MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Python Programming

SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY

Python Programming

Course Objectives:

- Knowledge
  - To recognize the core syntax and semantics of Python programming language.
  - To learn the process of structuring the data using lists, dictionaries, tuples.
- Skills
  - To infer the Object-oriented Programming concepts & Exception Handling in Python.
  - To learn advanced applications such as Multithreading & Regular Expressions.
- Attitude
  - To identify the commonly used operations involving file systems and
  - To articulate proficiency in the Data analysis and GUI Programming.

Python Programming

Course Outcomes:

After completion of the course the students will be able to :-

- To apply the fundamental syntax and semantics of Python Programming
- To demonstrate the Python data structures–lists, tuples, dictionaries.
- To demonstrate the concepts of object oriented Concept, Exception Handling etc.
- To apply the features of python programming languages to solved the real world problem

Syllabus

1. Introduction to Python
2. Function and Input / Output in Python
3. Introduction to String, List, Tuple and Dictionary
4. Python Object Oriented and Exception Handling
5. Python Regular Expressions
6. Python for Data Analysis and Python GUI (Tkinter)

Laboratory Exercises

1. Introduction to Basic Python Commands.

2. Write a python program to find largest of three numbers

3. Write a python program that accepts the length of three sides of a triangle as inputs. The program should indicate whether or not the triangle is a right-angled triangle using function with exception handling.

4. Write a python program to create, append and remove etc. operation on list.

5. Write a python program to create, append and remove etc. operation on Dictionary and Tuple

6. Write a program to read 3 subject marks and display pass or failed using class and object

7. Write a python code to read a csv file using panda’s module and print the first and last five records of the file. Using Matplotlib shows data analysis.

8. using a numpy module create an array and check the following: 1. Type of array 2. Axes of array 3. Shape of array4. Type of elements in array.

9. Create a Regular Expression and implement the following a) Recognize the following strings: “bat,” “bit,” “but,” “hat,” “hit,” or “hut.” b) Match any pair of words separated by a single space, i.e., first and last names. c) Match any word and single letter separated by a comma and single space, as in last name, first initial.

10. Write a python program to implement multithreading scenarios.

11. Create a program to take input of your date of birth and output your age.

Learning Resources

Reference Books:

1. Let us Python, Yashavant Kanetkar and Aditya Kanetkar, First Edition, 2019, BPB Publications

2. Learn Python 3 the Hard Way, Zed A. Shaw, First Edition, 2018, Pearson Education Inc.

3. Jake VanderPlas, “Python Data Science Handbook: Essential Tools for Working with Data”, 1st Edition, O'Reilly Media, 2016. ISBN-13: 978-1491912058

4. Gowrishankar S, Veena A, “Introduction to Python Programming”, 1st Edition, CRC Press/Taylor & Francis, 2018. ISBN-13: 978-0815394372

Reference Books:

- **Supplementary Reading:**
- **Web Resources:**

1. https://nptel.ac.in/courses/106/102/106102067/
2. https://nptel.ac.in/courses/106/106/106106182/
3. https://nptel.ac.in/courses/106/106/106106212/

- **Weblinks:**

1. https://www.python.org
2. https://realpython.com/beginners-guide-python-turtle/
3. https://realpython.com/python-gui-tkinter/
4. http://www.codecademy.com/tracks/python
5. http://learnpythonthehardway.org/book/

- **MOOCs:**

1. https://www.coursera.org/learn/python-programming-intro

Assessment Scheme

Laboratory Continuous Assessment (LCA)*:* 50 Marks

Practical : 30 Marks

Oral based on practical : 20 Marks

Introduction to Python

Why Programming ??

- We live in a **digital society** where everyone uses a computer or a mobile phone or most of the times both.

- It's one thing to know **how to use** the **apps/programs on such digital devices** and it's totally another to know how the **logic behind them works**.

- In this digital age the **knowledge of programming is essential in order to bring innovation and change.**

- To create value with your own ideas you need to know how to code.

- **Programming** has become **basic literacy** for the 21st century**.**

Introduction to Python

What is Python??

- Python is a widely used **high-level**, **general-purpose**, **interpreted**, **dynamic programming language.**
- Python is created by **Guido van Rossum is a Dutch programmer in 1990** who is best known as the author of the Python programming language.
- Python3 released in 2008
- Easy to use
- Code readability

Introduction

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You **do not need to compile** your program before executing it.

- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language:** Python is a great language for the beginner level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

About Python Programming

- **Free and open-source -** You can freely use and distribute Python, even for commercial use.

- **Easy to learn -** Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like C++, Java, C#.

- **Portable -** You can move Python programs from one platform to another, and run it without any changes.

Python Features

- Easy to Learn
- Easy to read
- Easy to Maintain
- A broad standard library
- Interactive mode
- Portable
- Extendable
- Databases
- GUI programming
- Scalable

Python Applications

- Web Development. ...
- Game Development. ...
- Scientific and Numeric **Applications**. ...
- Artificial Intelligence and Machine Learning. ...
- Software Development. ...
- Enterprise-level/Business **Applications**. ...
- Education programs and training courses. ...
- Language Development.

MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Why Python?

- Python is generally:
  - Comparatively easy to learn
  - Freely available
  - Cross-platform (Windows, Mac, Linux)
  - Widely used – extensive capabilities, documentation, and support
  - Access to advanced math, statistics, and database functions
  - Integrated into ArcGIS and other applications
  - Simple, interpreted language – no compilation step

Programming: Input Output Processing

1. **take input from a user**
2. Accept an integer, float, character, and string input from a user.
3. Convert the user input to a different data type.
4. Do processing
5. **output in desired format**

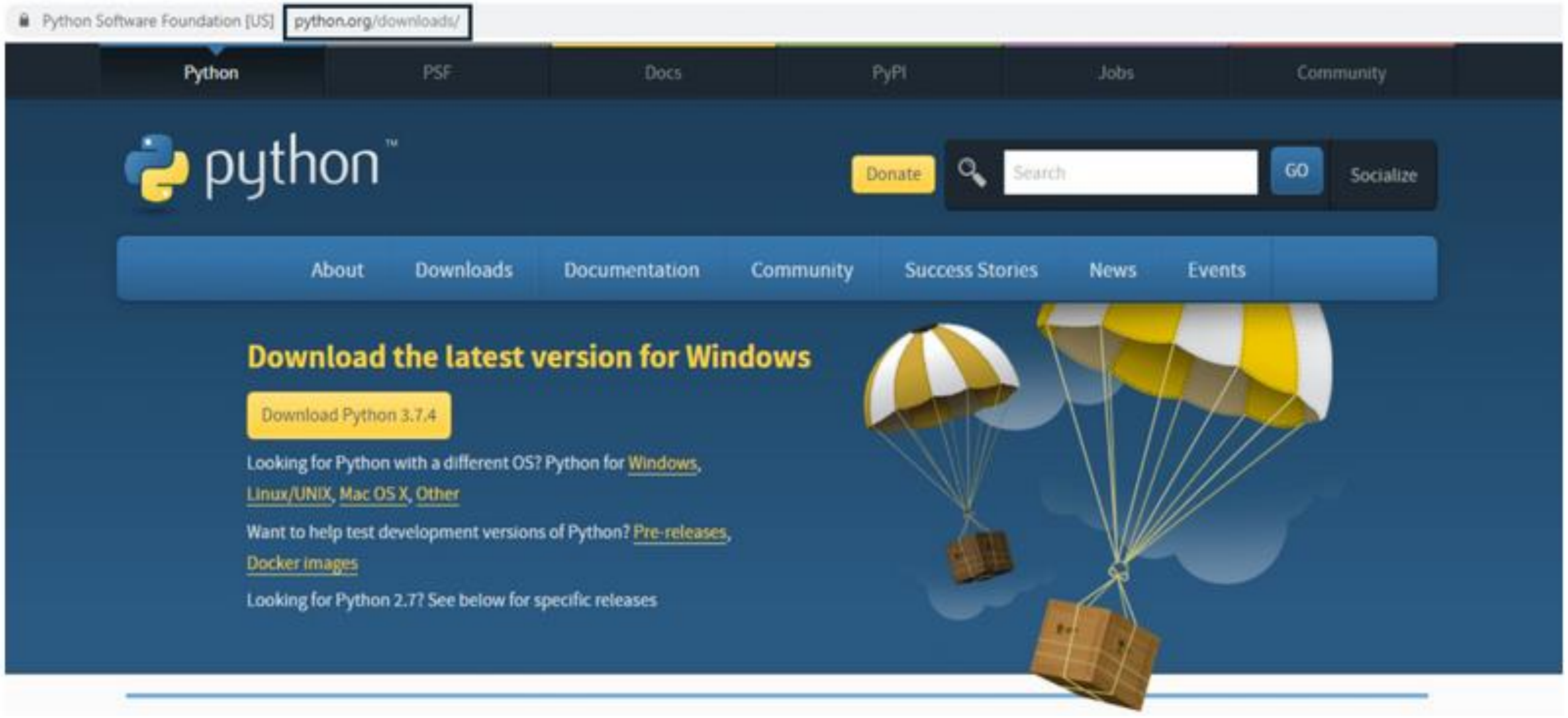
Running Python Scripts

How to run Python scripts by using ??

