

Programming for Data Analytics

Week1: Introduction to programming in Python



Dr. Mohammed Hasanuzzaman

Room J102, Melbourne Building (Office)

e-mail:mohammed.hasanuzzaman@cit.ie

[Website: https://mohammedhasanuzzaman.github.io](https://mohammedhasanuzzaman.github.io)

- Educational Background
 - **B.Tech**, University of Kalyani, India
 - **M.E**, West Bengal University of Technology, India
 - **PhD**, French National Centre for Scientific Research (CNRS), Ecole Nationale Supérieure d'Ingénieurs de Caen – ENSICAEN and University of Caen Normandie, France
- Key Experiences
 - **Assistant Professor**, IIT Jammu, India
 - **Postdoctoral Researcher**, ADAPT Centre, Dublin, Ireland
 - **Guest Researcher**, L3S Research Centre, Hanover, Germany
 - **NLP Researcher**, BrandCrafter Ltd., Hong Kong
 - **Officer-on-Special Duty (OSD)**, Govt. of West Bengal, India

- Research Interests
 - **Natural Language Processing (NLP)** especially Information Extraction and Information Retrieval, Knowledge Representation, Machine Translation, Psycholinguistics, Social Media Analytics and Machine Learning (deep learning) applications to NLP
- Academic Activities
 - Teaching
 - Statistical Machine Translation, Dublin City University, Ireland (2017, 2018)
 - Introduction to Machine Learning, Dublin City University, Ireland (2019)
 - Student Supervision
 - PhD (Ongoing)-2 (IIT Patna, India)
 - Master (completed)- 14 (8 From DCU and 6 from IIT Patna)
 - Undergraduate-2 (IIT Patna)

Acknowledgments

I am grateful to Dr. Ted Scully and Dr. Haithem Afli who shared with me the following material.

Module Content

- Introduction to Python Programming Concepts
 - Decision Structures, Iteration, Functions, Lists, Dictionaries, Tuples, Files, Objects
- Development of Python Programs for Solving Data Driven Problems

 - NumPy, Pandas, SciPy, Mathplotlib, Seaborn



Assessment (100% CA)

- Practical Skills Evaluation (20%)
 - Week 6 (2hr Lab Assessment) – (open book)
- Project Work (80%)
 - Project A – Week 8 (Due Week 9) – 30%
 - Project B – Week 10 (Du Week 12) – 50%

Useful Websites

- Python Documentation
 - <http://www.python.org/doc/>
- A Byte of Python
 - <https://www.gitbook.com/book/swaroopch/byte-of-python/details>
- New Boston Python Video Tutorials
 - <https://thenewboston.com/videos.php?cat=98>
- Python Data Analysis Library
 - <http://pandas.pydata.org/>
- Python Scientific Library
 - <http://www.scipy.org/>
- Python Numerical Library
 - <http://www.numpy.org/>
- Continuum Analytics (Download Anaconda)
 - <http://continuum.io/downloads>

