

- If
- Elif
- else

IF STATEMENT

- **Other Language**

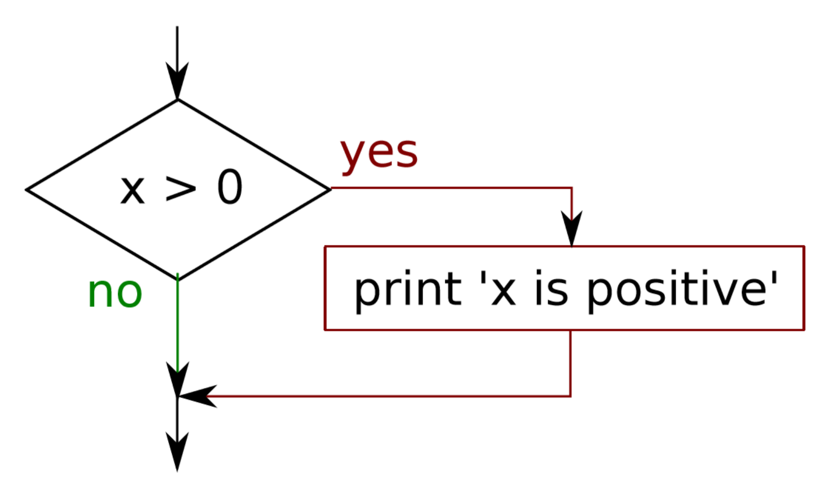
```
if (a>b){
      a = 2;
      b = 4;
}
```

- **Python**

```
if a>b:
          a = 2
          b = 4
```

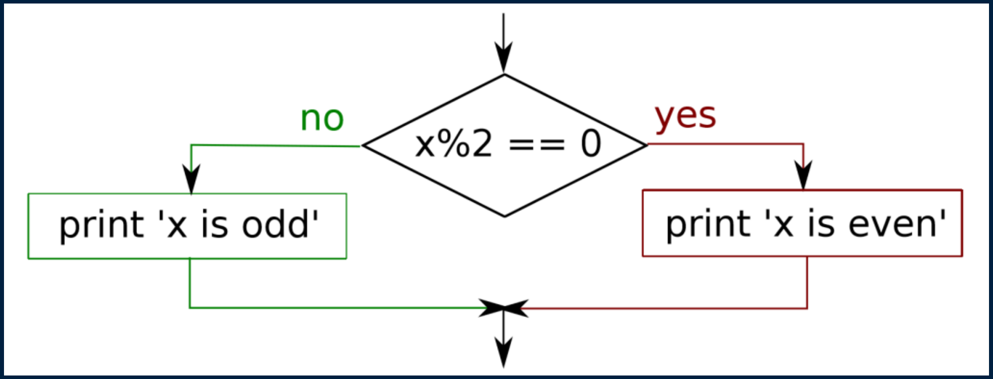
Example:

```
if x > 0 :
          print 'x is positive’
```



IF-ELSE

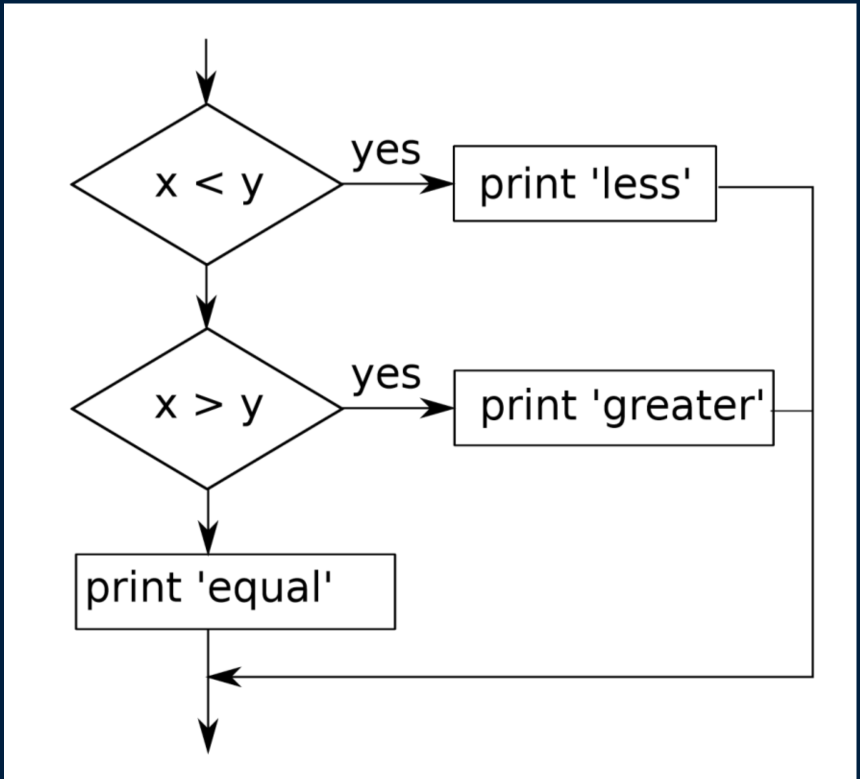
```
if x%2 == 0 :
        print 'x is even’
else :
        print 'x is odd'
```