- The operating system **command-line or terminal**
- The Python interactive mode
- The IDE or text editor you like best

Installing Python

1. Start off by going to this website -> https://www.python.org/downloads/



Download the latest version for Windows

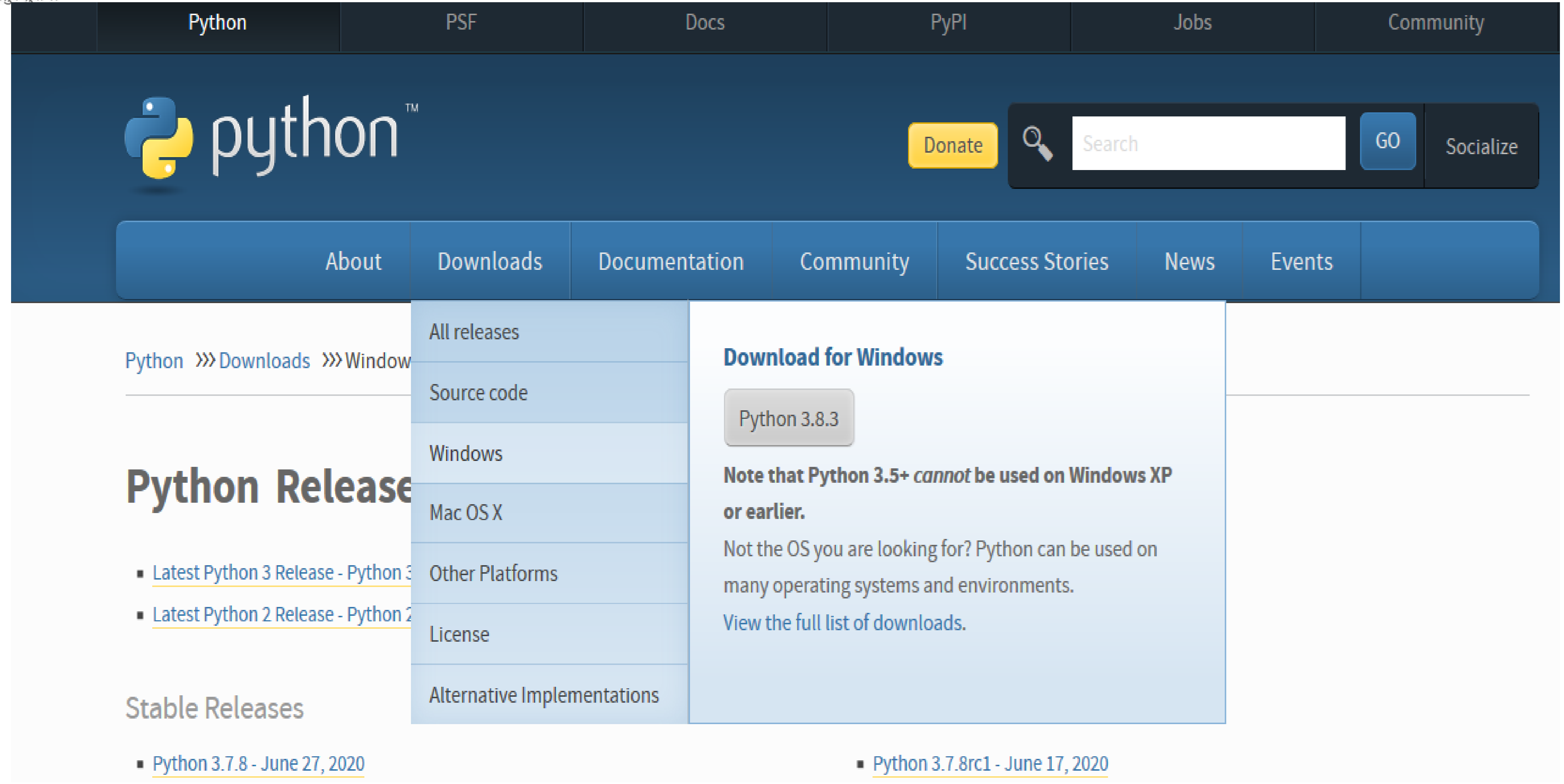
Download Python 3.7.4

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

2. Click on the downloads tab and choose the operating system and python version. So, here I am downloading python version 3.8.4 for Windows operating system



Download for Windows

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

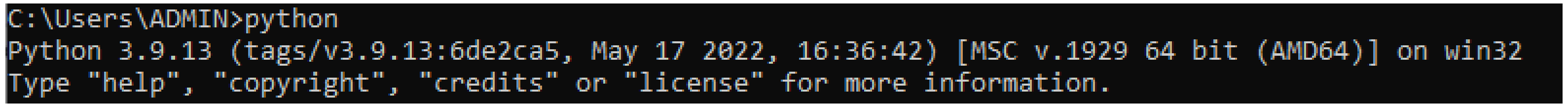
Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

- Download Python 3.8 from https://www.python.org/downloads/release/python-360/ and install it. Be sure to check the box that says to add Python 3.8 to your path.

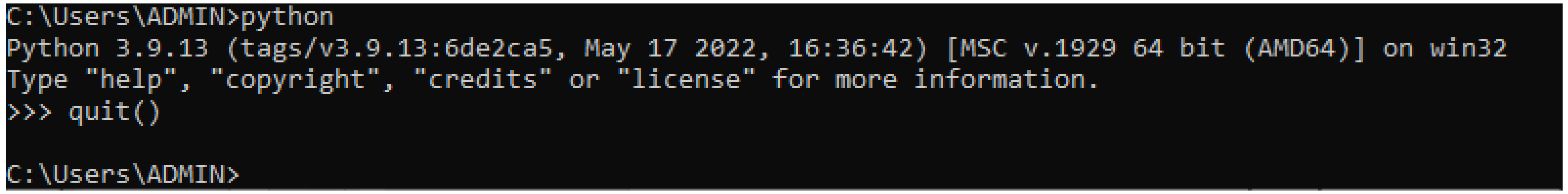
- In your PowerShell (Terminal) /cmd (windows) program, run python.

- You run things in Terminal by just typing the name and pressing Enter.
  (a) If you type python and it does not run, then you have to reinstall Python and make sure
  you check the box for ”Add python to the PATH.”



```
C:\Users\ADMIN>python
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

- Type quit(), and press Enter to exit python.
- You should be back at a prompt similar to what you had before you typed python.