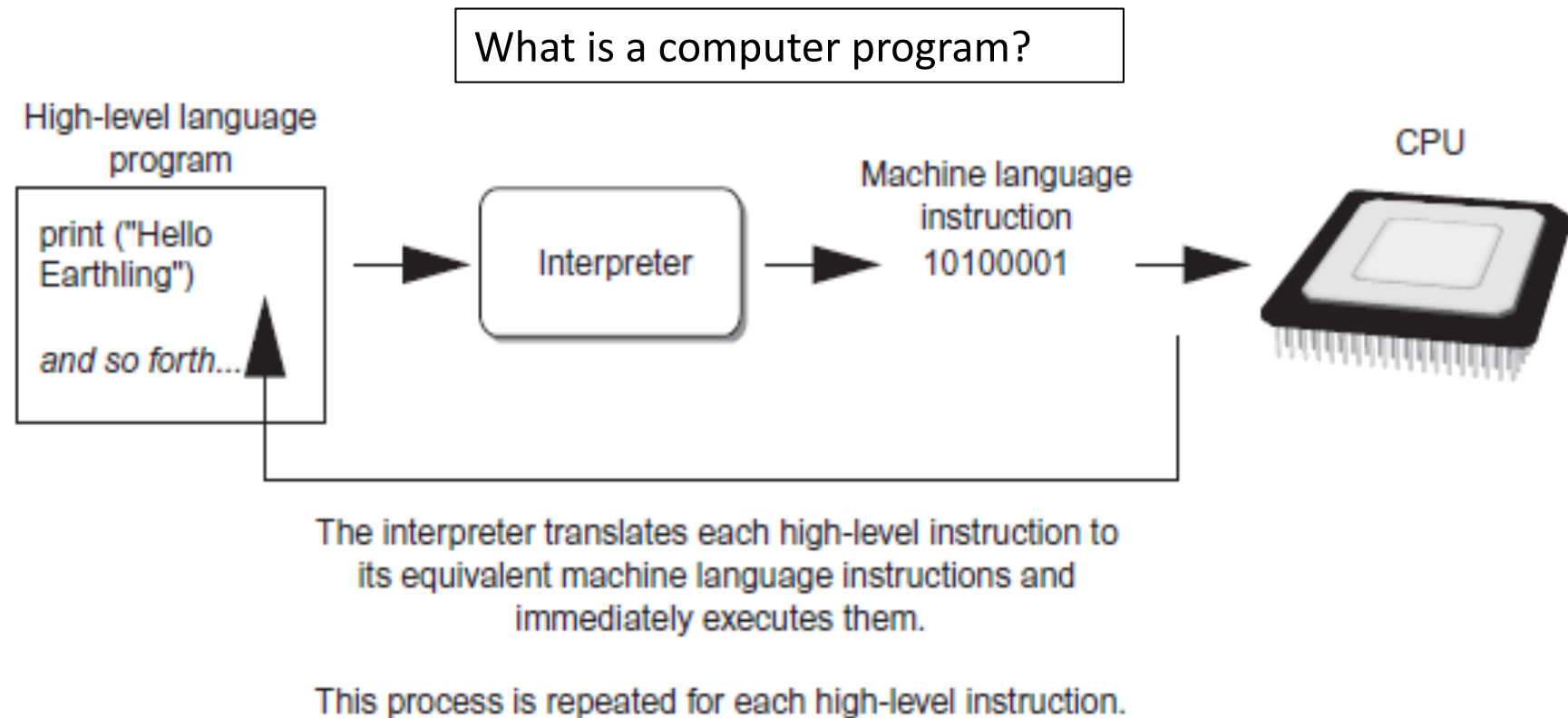
Overview



Overview of Python and its runtime environment

- Input/Out and Processing in Python
- Data Types
- Operators
- Selection

Python is an Interpreted Language



Executing a high-level program with an interpreter

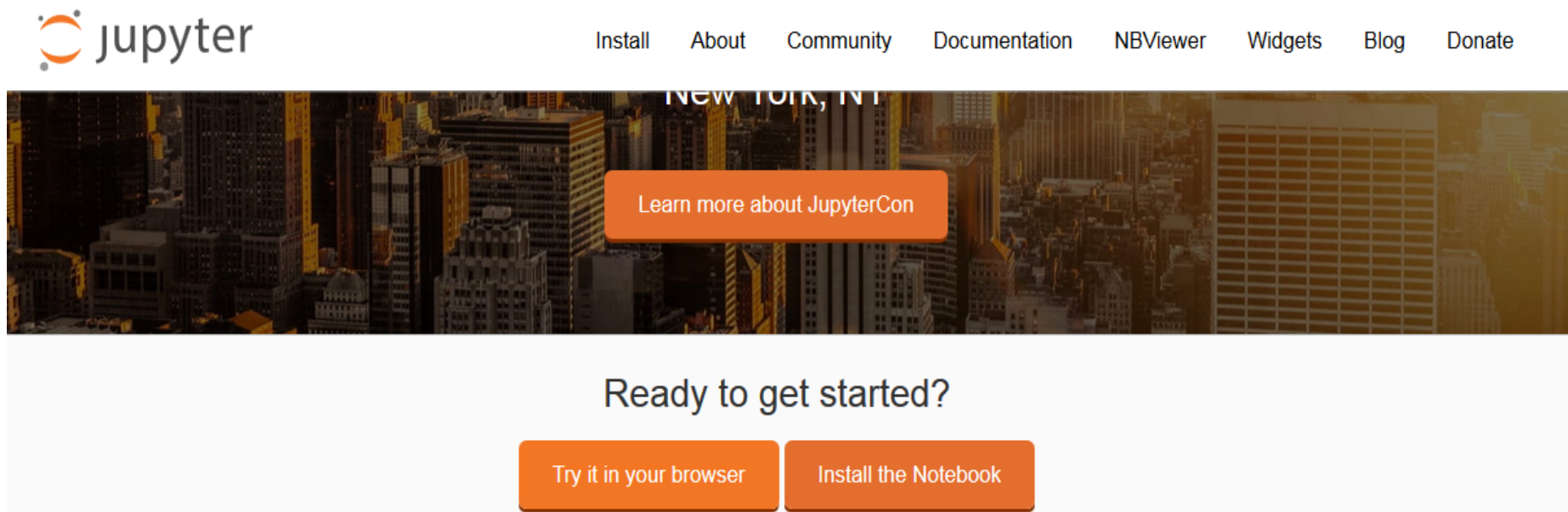
Using Python

- You can download a version Python at <http://www.python.org/download/>
 - Current stable version is Python 3.6. (you can use 3.7.0)
- We will be using an integrated development environment. There is a range of IDEs available for Python
 - Syedr
 - PyCharm
 - Geany
 - DrPython
 - SPE

Aside!

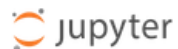
Jupyter Notebooks


- Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code.
- Go to <http://jupyter.org/> and click “Try it in your browser” as shown below.



Jupyter Notebooks

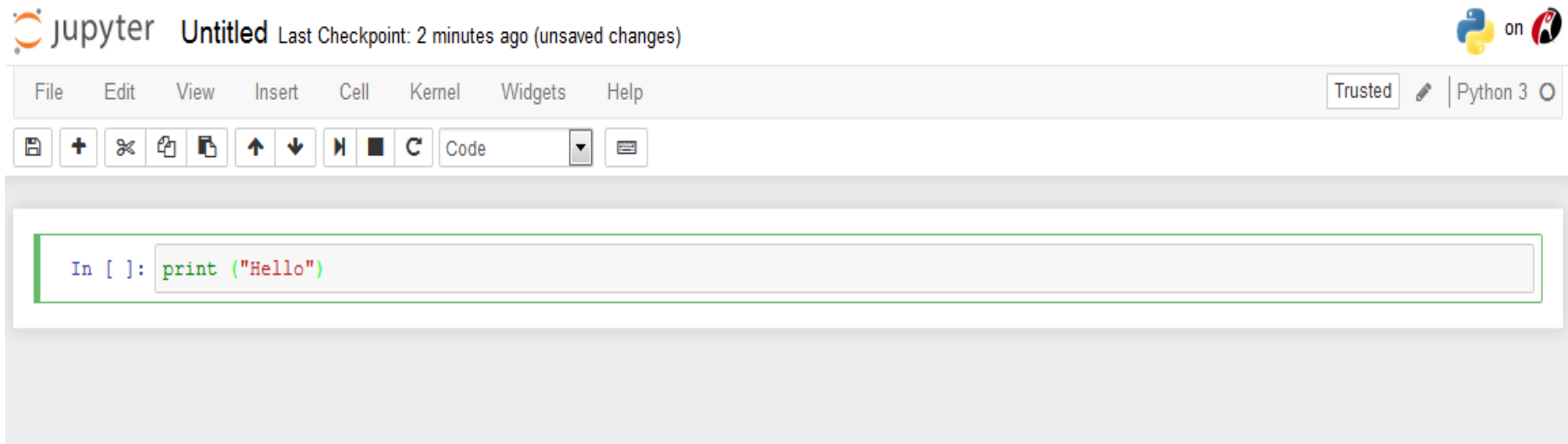
- You can create a Jupyter notebook for a range of programming languages as you see below.
- It's a great way for learning a language as it provides you with an interactive shell that allows you to type/run commands and see the output.




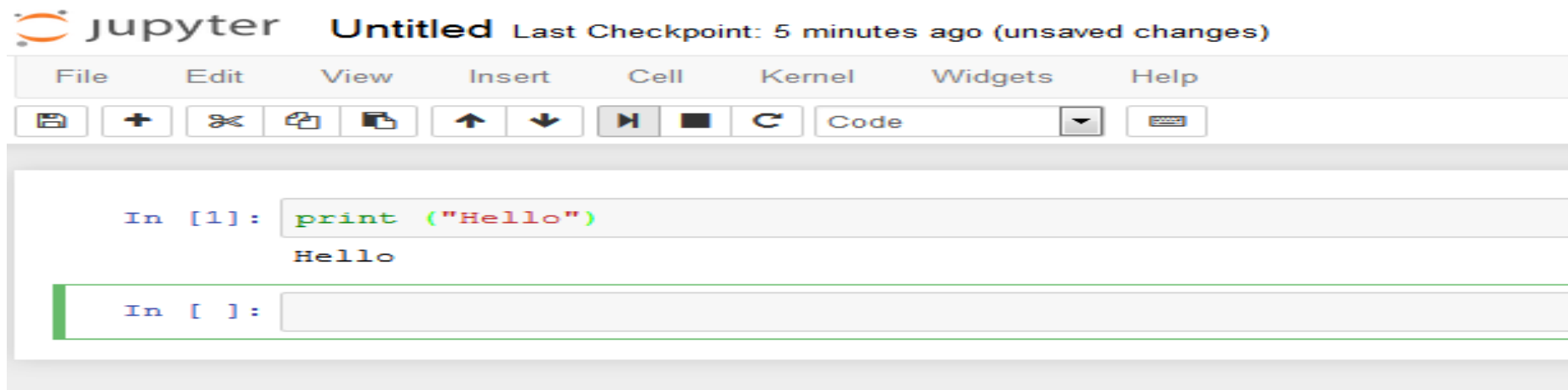
Hosted by Rackspace 



The screenshot shows the Jupyter web interface. At the top, there are tabs for 'Files', 'Running', and 'Clusters'. Below the tabs, a message says 'Select items to perform actions on them.' The main area displays a list of files and folders: 'communities', 'datasets', 'featured', and several 'Welcome' notebooks for Julia, R, Haskell, Python, and Spark. On the right side, there is a toolbar with 'Upload', 'New', and a refresh icon. The 'New' dropdown menu is open, showing a list of options: 'Notebook:', 'Apache Toree - Scala', 'Bash', 'Haskell', 'Julia 0.5.2', 'Python 2', 'Python 3' (which is highlighted), 'R', 'Ruby 2.1.5', 'spylon-kernel', 'Other:', 'Text File', 'Folder', and 'Terminal'. A tooltip next to 'Python 3' says 'Create a new notebook with Python 3'.



- The screenshot above shows my notebook and I've typed a simple command `print ("Hello")`. I can click the run cell button () and it will run this line of code and display the result. See below.



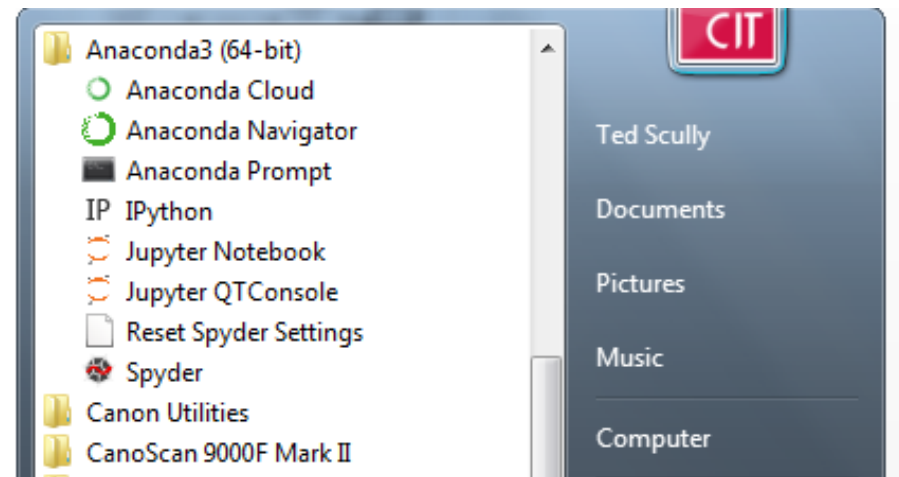
Jupyter Notebooks

- It's a simple easy way to learn a new language.
- You can compose an save an entire notebook and share them easily online.
- Very popular online. Useful for running exploratory data analysis.
- Not suitable for serious software development.