ifelse.ipynb

CHAINED CONDITIONALS- ELIF

```
if x < y:
    print 'x is less than y’
elif x > y:
    print 'x is greater than y’
else:
    print 'x and y are equal'
```



FOR LOOP

- It goes through items that are in a *sequence* or any other iterable item.

```
for item in object:
    statements to do stuff
```

- The **object** can be a tuple, dictionary, list, strings etc.

ForLoop.ipynb

WHILE LOOP

- *while* statement will repeatedly execute a single statement or group of statements as long as statement is true.
- The general format of a while loop is:

```
while test:
    code statements
else:
    final code statements
```

While.ipynb

METHODS AND FUNCTIONS

METHODS AND FUNCTIONS

- **Methods** are in-built functions built into the objects.
- Perform specific actions on objects
- Methods take arguments like functions.

object.method(arg1,arg2,etc...)

Example: Lists

- append
- count
- extend
- insert
- pop
- remove
- reverse
- sort

FUNCTIONS

- Function Definition consists of
- **A header and the body**

- **Function Definition**

```
def my_function():
    “Function Documentation”
    print (“hello world”)
```

FUNCTION ARGUMENTS

- **Positional**

```
def add(x,y):
    return x+y
```

- **Keyword**

```
def shout(phrase=“Yipee!”):
    print phrase
```

methodsFunction.ipynb

ASSIGNMENT 2

1. Check if number is even or not.

2. Write a function named pig_latin

If the word starts with a vowel, add ‘ay’ to the end

If the word does not start with a vowel, put first letter at the end, then add ‘ay’

Word→ orday

Apple→appleay

PACKAGES

PACKAGES

Using PyPi with pip install

- PyPi is a repository for open-source third-party Python packages.
- There are many libraries available open-source and shared on PyPi.
- We can use *‘pip install’* at command line to install these packages.
- pip comes pre-installed from python.org.
- pip is used to download packages at command line directly from PyPi repository.

INSTALLING PACKAGES

- Windows Users: Command Prompt
- MacOS/Linux Users: Terminal

- *pip install PACKAGE_NAME*

MODULES

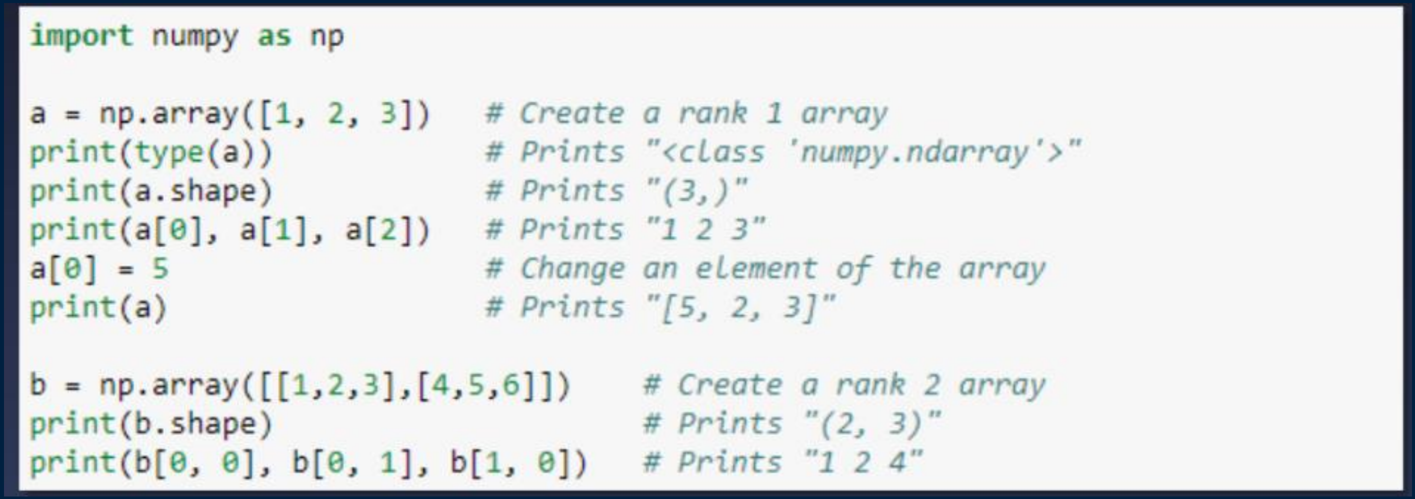
MODULES

- Simply put, **modules are programs**
- Write some code in a file, save it, run it from command line.
- **python hello.py** is an example run, if your file was named "hello".
- Always have the **.py** extension to your Python files.