```
C:\Users\ADMIN>python
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()

C:\Users\ADMIN>
```

-V

Basic Commands while working with command line interface

Windows:

1. **cd:**
   To get current working directory
   To move from one directory to another.
   E.g. > cd Desktop

2. **dir :** To see files and folders in a directory

1. **cls :** Used to clear the screen
2. **cd .. :** To move one level up
3. **python :** provides python command line interface
4. **import :** Import in python is **similar to #include header_file in C/C++**. Python modules can get access to code from another module by importing the file/function using import.
5. **To check python version**

import sys
sys. version
8. **quit ()** : to exit from python

-V

Basic Commands while working with command line interface

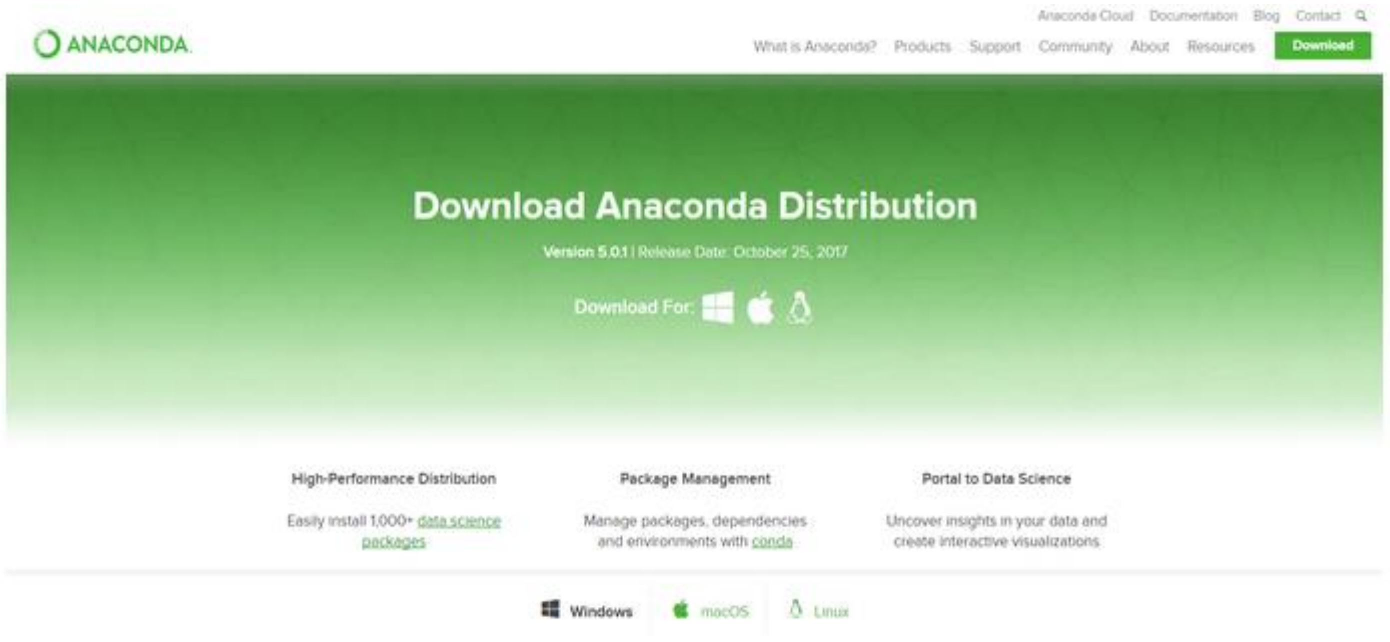
Linux:

1. **pwd :** To get current working directory
2. **ls:** To see files and folders in a directory
3. **cd :** Used to change directory
4. **clear:** Used to clear the screen
5. **cd .. :** Used to move one level up
6. **python :** To get access to python command line interface
7. **python --version , python -V, python -VV:** To check python version
8. **import :** Import in python is **similar to #include header_file in C/C++**. Python modules can get access to code from another module by importing the file/function using import. E.g. import maths
9. **quit() :** to exit from python

Installing IDE: Anaconda

Steps:

1. Visit [Anaconda.com/downloads]()
2. Select Windows
3. Download the ***.exe*** installer
4. Open and run the ***.exe*** installer
5. Open the **Anaconda Prompt** and run some Python code



Download Anaconda Distribution

Version 5.0.1 | Release Date: October 25, 2017

High-Performance Distribution

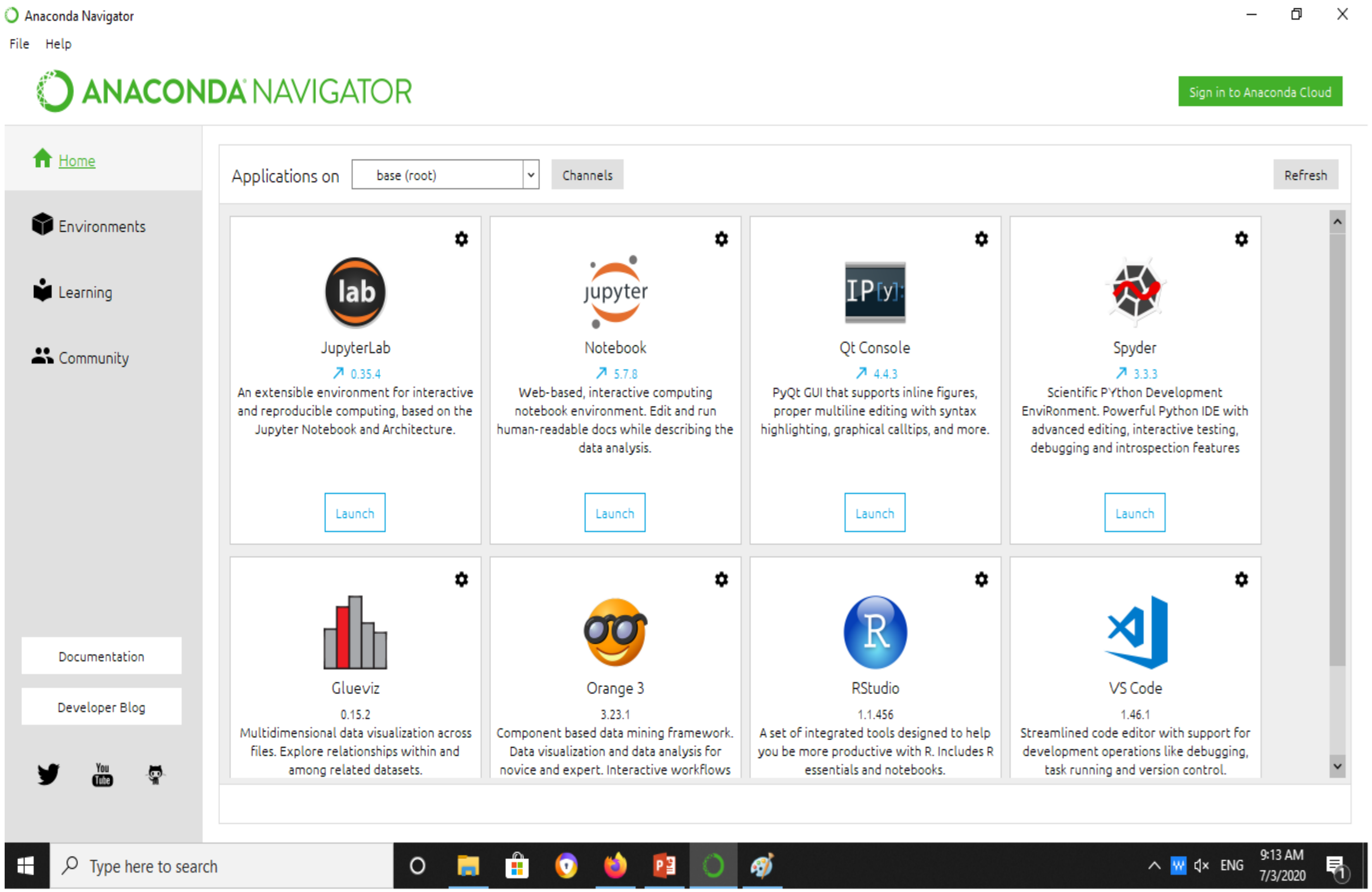
Easily install 1,000+ data science packages

Package Management

Manage packages, dependencies and environments with conda

Portal to Data Science

Uncover insights in your data and create interactive visualizations


Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Documentation

Developer Blog

Applications on base (root) Channels Refresh

JupyterLab
0.35.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
5.7.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Qt Console
4.4.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

Spyder
3.3.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code
1.46.1
Streamlined code editor with support for development operations like debugging, task running and version control.

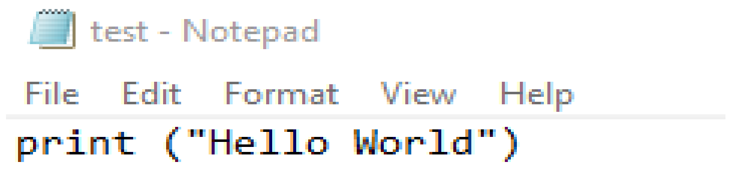
Type here to search

ENG 9:13 AM 7/3/2020

Basic Commands in Python

- **print**
- print ("Hello World")

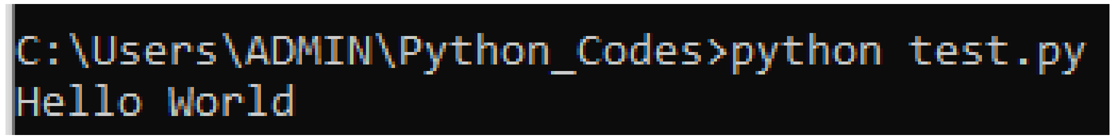
- Open an Editor and type print ("Hello World")



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

- Save file with extension .py for e.g. test.py
- Run python program using terminal and command prompt with the following command

python test.py