Using Python with the Spyder IDE

- The IDE we will be using is called Spyder and is part of the Anaconda package.

- <https://www.anaconda.com/download/>



Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Ted.Scully\Desktop\ML Class Material

logistic_new.py process.py 2 bloodPressure_MultiVariate_WithVectors.py Test.py bloodPressure

```
1 import numpy as np
2 import pandas as pd
3 import os
4
5 # so scripts from other folders can import this file
6 dir_path = os.path.abspath(os.path.dirname(os.path.realpath(__file__)))
7
8 # normalize numerical columns
9 # one-hot categorical columns
10
11 def get_data():
12     df = pd.read_csv(dir_path + '/eComData.csv')
13
14     # just in case you're curious what's in it
15     # df.head()
16
17     # easier to work with numpy array
18     data = df.as_matrix()
19
20     X = data[:, :-1]
21     Y = data[:, -1]
22
23     # normalize columns 1 and 2
24     X[:, 1] = (X[:, 1] - X[:, 1].mean()) / X[:, 1].std()
25     X[:, 2] = (X[:, 2] - X[:, 2].mean()) / X[:, 2].std()
26
27     # create a new matrix X2 with the correct number of columns
28     N, D = X.shape
29     X2 = np.zeros((N, D+3))
30     X2[:, 0:(D-1)] = X[:, 0:(D-1)] # non-categorical
31
32     # one-hot
33     for n in range(N):
34         t = int(X[n, D-1])
35         X2[n, t+D-1] = 1
36
37     # method 2
38     # Z = np.zeros((N, 4))
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing `<object>?` in the console.

Variable explorer File explorer Help

IPython console

Console 1/A

Python 3.6.0 |Anaconda custom (64-bit)| (default, Dec 23 2016, 11:57:41) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.

? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]:

Python console History log IPython console

Python Interpreter

- Python interpreter can be used in two modes
 - Interactive mode: enter statements on keyboard
 - Script mode: save statements in Python script

The screenshot shows the Anaconda IDE interface. The main editor window displays a Python script with the following code:

```
can import this file
ath.dirname(os.path.realpath(__file__)))

'/eComData.csv')

us what's in it

array

mean() / X[:,1].std()
mean() / X[:,2].std()

th the correct number of columns

1 # non categorical
```

The IPython console is open at the bottom, showing the following output:

```
Python 3.6.0 |Anaconda custom
v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "
for more information.

IPython 5.1.0 -- An enhance
? -> Introducti
%quickref -> Quick
help -> Pyth
object? -> D
about 'object', use 'object??' for extra details.

In [0]:

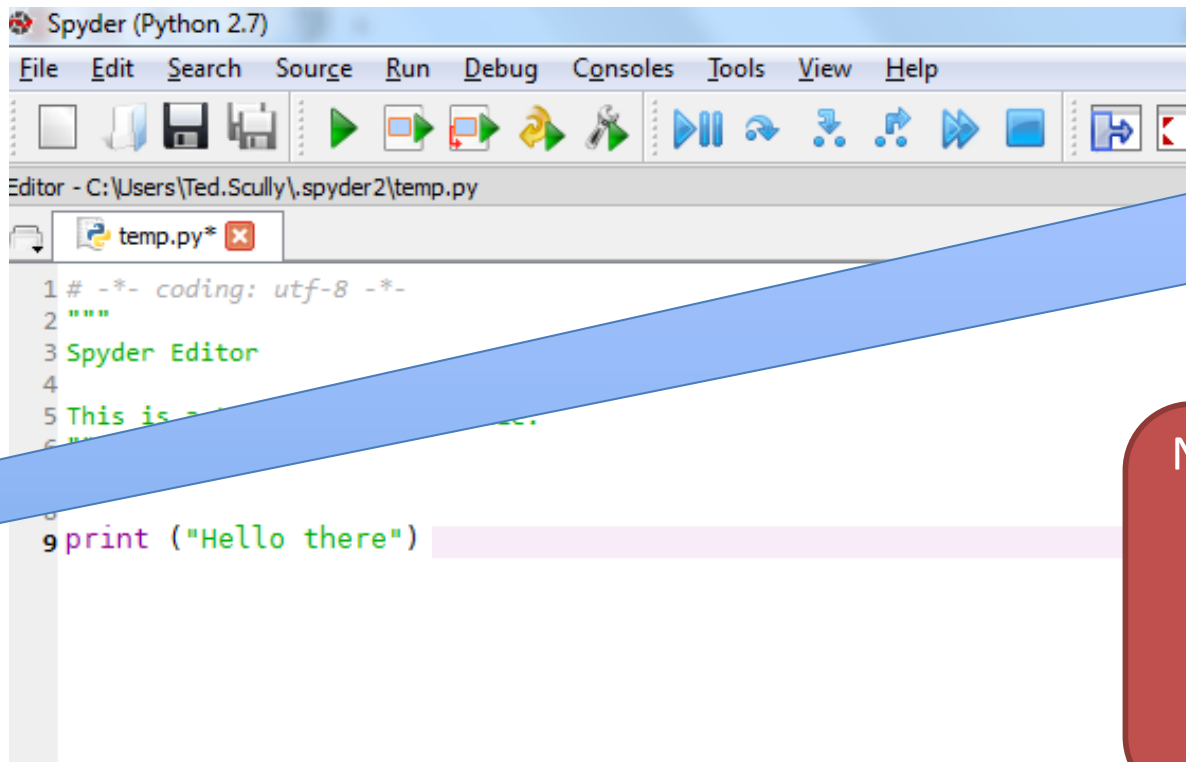
In [ ]: print ("Hello")
```

A blue callout box with a pointer to the console contains the text:

Interactive Mode. Enter commands at the command line.

Sypder – Scripting Mode

- When you open Sypder initially it will open a Python file (probably called untitled.py or temp.py).
- You can write lines of Python code into this file.



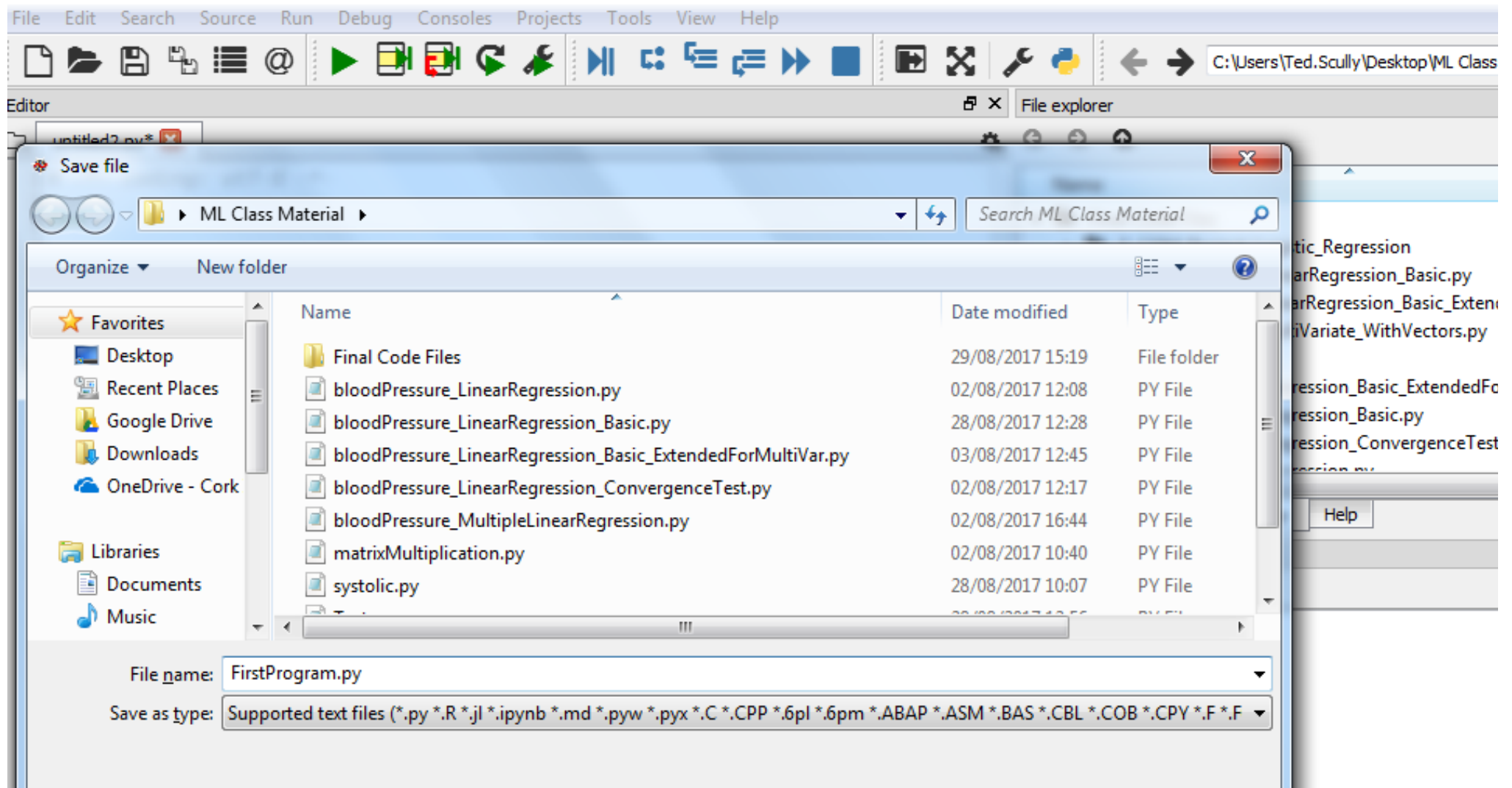
Your Python code can be written here.
It can then be saved as a file to the disk.

Note: When saving your Python file you should save it in the format **filename.py**

Python files should have a .py extension

Spyder – Scripting Mode – Saving a File

Click File – Save As



Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Run F5
Run cell Ctrl+Return
Run cell and advance Shift+Return
Run selection or current line F9
Re-run last script F6
Configure... Ctrl+F6
Profile F10

Editor

FirstProgram.py