NUMPY

- Numpy is the core library for scientific computing in python.

- It provides a high-performance multi-dimensional array object, and tools for working with these arrays.



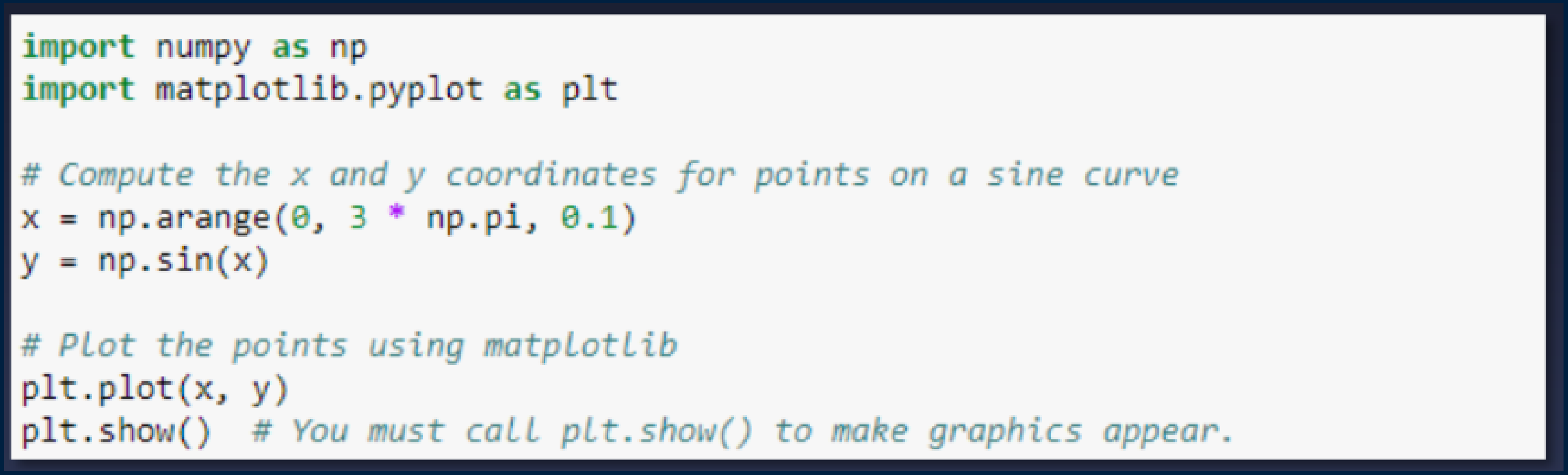
```
import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

MATPLOTLIB

Matplotlib is the plotting library.



```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# Plot the points using matplotlib
plt.plot(x, y)
plt.show()  # You must call plt.show() to make graphics appear.
```

ASSIGNMENT 3

1. Starting from a fresh file, write a function **square_nums** that takes a list of numbers between 0 and 10 as input, squares each it in the list, and returns another list with all the squared numbers.

- Call your function, harvesting the returned squared list in a variable **my_squared_list**.
- Print the items of **my_squared_list**
- Save your file as **square.py**
- From the command line, run your script (Hint: run **python square.py**)

2. Run the function "yearly_raise()" on a salary = 85000

- Create a database of book prices with entries
- The database can be a dict, with "key" as book name and "value" as price
- Create a "sale" function that returns the price of a book after a 10% di scount of its total price
- Use the funtion to print all the 5 entries in the database and their new

LEARN MORE

- **PYTHON TUTORIALS AND COURSES**

https://hackr.io/tutorials/learn-python

https://github.com/wzpan/Learn-Python-The-Hard-Way/tree/master/Python2

- **PYTHON 3 PROGRAMMING INTRODUCTION TUTORIAL**

https://pythonprogramming.net/introduction-to-python-programming/

- **PYTHON TUTORIAL FOR BEGINNERS**

https://thepythonguru.com/

THANK YOU !!

REFERENCE

1. https://github.com/AziziShekoofeh/Introduction-to-Python-Programming/
2. Geeksforgeeks.org
3. Python.org

THE UNIVERSITY OF BRITISH COLUMBIA