```
C:\Users\ADMIN\Python_Codes>python test.py
Hello World
```

Get Hello World displayed on screen

Basic Commands in Python

Exercise1: Write a python program to Print following statements as an output using print statement.

Student Name:
Address:
Contact _No:
Mother Tongue:
School_Name:
Year:
Panel:
Roll_No:

Comments in Python

- Comments in any programming language are used to **increase the readability of the code**
- by reading a comment you **can understand the purpose of code much faster** then by just going through the actual code.

In this Python program we are seeing **three types of comments**. Single line comment, multi-line comment and the comment that is starting in the same line after the code.

Single line comment

```
# This is a single line comment in Python
```

Inline comment:

```
print("Hello World")          # This line prints "Hello World“
```

multi-line comment

```
"""" We are writing a simple program here
First print statement.
This is a multiple line comment""""

print("#this is  a comment ")        ???
```

Comments in Python

Exercise2: In the previous code you written, modify the statements printing following fields into multi-line comments, so these fields will not be the part of the output.

Address:

Contact _No:

Mother Tongue

On command Line

```
>>> print("Hello World")
Hello World
>>> 2+3
5
>>> 5-2
3
>>> 3*5
15
>>> 45/5
9.0
>>> 2**4
16
>>>18%4
2
```

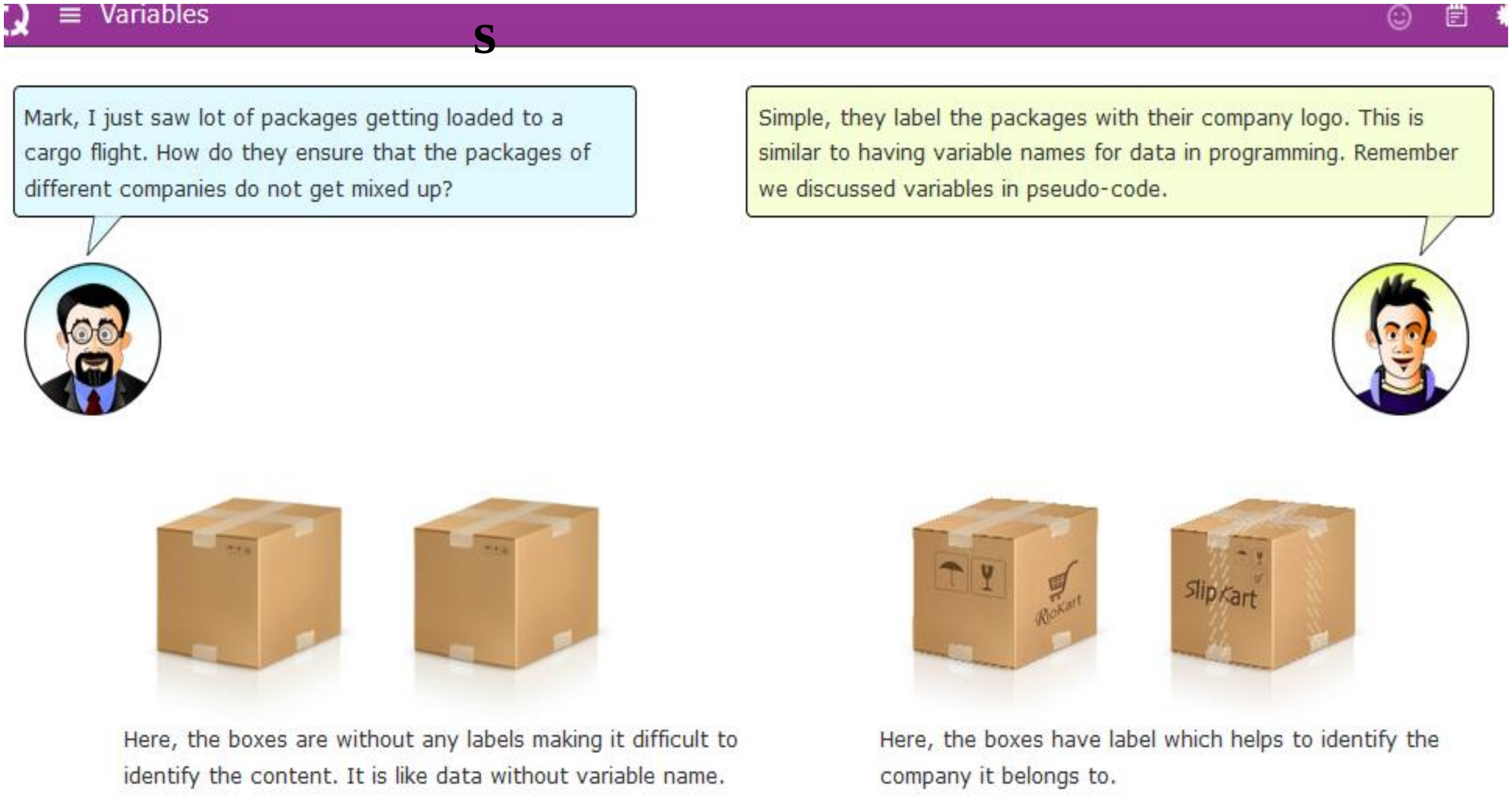
```
>>>5 < 9
True
>>>11>6
True
>>>6<11
False
>>>6<=6
True
>>>11>=5
True

>>> 5// 4                        #floor division


>>> 2 + 10 * 10 + 3              # Order of Operations
followed in Python

>>> (2+10) * (10+3)     # Can use parentheses to specify
orders
```

Variables



Mark, I just saw lot of packages getting loaded to a cargo flight. How do they ensure that the packages of different companies do not get mixed up?

Simple, they label the packages with their company logo. This is similar to having variable names for data in programming. Remember we discussed variables in pseudo-code.

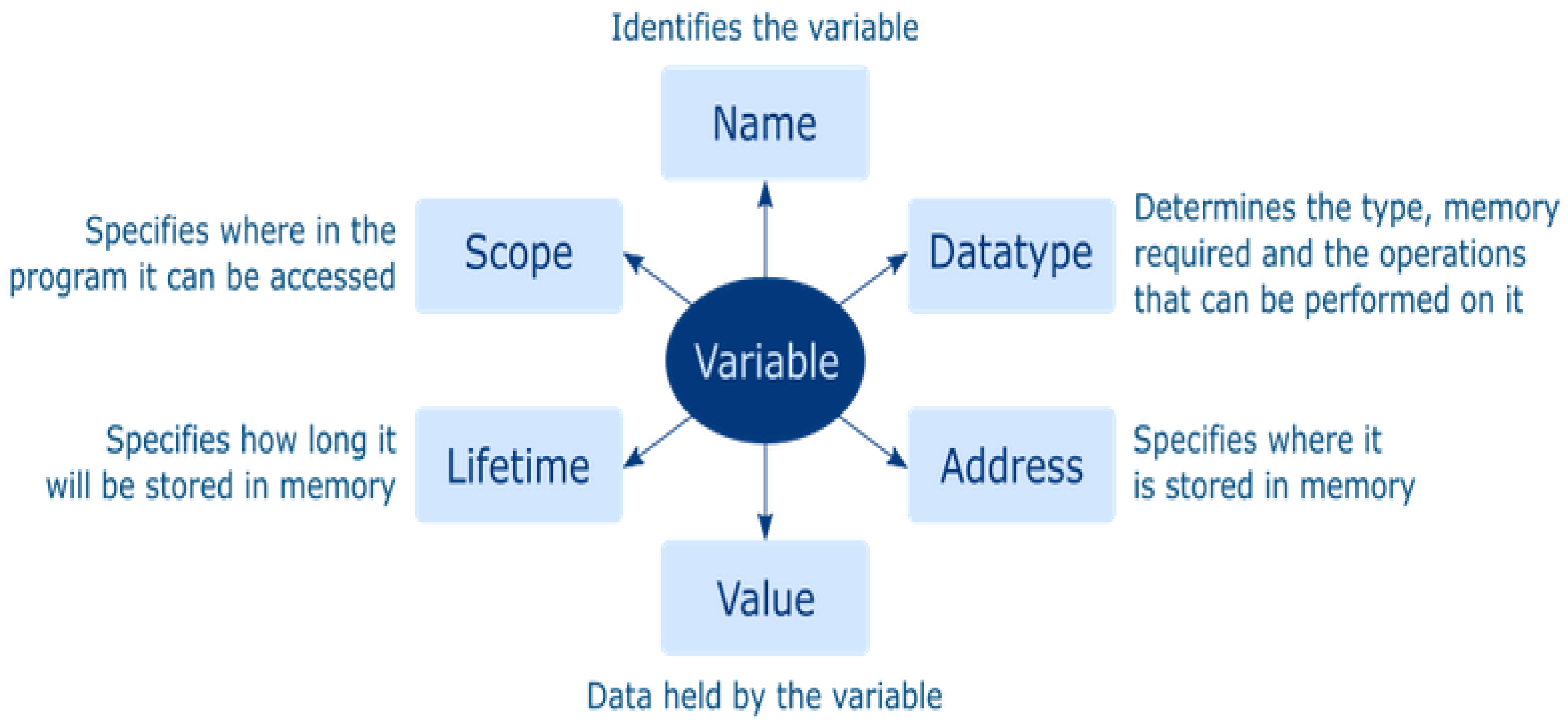
Here, the boxes are without any labels making it difficult to identify the content. It is like data without variable name.

Here, the boxes have label which helps to identify the company it belongs to.

Variables

Variables and its dimensions

We have seen that a variable will have a name, value, type and it will occupy memory. Apart from these, it has two more dimensions – scope and lifetime. Thus we can say that any variable will have the following six dimensions.



Identifies the variable
Name
Specifies where in the program it can be accessed
Scope
Datatype
Determines the type, memory required and the operations that can be performed on it
Variable
Specifies how long it will be stored in memory
Lifetime
Address
Specifies where it is stored in memory
Value
Data held by the variable

Variables

Variables in Python

You can consider a variable to be a temporary storage space where you can keep changing values

we have this cart and initially we store an apple in it.

after some time, we replace this banana with a mango.

So, here this **cart acts like a variable**, where the **values stored in it keep on changing**.

Variables Types

Assigning values to variables

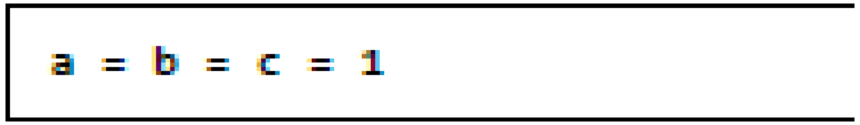
- Python variables do not need explicit declaration to reserve memory space.
- The declaration happens automatically when you assign a value to a variable.
- The equal sign (=) is used to assign values to variables.

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
print (counter)
print (miles)
print (name)
```