```
1 # -*- coding: utf-8 -*-  
2  
3 """  
4 @author: Ted.Scully  
5 """  
6  
7  
8 print ("Hello there")  
9
```

File explorer

Name

- Final Code Files
 - E_COM_Project_Logistic_Regression
 - 1_bloodPressure_LinearRegression_Basic.py
 - 2 bloodPressure_LinearRegression_Basic_Ext...
 - 2 bloodPressure_MultiVariate_WithVectors.py
 - mlr02.xlsx
 - bloodPressure_LinearRegression_Basic_ExtendedF
 - bloodPressure_LinearRegression_Basic.py
 - bloodPressure_LinearRegression_ConvergenceTes
 - bloodPressure_LinearRegression.py

Variable explorer File explorer Help

IPython console

Console 1/A

In [9]:
In [9]:
In [9]:
In [9]:
In [9]:
In [9]:
In [9]:
In [9]:
In [10]:

Python console History log IPython console

Run file Permissions: RW End-of-line: CRLF

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Ted.Scully\Desktop\ML Class Material

Editor

FirstProgram.py

```
1 # -*- coding: utf-8 -*-
2
3 """
4
5 @author: Ted.Scully
6 """
7
8 print ("Hello there")
9
```

File explorer

Name	Size
Final Code Files	
E_COM_Project_Logistic_Regression	
1_bloodPressure_LinearRegression_Basic.py	
2_bloodPressure_LinearRegression_Basic_ExtendedForMultiVar.py	
2_bloodPressure_MultiVariate_WithVectors.py	
mlr02.xlsx	
bloodPressure_LinearRegression_Basic_ExtendedForMultiVar.py	
bloodPressure_LinearRegression_Basic.py	
bloodPressure_LinearRegression_ConvergenceTest.py	
bloodPressure_LinearRegression.py	

Variable explorer File explorer Help

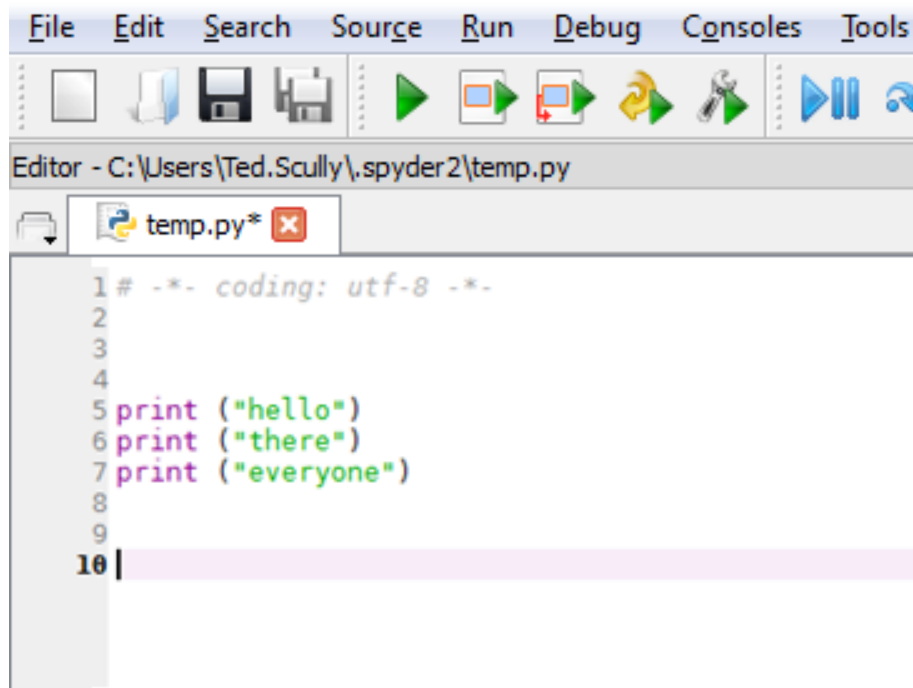
IPython console

Console 1/A

```
In [9]:
In [9]:
In [9]:
In [9]:
In [9]:
In [9]:
In [10]: runfile('C:/Users/Ted.Scully/Desktop/ML Class Mat
wdir='C:/Users/Ted.Scully/Desktop/ML Class Material')
Hello there
In [11]:
```

Multiple Lines of Code and Sequence of Execution

- We can place multiple lines of code into our program.
- When we run the program the interpreter starts at the top of the file and executes statements from top to bottom.



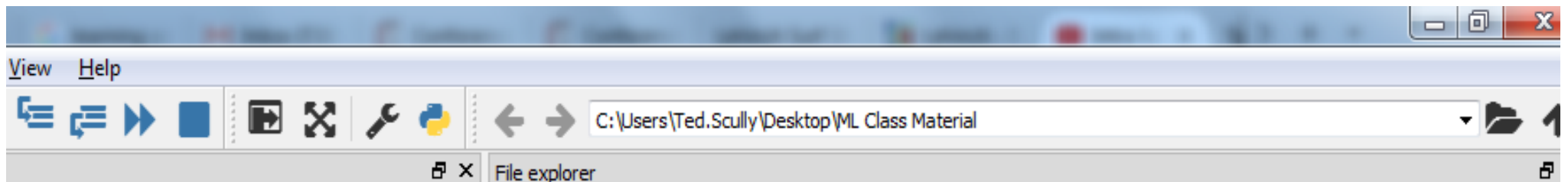
The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, and Tools. Below the menu is a toolbar with icons for file operations and execution. The editor window title is "Editor - C:\Users\Ted.Scully\spyder2\temp.py". The file explorer shows "temp.py*" as the active file. The code editor contains the following Python code:

```
1 # -*- coding: utf-8 -*-  
2  
3  
4  
5 print ("hello")  
6 print ("there")  
7 print ("everyone")  
8  
9  
10 |
```

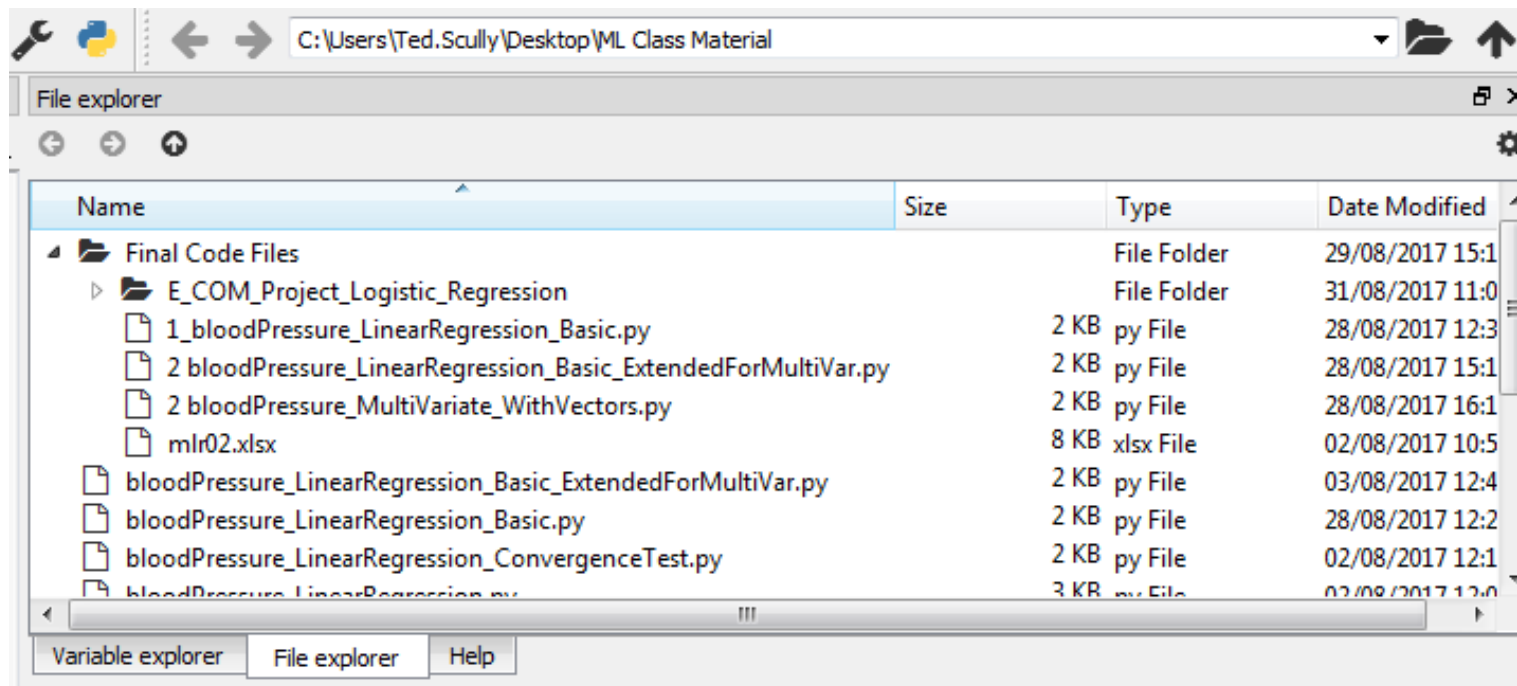
```
>>>  
hello  
there  
everyone
```

Working Directory

- The current working directory is specified in the top right hand corner of the IDE.
- The working directory simply refers to the directory in which the IDE is currently working.
- So for example, if you want to open a file, the first place the IDE will look for this file is in the current working directory.

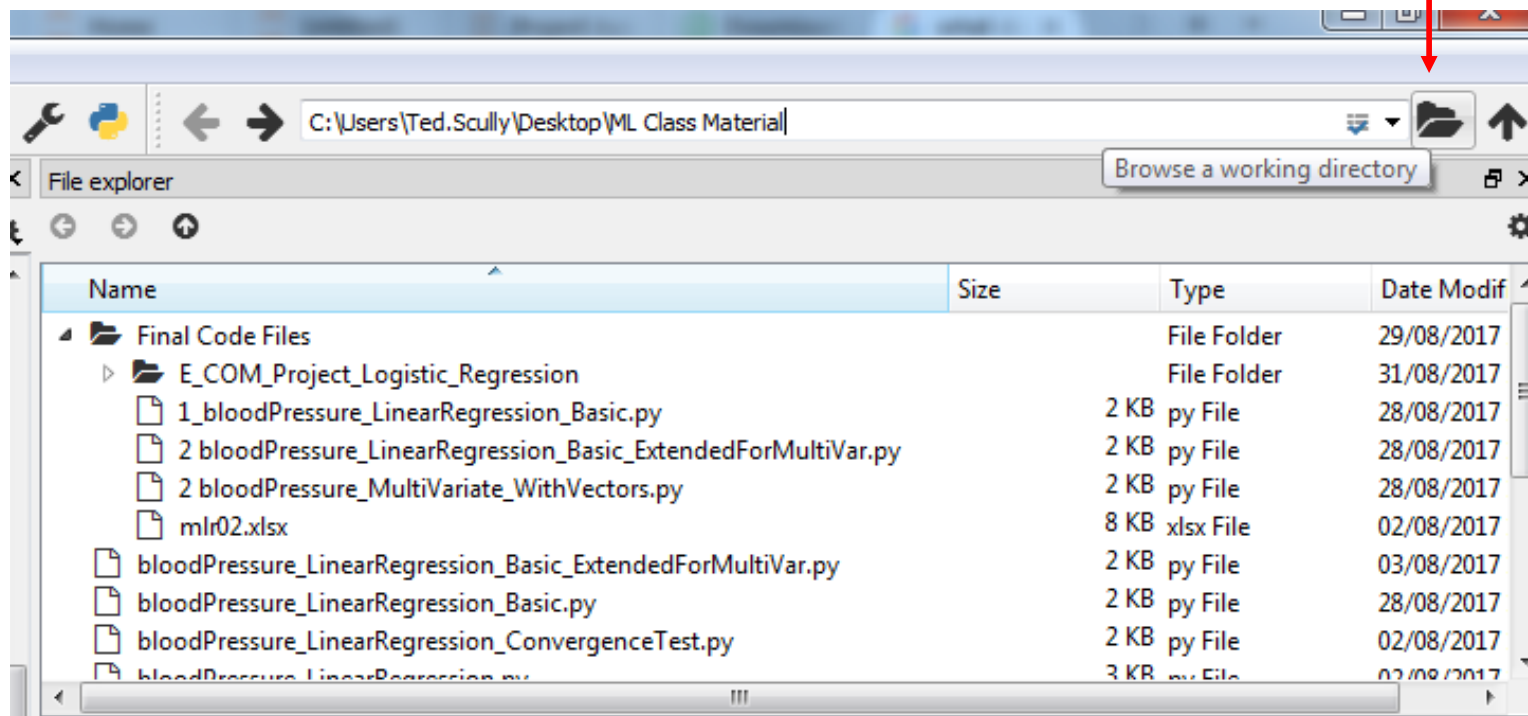


- You will also notice that you can select the file explorer tab (underneath the current working directory).
- There is a useful tab that shows you all the files in the current working directory.
- If you double click on any of these files it will open the file in the IDE.



Working Directory

- You can easily change the current working directory by clicking the folder symbol below.



Some Useful Shortcuts

- ▶ F5 executes the current file
- ▶ F9 executes the currently highlighted code: If nothing is selected F9 executes the current line.
- ▶ Auto-completes commands are very useful.
 - ▶ Tab enables auto complete for function names and variable names in the Console (both Python and IPython) and in the Editor.

Overview

- Overview of Python and its runtime environment



- Input/Output and Processing in Python
- Data Types
- Operators

Input, Processing, and Output

- Typically, computer programs perform a three-step process
 - Receive input
 - Input: any data that the program receives while it is running
 - Perform some process on the input
 - Example: mathematical calculation
 - Produce output

Displaying Output with the `print` Function

- As we have already seen one of the most fundamental actions of a program is to display a message on the computer screen.

- `print ("Hello there")`

- Simply an instruction to the interpreter to print the **String** "Hello there"

There')

Note: The following statements are not equivalent and cause the interpreter to take different actions.

`print (hello)`
`print ('hello')`

Comments

- Comments: notes of explanation within a program
 - Ignored by Python interpreter
 - Intended for a person reading the program's code
- There are two ways of commenting your code in Python:
 - In Python a comment begins with a # character. Everything after the # on the same line is ignored by the interpreter
 - Comments spanning more than one line are achieved by inserting a multi-line string (with """ as the delimiter one each end)

Comments