Variables Types

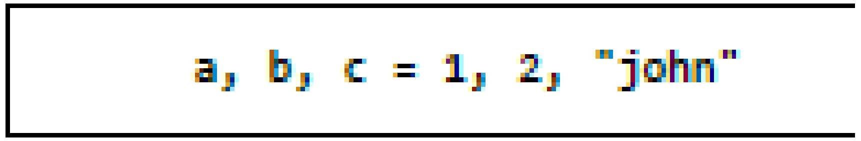
Multiple Assignment

Python allows you to assign a single value to several variables simultaneously



```
a = b = c = 1
```

You can also assign multiple objects to multiple variables



```
a, b, c = 1, 2, "john"
```

0 = x

print(0)

Output: Syntax Error: can't assign to literal

x = True # valid

_y = True # valid

print(x,_y)

Variables
Types

```
9x = False        # starts with numeral => SyntaxError: invalid syntax
$y = False        # starts with symbol => SyntaxError: invalid syntax
```

The remainder of your variable name may consist of letters, numbers and underscores.

e.g.   has_0_in_it = "Still Valid"

Names are case sensitive.

```
x = 9
y = X*5
=>NameError: name 'X' is not defined
```

Variables Types

The Python interpreter automatically picks the most suitable built-in type for it:

```
a = 2
print(type(a))

b = 9223372036854775807
print(type(b))

pi = 3.14
print(type(pi))

c = 'A'
print(type(c))

name = 'John Doe'
print(type(name))

q = True
print(type(q))
```

Exercise 3: Declare two integer and two float variables. Perform following operations on integer and floating point numbers and display the output.

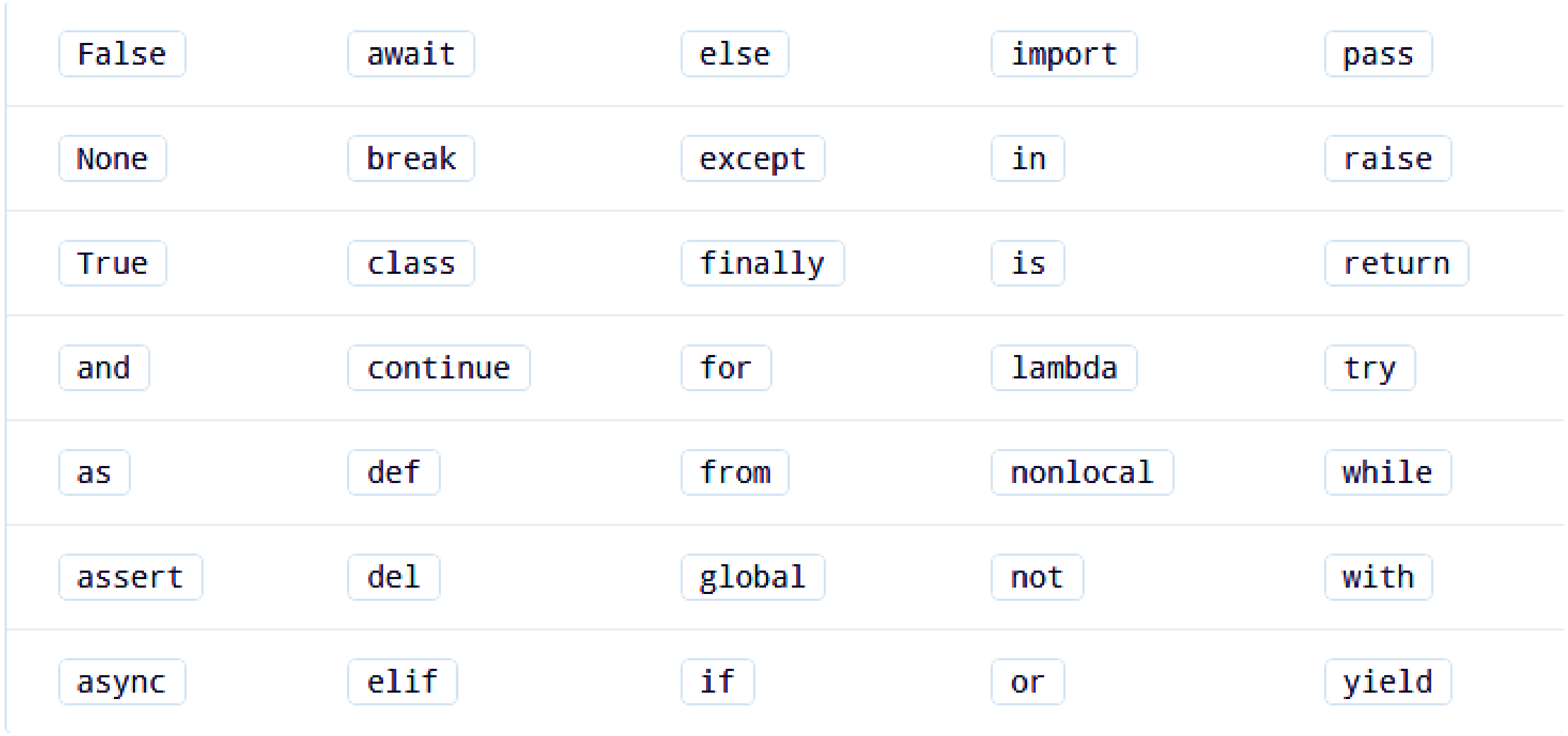
+, -, *,**,/,%, <,>,<=,>=

keywords

- Keywords are the **reserved words** in Python.
- **We cannot use a keyword as a variable name, function name or any other identifier**. They are used to define the syntax and structure of the Python language
- In Python, keywords are **case sensitive**.
- There are 33 keywords in Python 3.7. This number can vary slightly over the course of time.
- All the keywords **except True, False and None are in lowercase** and they must be written as they are

```
import keyword
print(keyword.kwlist)
```

keywords



False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

keywords

```
a=10; True=20;
c=a+True
print(c)

a=10; true=20;
c=a+true
print(c)

and=“ Hello World “
print(and)

AND=“ Hello World “
print(AND)
```

Reading Input from the Keyboard

- **Python** user **input from the keyboard** can be **read** using the **input**() built-in function.
- The **input** from the user is **read** as a string and can be assigned to a variable.
- After entering the value from the **keyboard**, we have to press the “Enter” button.
- Then the **input**() function reads the value entered by the user

```
x=input("Enter Your Name")
print(x)

x = input('Enter your name:')
print('Hello ' + x)

x=int(input("Enter vaule x"))
y=int(input("Enter vaule y"))
z=x+y
print(z)
```

Python Identifiers

An identifier is a **name given to entities** like class, functions, variables, etc.

It helps to differentiate one entity from another.

Rules for writing identifiers

- Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _. Names like myClass, var_1 and print_this_to_screen, all are valid example.
- An identifier cannot start with a digit. 1variable is invalid, but variable1 is a valid name.
- Keywords cannot be used as identifiers.

e.g. global=1

SyntaxError: invalid syntax

- We cannot use special symbols like **!**, **@**, **#**, **$**, **%** etc. in our identifier e.g. a@=0
- An identifier can be of any length.

Python Identifiers

- Python is a case-sensitive language. This means, Variable and variable are not the same.
- Always give the identifiers a name that makes sense.
- While c = 10 is a valid name, writing count = 10 would make more sense, and it would be easier to figure out what it represents when you look at your code after a long gap.
- Multiple words can be separated using an underscore, like this_is_a_long_variable.

Python Literals

Literal is a **raw data given in a variable or constant**. In Python, there are various types of literals

Numeric Literals

- Numeric Literals are immutable (unchangeable).
- Numeric literals can belong to 3 different numerical types: Integer, Float, and Complex.

```
 a = 0b1010 #Binary Literals
b = 100 #Decimal Literal
c = 0o310 #Octal Literal
d = 0x12c #Hexadecimal Literal
#Float Literal
float_1 = 10.5
float_2 = 1.5e2
```

```
#Complex Literal
x = 3.14j

print(a, b, c, d)
print(float_1, float_2)
print(x, x.imag, x.real)
```

Python Literals

Literal is a **raw data given in a variable or constant**. In Python, there are various types of literals

Numeric Literals

- Numeric Literals are immutable (unchangeable).
- Numeric literals can belong to 3 different numerical types: Integer, Float, and Complex.

```
 a = 0b1010 #Binary Literals
b = 100 #Decimal Literal
c = 0o310 #Octal Literal
d = 0x12c #Hexadecimal Literal
#Float Literal
float_1 = 10.5
float_2 = 1.5e2
```

```
#Complex Literal
x = 3.14j
print(a, b, c, d)
print(float_1, float_2)
print(x, x.imag, x.real)
# When we print the variables, all the literals are converted
into decimal values.
```

Python Literals

Boolean literals

A Boolean literal can have any of the two values:
True or False

```
x = (1 == True)
y = (1 == False)
a = True + 4
b = False + 10

print("x is", x)
print("y is", y)
print("a:", a)
print("b:", b)
```

Special literals
Python contains one special literal i.e. **None.**
We use it to specify that the field has not been created

```
drink = "Available"
food = None

def menu(x):
    if x == drink:
        print(drink)
    else:
        print(food)

menu(drink)
menu(food)
```

Operators

Common Operators	Python
Arithmetic Operators	+,-,*,/, %,//
Relational Operators	==,!=,>,<,>=,<=
Assignment Operators	=,+=,-=,*=,/=,%=
Logical Operators	and,or,not

What do you think is the output of 5+4*9%(3+1)/6-1?
How do you think the result of this expression is computed?

It is done based on the precedence of the operator. Precedence of an operator can be identified based on the rule - BODMAS. Brackets followed by Orders (Powers, Roots), followed by modulo, Division and Multiplication, followed by Addition and Subtraction.

1. Brackets have the highest precedence followed by orders.
2. Modulo, Division and Multiplication have the same precedence. Hence if all appears in an expression, it is evaluated from Left to Right.
3. Addition and Subtraction have the same precedence. Hence if both appears in an expression, it is evaluated from Left to Right.

```
5 + 4 * 9 % (3 + 1) / 6 - 1
     5 + 4 * 9 % 4 / 6 - 1
       5 + 36 % 4 / 6 - 1
           5 + 0 / 6 - 1
             5 + 0 - 1
               5 - 1
                 4
```

Arithmetic Operators

Arithmetic Operator	Operator Name	Description	Example
+	Addition	Performs addition	I=40, J=20 >>>I+ J >>>60
–	Subtraction	Performs subtraction	I=40, J=20 >>>I – J >>>20
*	Multiplication	Performs multiplication	I=40, J=20 >>>I * J >>> 800
/	Division	Performs division	I=30, J=20 >>>I /J >>> 2.5
%	Modulus	Returns the remainder after the division	I=40, J=20 >>>I /J >>> 0
**	Exponent	Performs exponential (power) calculation	I=2, J=3 >>>I **J >>> 8
//	Floor Division	Performs division, removes the decimal value, and returns the quotient value	I=30, J=20 >>>I//J >>> 1

Relational Operators in Python

Operator	Operator Name	Description	Example
==	Equal to	If values of two operands are equal, then it returns true.	I = 20, J = 20 (I == J) is True
!=	Not Equal to	If values of two operands are not equal, then it returns true.	I = 20, J = 20 (I == J) is False
<	Less than	If the value of the left operand is less than the value of the right operand, then it returns true.	I = 40, J = 20 (I < J) is False
>	Greater than	If the value of the left operand is greater than the value of the right operand, then it returns true.	I= 40, J = 20 (I > J) is True
<=	Less than or equal to	If the value of the left operand is less than or equal to the value of the right operand, then it returns true.	I = 40, J = 20 (I <= J) is False
>=	Greater than or equal to	If the value of the left operand is greater than or equal to the value of the right operand, then it returns true.	I = 40, J = 20 (I >= J) is True
<>	Not equal to (similar to !=)	If values of two operands are not equal, then the condition becomes true.	I=40, J = 20 (I <> J) is True.

Assignment Operators in Python

Operator	Operator Name	Description	Example
=	Assignment	It assigns a value from the right-side operand to the left-side operand.	I = 40 It assigns 40 to I
+=	Add then assign	It performs addition, and then the result is assigned to the left-hand operand.	I+=J that means I = I + J
-=	Subtract then assign	It performs subtraction, and then the result is assigned to the left-hand operand.	I-=J that means I = I – J
*=	Multiply the assign	It performs multiplication, and then the result is assigned to the left-hand operand.	I*=J that means I = I * J
/=	Divide then assign	It performs division, and then the result is assigned to the left-hand operand.	I/=J that means I = I / J
%=	Modulus then assign	It performs modulus, and then the result is assigned to the left-hand operand.	I%=J that means I = I % J
**=	Exponent then assign	It performs exponent, and then the result is assigned to the left-hand operand.	I**=J that means I = I ** J
//=	Floor division then assign	It performs floor division, and then the result is assigned to the left-hand operand.	I//=J that means I = I // J

Logical Operators in Python

Operator	Operator Name	Description	Example
and	Logical AND	When both sides' conditions are true, the result is true; otherwise false.	2<1 and 2<3 False
or	Logical OR	When at least one condition is true, then result is true; otherwise false.	2<1 or 2<3 True
not	Logical NOT	Reverse the condition	Not (5>4) False

Basic Data Types

One way to categorize these basic data types is in one of four groups:

- **Numeric**:

  int, float and the less frequently encountered complex

- **Sequence:** (string) Integer Data Type – int

The int data type

- deals with integers values. This means values like 0, 1, -2 and -15, and not numbers like 0.5, 1.01, -10.8, etc.

```
x = 5
print(type(x))
```

Basic Data Types

Floating Point Data Type – float

- The float data type can represent floating point numbers, up to 15 decimal places.
- This means that it can cover numbers such as 0.3, -2.8, 5.542315467, etc. but also integers.

```
x = 5.5
print(type(x))
```

Basic Data Types

Complex Numbers - complex

- The last numeric type we need to cover is the complex type.
- It's a rarely used data type, and its job is to represent imaginary numbers in a complex pair.
- The character j is used to express the imaginary part of the number, unlike the i more commonly used in math.

```
com = complex(1 + 2j)
print(type(com))
```

Python Data Type: Strings

- **Strings:** Strings in Python are used to **store textual information**
- A string is a **sequence of characters**. E.G. ‘MITWPU’ “Pune”
- A character is simply a symbol. For example, the English language has 26 characters.
- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows either pair of **single or double quotes**.