```
# This program displays a person's  
# Christian name and surname.  
print ('Ted')  
print ("Scully")          # This line prints the surname
```

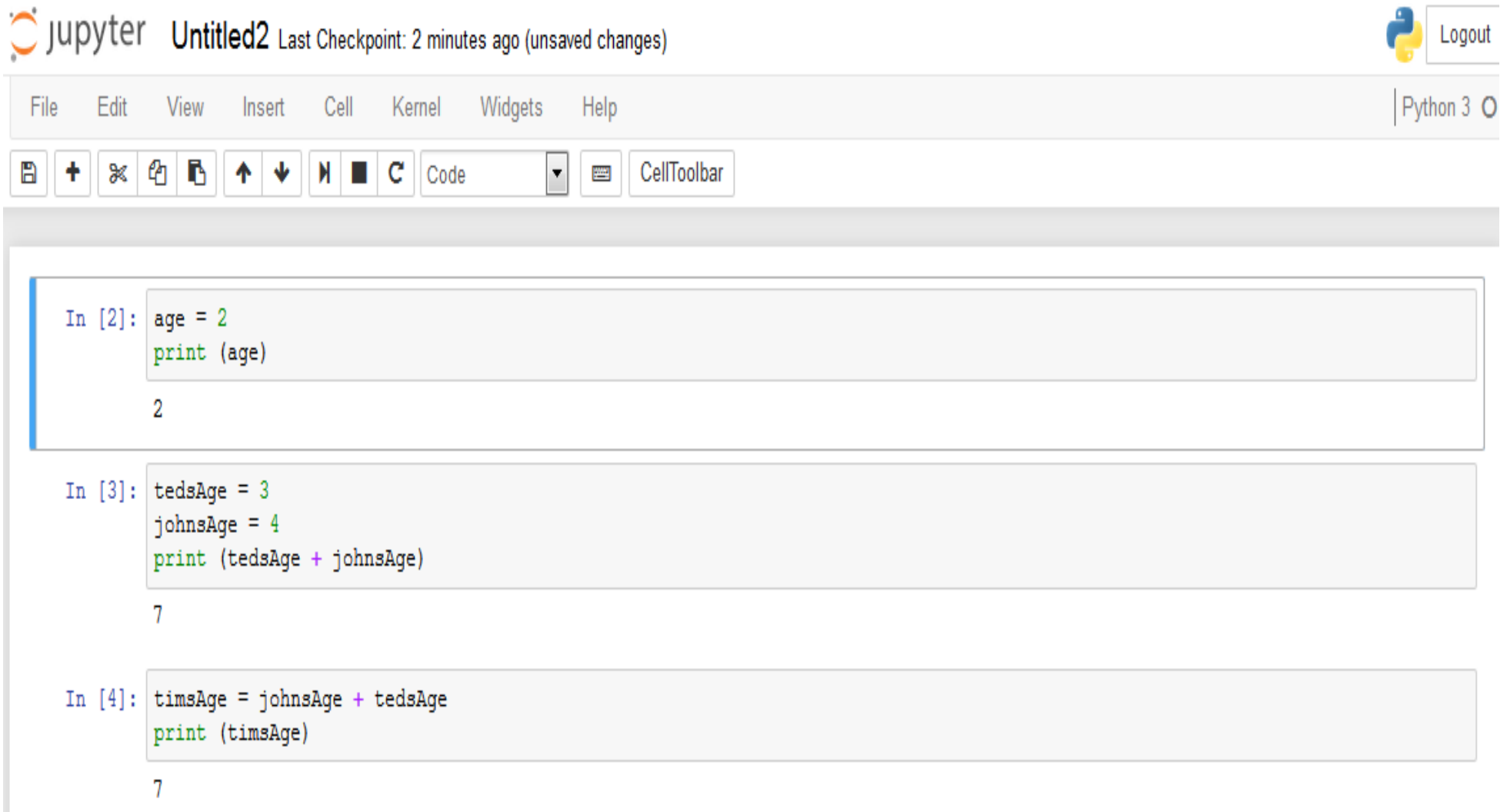
```
"""  
  
This purpose of the code below is  
to .....  
  
"""  
  
print ("hello")  
print ("there")
```


Variables

- **Variable**: name that represents a value stored in the computer memory
 - Used to access and manipulate data stored in memory
 - Labelled box analogy
- Assignment of a value to a variable: used to create a variable and make it reference data
 - General format is ***variable = expression***
 - Example: age = 29

Variables

- A variable references the value it represents



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is on the left, followed by the text "jupyter Untitled2" and "Last Checkpoint: 2 minutes ago (unsaved changes)". On the right, there is a Python logo and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar is "Python 3" with a dropdown arrow. Below the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and a "Code" dropdown menu. To the right of the toolbar is a "CellToolbar" button. The main area contains three code cells. The first cell, labeled "In [2]:", contains the code `age = 2` and `print (age)`, with the output `2` displayed below. The second cell, labeled "In [3]:", contains the code `tedsAge = 3`, `johnsAge = 4`, and `print (tedsAge + johnsAge)`, with the output `7` displayed below. The third cell, labeled "In [4]:", contains the code `timsAge = johnsAge + tedsAge` and `print (timsAge)`, with the output `7` displayed below.

jupyter Untitled2 Last Checkpoint: 2 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

```
In [2]: age = 2
        print (age)
2
```

```
In [3]: tedsAge = 3
        johnsAge = 4
        print (tedsAge + johnsAge)
7
```

```
In [4]: timsAge = johnsAge + tedsAge
        print (timsAge)
7
```

Variable Naming Rules

- Rules for naming variables in Python:
 - Variable name cannot be a **Python keywords**
 - Variable name cannot contain **spaces**
 - First character must be a **letter** (or an underscore)
 - Variables names are **case sensitive**
- camelCase naming convention
 - The variable name begins with lowercase letters.
 - The first character of the second and subsequent words is written in uppercase.
- Variable name should reflect its use
 - User bankBalance as opposed to b or bba

Displaying Multiple Items with the `print` Function

- Python allows you to display multiple items with a single call to `print`
 - Items are separated by commas when passed as arguments
 - Arguments displayed in the order they are passed to the function
 - Items are automatically separated by a space when displayed on screen

```
In [7]: age = 105
        name = "ted"
        print ("name is ", name, " age is ", age)

        name is ted age is 105
```

Printing on the same line

- By default the execution of a print command will move the cursor on to the next line
- It can sometimes be useful to suspend this behaviour so that the cursor stays on the same line.
- In python you can override the newline behaviour and specify directly what you wish to happen after the print statement.
- You can do this using the *end* argument.

Printing on the same line

- By default the execution of a print command will move the cursor on to the next line
- It can sometimes be useful to suspend this behaviour so that the cursor stays on the same line.
- In python you can override the newline behaviour and specify directly what you wish to happen after the print statement.
- You can do this using the *end* argument.

Printing on the same line

- By default the execution of a print command will move the cursor on to the next line
- It can sometimes be useful to suspend this behaviour so that the cursor stays on the same line.
- You can do this by placing a comma at the end of the print line