Python Data Type: Strings

```
#defining strings in Python .   # all of the following are equivalent
my_string = 'Hello'
print(my_string)

my_string = "Pune"
print(my_string)

my_string = '''MITWPU '''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
           the world of Python programming"""
print(my_string)
```

Python Literals

String literals

- A string literal is a **sequence of characters surrounded by quotes**.
- We can use both single, double, or triple quotes for a string.
- And, a character literal is a single character surrounded by single or double quotes

```
strings = "This is Python"
char = "C"
multiline_str = """This is a multiline string with more than
one line code."""
unicode = u"\u00dcnic\u00f6de"
raw_str = r"raw \n string"

print(strings)
print(char)
print(multiline_str)
print(unicode)
print(raw_str)
```

The string u"\u00dcnic\u00f6de" is a Unicode literal which supports characters other than English.

In this case, \u00dc represents Ü and \u00f6 represents ö.

r"raw \n string" is a raw string literal.

Python Data Type: Strings

Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string.
The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator

```
 #Accessing string characters in Python
str = 'Python Programming'
print('str = ', str)

#first character
print('str[0] = ', str[0])

#last character
print('str[-1] = ', str[-1])

#slicing 2nd to 5th character
print('str[1:5] = ', str[1:5])

#slicing 6th to 2nd last character
print('str[5:-2] = ', str[5:-2])
```

```
str[22]
IndexError: string index out of range
# index must be an integer
str[2.2]
...
TypeError: string indices must be integers

str="python"
print(str)
```

Python Data Type: Strings

```
str[22]
IndexError: string index out of range
# index must be an integer
str[2.2]
...
TypeError: string indices must be integers

str="python"
print(str)

str="python"
str[1]=q
print(str)
```

Strings **are immutable**. This means that elements of a string cannot be changed once they have been assigned

```
str1="python"
print(str1)

print(str1)

#del str
print(str1)
```

We cannot delete or remove characters from a string

Python String Operations

Concatenation of Two or More Strings

```
# Python String Operations
str1 = 'Hello'
str2 ='World!'

# using +
print('str1 + str2 = ', str1 + str2)

# using *
print('str1 * 3 =', str1 * 3)
```

```
# Iterating through a string
count = 0
for x in 'Hello World':
    if(x == 'l'):
        count += 1
print(count,'letters found')
```

String Membership Test

```
't' in 'mitwpu pune‘
‘x' in 'mitwpu pune‘
```

Built-in functions to Work with Python

enumerate() and len(). The enumerate() function returns an enumerate object. It contains the index and value of all the items in the string as pairs. This can be usef for iteration.

Similarly, len() returns the length (number of characters) of the string.

```
str = 'MITWPU'
# enumerate()
xx = list(enumerate(str))
print('list(enumerate(str) = ',xx)

#character count
print('len(str) = ', len(str))
```

Common Python String Methods

Some of the commonly used methods are **lower(), upper(), join(), split(), find(), replace()**

```
"MITwpu".upper()

"MITwpu".lower()

"This will split all words into a list".split()

' '.join(['This', 'will', 'join', 'all', 'words', 'into', 'a', 'string'])

'Welcome to MITWPU Pune'.find('co')

'Happy New Year'.replace('Happy','Brilliant')
```

Control Structures: Conditional statement

- **Python if...else Statement**
- In this article, you will learn to create decisions in a Python program using different forms of if..else statement.
- Python if Statement Syntax

```
if test expression:
        statement(s)
```

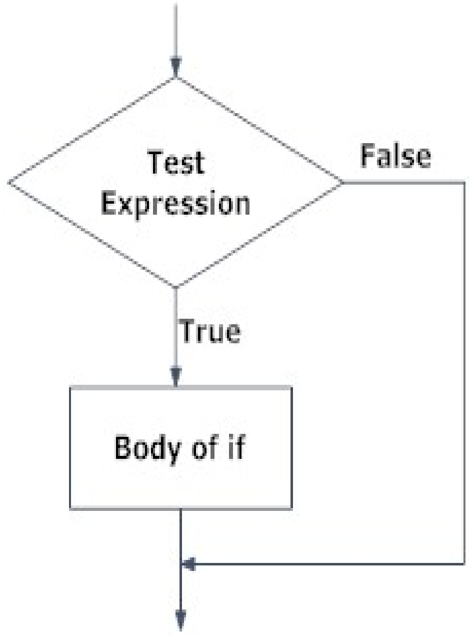


Fig: Operation of if statement

Example: Python if Statement

# If the number is positive, we print an appropriate message

```
num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

Output

```
3 is a positive number
This is always printed
This is also always printed.
```

Control Structures: Conditional statement

Python if...else Statement

Syntax of if...else

```
if test expression:
    Body of if
else:
    Body of else
```

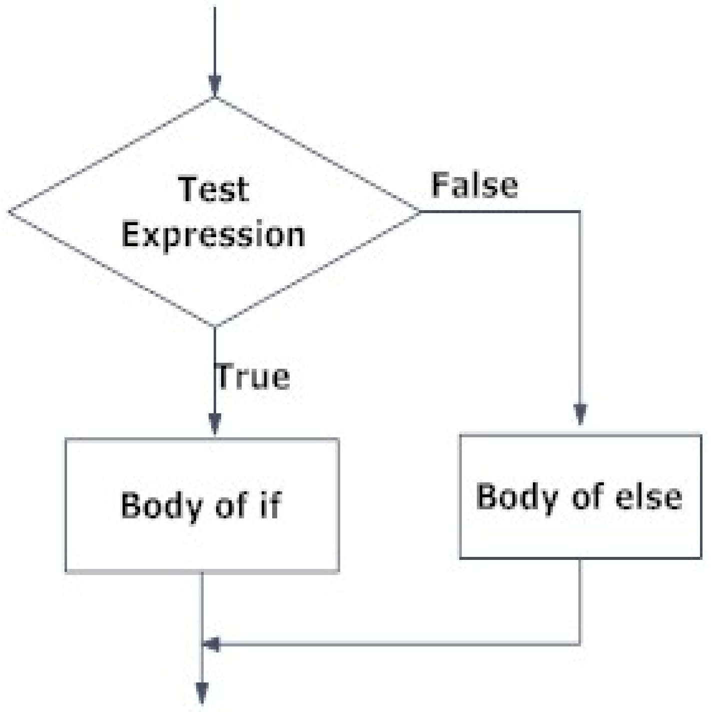


Fig: Operation of if...else statement

Example of if...else
# Program checks if the number is positive or negative
# And displays an appropriate message

```
num = 3

# Try these two variations as well.
# num = -5
# num = 0

if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

Output

```
Positive or Zero
```

Exercise4: **Accept Student Name, Roll Number and Marks of the 3 subjects from the user.**
Calculate the percentage of the marks and display it.
Display the Subject with Highest and lowest marks.

Exercise 5: **Accept an integer number form the user and display is it an even number or odd number.**

Built in Functions

Built in Mathematical functions:

• abs(x)	#absolute value of x
• pow(x,y)	# value of x raised to y
• min(x1,x2,…..)	#smallest argument
• max (x1,x2,…..)	#largest argument
• divmod(x,y)	# returns a pair(x//y, x%y)
• bin(x)	#Binary equivalent
• oct(x)	#octal equivalent
• hex(x)	#hexadecimal equivalent
• round(x, [,n])	#x rounded to n digits after decimal point

Mathematical Functions in math module

- You need to import math module by command
  import math

sqrt(x)	#square root of x
factorial(x)	#factorial of x
fabs(x)	#absolute value of flaot x
log(x)	#natural log of x (log to the base e)
log10(x)	#base-10 logarithm of x
exp(x)	#e raised to x
trunc(x)	#truncates to integer
ceil(x)	#smallest integer>=x
floor(x)	#largest integer <= x
modf(x)	#fractional and integer parts of x
round()	#function can round to a specific number of decimal places #whereas **truc(), ceil()** and **floor()** always round to zero decimal places.