```
print ("Welcome to", end= " ")  
print ("Analytical Programming")
```

Welcome to Analytical Programming

Error Messages in Python

- When you execute a Python program the interpreter takes each line in turn, turns it into machine code, which is then executed.
- If the interpreter encounters an error it will **output an error message**.
- It is important to be able to interpret these errors as they will help you

```
number = 5  
print (Number + 3)
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-10-9a7e20361ef9> in <module>()  
      1 number = 5  
----> 2 print (Number + 3)  
  
NameError: name 'Number' is not defined
```


Sample Error Messages

- You cannot use a variable until you have assigned a value to it
- Clearly the problem on the previous slide is that I'm trying to print the variable Number but I haven't assigned it a value
 - Always look for the line number
 - Gives a basic description of error
 - Takes practice in order to identify the source

Sample Error Messages

Some errors can be difficult to identify and this process takes practice

```
num = 4
print ("The value of num is " num)
```

```
File "<ipython-input-11-a8b2df75ee99>", line 2
    print ("The value of num is " num)
                                   ^
```

```
SyntaxError: invalid syntax
```

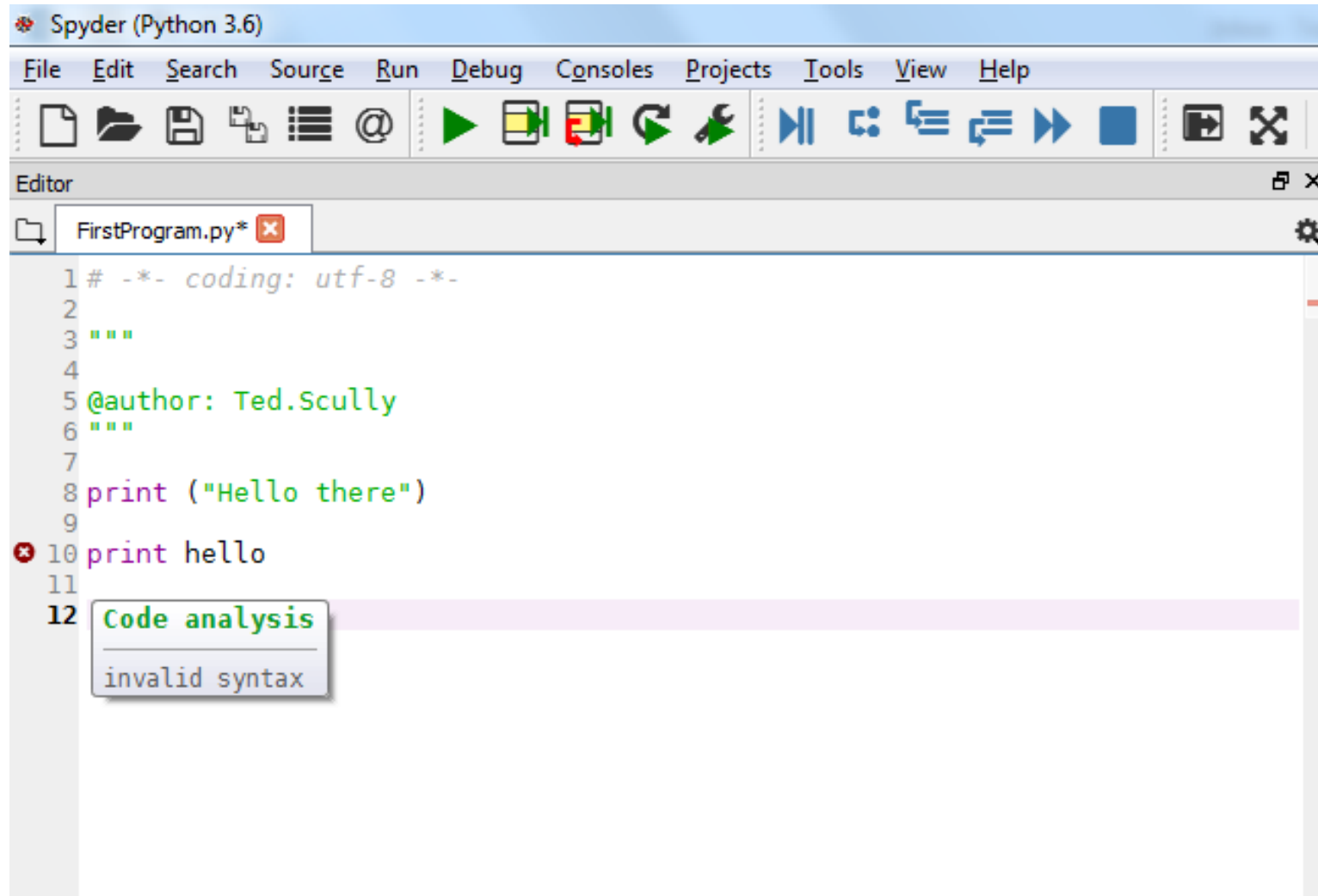
Sample Error Messages

Some errors can be difficult to identify and this process takes practice

```
num = 4  
print ("The value of num is ", num)
```

```
The value of num is 4
```

- The IDE will provide some basic error checking and will highlight lines that contains errors.



Overview

- Overview of Python and its runtime environment



- Input/Out and Processing in Python
- Data Types
- Operators

Data Types


- In your program you may need to use different types of data such as integer numbers, strings, real numbers etc.
 - Some of the code you will use only variables of a particular type
- Python reserves space in memory to store a particular value. However, based on the type of the variable the space allocated can vary.
- So how does Python decide the type of a variable?
 - Python is not a strongly typed language

Data Types

- When the Python interpreter creates a variable it reads the value being assigned to the variable from the program code.
- It then determines the data type of the variable based on this value.
- If I create a numeric variable it determines its data type according to the following rules:
 - A numeric literal that is written as a whole number with no decimal point is considered an int. Examples are 7, 12 4, and - 9.

Data Types (int and float)

- So, the following statement causes the number 503 to be stored in memory as an int
 - ***room = 503***
- The following statement causes the number 2.75 to be stored in memory as a float:
 - ***dollars = 2.75***
- Python also has a data type named str, which is used for storing strings in memory.
- The following will store the String John to be stored in memory as a String
 - ***firstName = 'John'***
 - ***secondName = "Tim"***



When you store an item in memory, it is important for you to be aware of the item's data type. As you will see, some operations behave differently depending on the type of data involved, and some operations can only be performed on values of a specific data type.

Data Types

- Once you create a variable, that variable is not permanently tied to a specific data type.
- The same Python variable can be assigned to different variables

```
# A variable is not permanently tied to a specific data type.
```

```
timsAge = "five"  
print (timsAge)
```

```
timsAge = 5  
print (timsAge)
```

```
timsAge = 5.5  
print (timsAge)
```

```
five
```

```
5
```

```
5.5
```

Boolean Data Type

- Python also provides a boolean datatype, that is a variable that is either True or False (True and False are both **keywords** in Python)
 - isOverAge = True
 - enrolledInModule = False

Checking a Variable Datatype

- If you are unsure of the data type of a variable you can use a in-built function called *type()*.

```
number = 15  
name = "scully"  
  
print (type(number))  
print (type(name))
```

```
<class 'int'>  
<class 'str'>
```

Reading User Input from the Keyboard

- Most programs need to read input from the user
- Python's input function is useful for reading input from the keyboard.
- It reads input as a String
- Format: ***variable = input(prompt)***
 - `prompt` is typically a string instructing user to enter a value

```
name = input("What is your name?")  
print (name)  
print ( type(name))
```

```
What is your name?Ted  
Ted  
<class 'str'>
```

Reading String Input

- The problem with the input function is that we may not always want to read string variables in from the user.
- Have a look at the following code and see if you can determine the source of the problem.

```
age = input("What is your age")  
print (age+2)
```

```
What is your age 13
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-17-38672b2b662f> in <module>()  
      1 age = input("What is your age")  
----> 2 print (age+2)  
  
TypeError: must be str, not int
```

Reading String Input

- The problem is the code below is that we have read in the user input as a string value.
- The variable age stores the string value "13".
- We then attempt to add the numerical value 2 to the string value "13". You cannot add a string and an int.

```
age = input("What is your age")  
print (age+2)
```

```
What is your age 13
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-17-38672b2b662f> in <module>()  
      1 age = input("What is your age")  
----> 2 print (age+2)
```

```
TypeError: must be str, not int
```

Reading String Input

- Python has a number of methods that allows us to force the conversation from one data type to another. This process is also called casting.
- It provides the following:
 - `int()`, which converts a value to an int (for example, it can convert a string to an int)
 - `flt()`, converts a value to a float (for example, it can convert a int to a float)
 - `str()`, converts to a string data type
- For each of the methods above you put the value you want to convert between the open and close brackets.
- So lets try to fix the problem on the previous slide.

Reading String Input

- Notice we pass the String variable age to the int function and it returns a new int value for age.

```
# Notice we convert age from being a string variable to an int variable using the int function.  
age = input("What is your age")  
age = int(age)  
print (age+2)
```

What is your age 13

15

Data Conversion

- Notice below we initially create a variable called number that has a float data type.
- We then use the int function to convert it from a float to an int data type.

```
number = 15.456  
number = int(number)  
print (number)  
print (type(number))
```

15

<class 'int'>

Data Conversion

- In the example below we create a int variable called number and cast (force it's conversion) to a float data type)

```
number = 15  
number = float(number)  
print (number)  
print (type(number))
```

15.0

<class 'float'>

Programming Task

- Write a simple program that will ask for the user for two numbers and add them together. It should then print out the result.

```
# Programming Task - Adding two numbers together.  
num1 = input("Please enter first number")  
num2 = input("Please enter second number")  
  
print ("Result is ", int(num1)+ int(num2))
```

```
Please enter first number 14  
Please enter second number 13  
Result is 27
```

Programming Task

- So now lets implement the same program in Sypder.

The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script named `FirstProgram.py` with the following code:

```
1 # -*- coding: utf-8 -*-
2
3 """
4
5 @author: Ted.Scully
6 """
7
8 # Programming Task - Adding two numbers together.
9 num1 = input("Please enter first number")
10 num2 = input("Please enter second number")
11
12 print ("Result is ", int(num1)+ int(num2))
```

The right-hand side of the IDE contains three panels. The **File explorer** panel shows the project structure, including a folder named `Final Code Files` and a sub-folder `E_COM_Project_Logistic_Regression` containing several Python files. The **Variable explorer** panel is empty. The **IPython console** panel shows the execution of the script, with the following output:

```
In [16]:
In [16]:
In [16]:
In [16]:

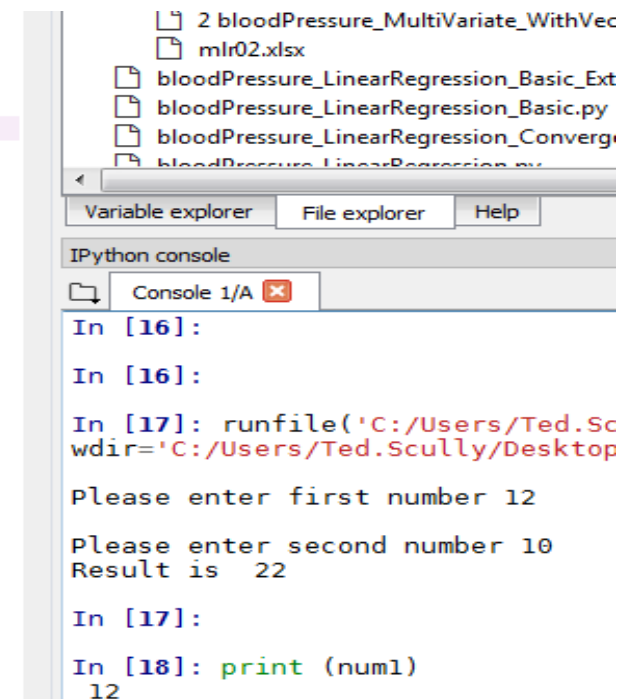
In [17]: runfile('C:/Users/Ted.Scully/Desktop/ML cl
wdir='C:/Users/Ted.Scully/Desktop/ML Class Material

Please enter first number 12

Please enter second number 10
Result is  22
```

- When we run a program using the console (it is an Ipython console) the variable are loaded into the console when the code is run. We can then access these variables from the console by printing them directly (see below).

```
8 # Programming Task - Adding two numbers together.  
9 num1 = input("Please enter first number")  
10 num2 = input("Please enter second number")  
11  
12 print ("Result is ", int(num1)+ int(num2))
```



Revision Overview

- Overview of Python and its runtime environment
- Input/Out and Processing in Python



Data Types

- Operators
- Decision Structures
- Iteration

Performing Calculations

- Math expression: performs calculation and returns a value

Symbol	Operation	Description
+	Addition	Adds two numbers
-	Subtraction	Subtracts one number from another
*	Multiplication	Multiplies one number by another
/	Division	Divides one number by another and gives the quotient
%	Remainder	Divides one number by another and gives the remainder
**	Exponent	Raises a number to a power

- Exponent operator (**): Raises a number to a power

- $x ** y = xy$

- Remainder operator (%): Performs division and returns the remainder

- a.k.a. modulus operator

- e.g., $4\%2=0$, $5\%2=1$

- Typically used to convert times and distances, and to detect odd or even numbers

Mixed-Type Expressions

- Data types resulting from math operations depends on the data types of operands
 - Two `int` values: result is an `int`
 - There is an exception to the above rule. If dividing an `int` by an `int` the result is a `float`. This is a change introduced in Python 3
 - Two `float` values: result is a `float`
 - `int` and `float`: `int` temporarily converted to `float`, result of the operation is a `float`
 - Mixed-type expression

Example of Precedence

```
>>> 12.0 + 6.0 / 3.0
```

```
>>> 1+2*2%3
```

Operator Precedence and Grouping with Parentheses

Python operator precedence:

- Operations enclosed in parentheses

 - Forces operations to be performed before others

- Exponent (**)

- Multiplication (*), division (/), and remainder (%)

- Addition (+) and subtraction (-)

Higher precedence performed first

- Same precedence operators execute from left to right

Example of Precedence

- Division
- Sum

```
>>> 12.0 + 6.0 / 3.0
```

```
>>> 1+2*2%3
```

- Multiplication
- Modulus
- Sum

Example of Precedence

Use of Parenthesis

Typically parts of a mathematical expression may be grouped with parentheses to force some operations to be performed before others.

```
>>> (12.0 + 6.0) / 3.0
```

```
>>> (1+2)*2%3
```

1. Sum
 2. Division
- Result is 6.0

1. Sum
 2. Multiplication
 3. Modulus
- Result is 0

Task

- . Write a program that will ask a user
 - how many books do you want to read
 - how many months will it take you to read them
- . The program should then tell the user how many books they will read per month

```
# Get the number of books the user plans to read.
books = int(input("How many books do you want to read? "))

# Get the number of months it will take to read them.
months = int(input("How many months will it take? " ))

# Calculate the number of books per month.
perMonth = books / months

# Display the result.
print ("You will read", perMonth, "books per month.")
```

```
How many books do you want to read? 19
How many months will it take? 4
You will read 4.75 books per month.
```

More About Data Output (cont'd.)

- Special characters appearing in string literal
 - Preceded by backslash (\)
 - Examples: newline (\n), horizontal tab (\t)
 - Treated as commands embedded in string
- When + operator used on two strings it performs string concatenation
 - Useful for breaking up a long string literal

```
name = input('Type your full name')
rest = " \n goes to college at \n CIT "
print (name+rest)
```

```
Type your full name Ted Scully
Ted Scully
goes to college at
CIT
```

Discussion



Thank you