Random number generation functions from random module

- You need to import random module by command

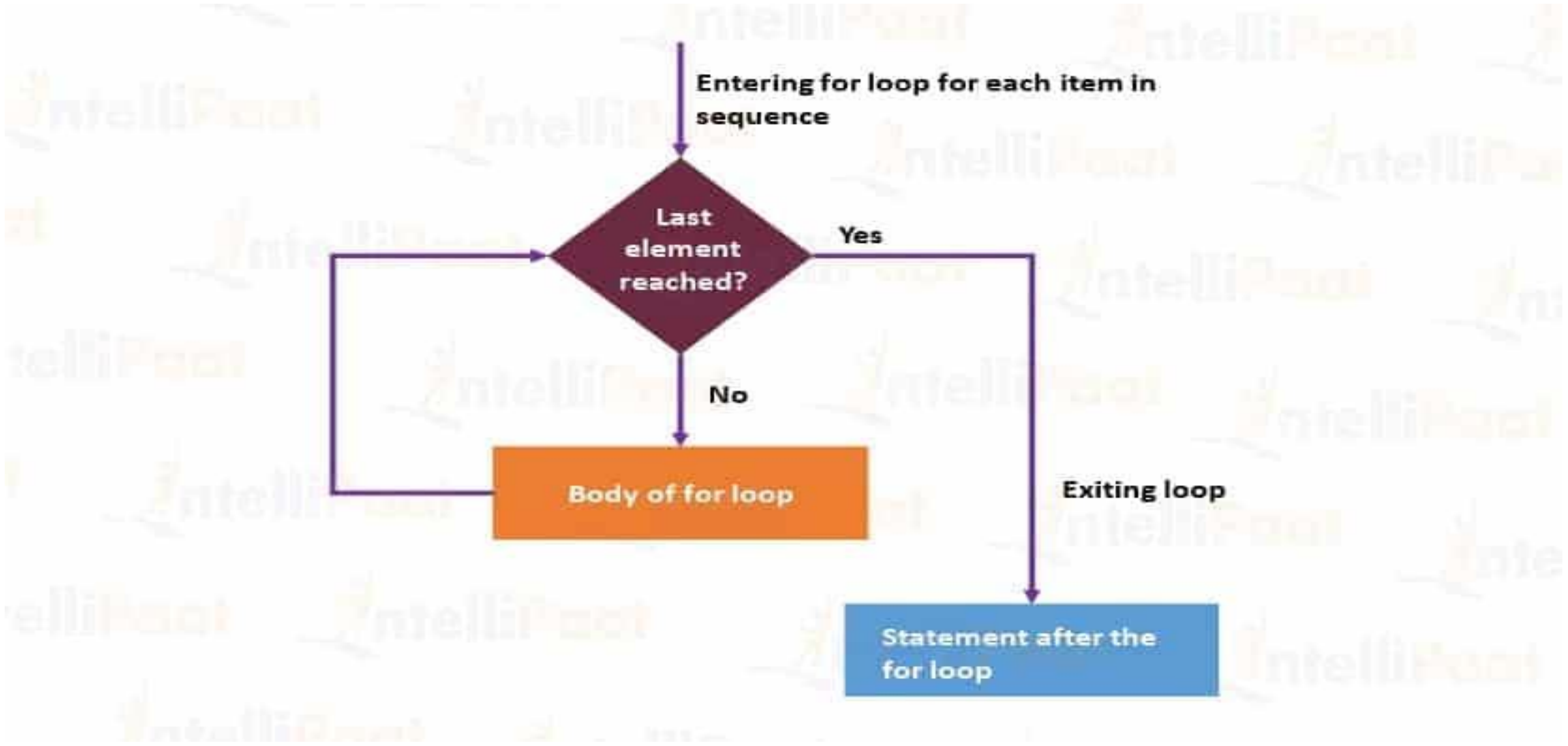
  import random

• random (x)	#random number between 0 and 1
• randint(start, stop)	#random number in the range
• seed()	#sets current time as seed for random number generation
• seed(x)	#sets x as a seed for random number generation logic

Control-Flow: For Loop

For Loop:

- For loops in Python, just like any other language, are used to repeat a block of code for a fixed number of times
- In Python, the iteration and incrementing value are controlled by generating a sequence


Entering for loop for each item in sequence
Last element reached?
Yes
No
Body of for loop
Exiting loop
Statement after the for loop

Control-Flow: For Loop

For Loop:

```
for number in 1,2,3,4,5:
    print("The current number is ",number)
```

range is a function that returns a series of numbers under an iterable form, thus it can be used in for loops:

```
for i in range(10):
    print(i)
for i in range(2,10,2):
    print(i)

for number in range(1,5):
    print ("The current number is ",number)

for number in range(1,7,2):
    print ("The current number is ",number)

for number in range(5,0,-1):
    print ("The current number is ",number)
```

Control-Flow: For Loop

For Loop:

```
numbers_list = [1,2,3,4,5,6,7]

for i in numbers_list:
    square = i*i
    print("The square of", i, "is", square)

for x in ['one', 'two', 'three', 'four']:
    print(x)
```

# Else in Python For Loop

```
for i in range(1,6):
    print(i)
else:
    print(" All iterations completed")
```

Exercise 6: Accept 5 numbers from the user and display their cube values.

Control-Flow: For Loop

```
# Nested Loop
number_of_passengers=5
number_of_baggage=2
security_check=True

for passenger_count in range(1, number_of_passengers+1):

    for baggage_count in range(1,number_of_baggage+1):

        if(security_check==True):

            print("Security check of passenger:", passenger_count, "-- baggage:", baggage_count,"baggage cleared")

        else:

            print("Security check of passenger:", passenger_count, "-- baggage:", baggage_count,"baggage not cleared")
```

Control-Flow: For Loop

break statement

- while loops, for loops can also be prematurely terminated using the break statement.
- The break statement will immediately terminate the execution of the loop and transfer the control of the program to the end of the loop.

```
number_list = [2,3,4,5,6,7,8]
for i in number_list:
    print(i)
    if i == 5:
        break
```

continue Statement : the continue statement can also be used in Python for loops to terminate the ongoing iteration and transfer the control to the beginning of the loop to continue the next iteration.

```
number_list = [2,3,4,5,6,7,8]
for i in number_list:
    if i == 5:
        continue
    print(i)
```

Control-Flow: For Loop

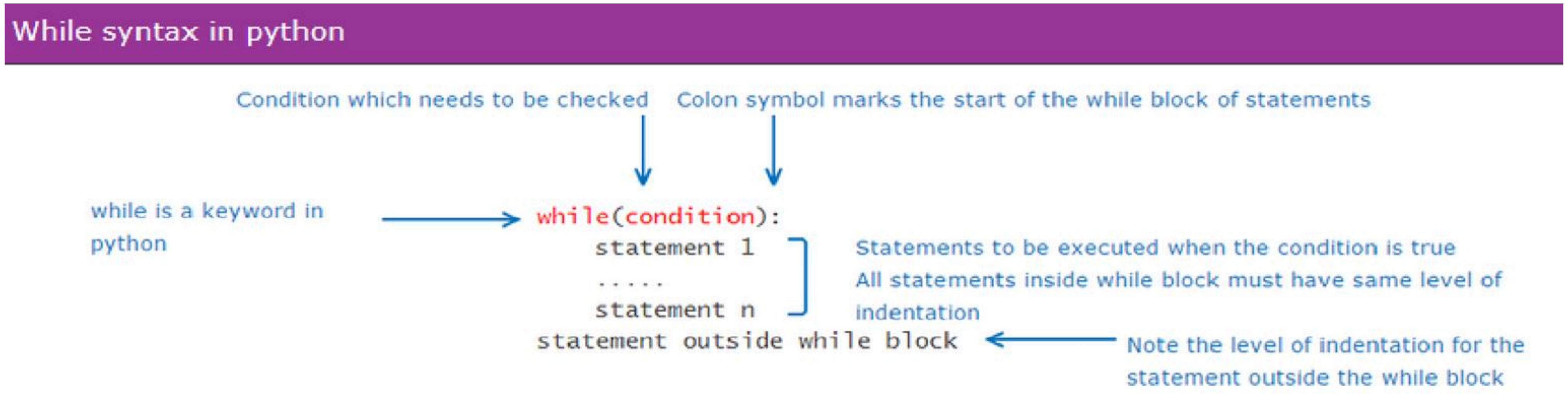
```
for passenger in "A","A", "FC", "C", "FA",  "SP", "A", "A":
    if(passenger=="FC" or passenger=="FA"):
        print("No check required")
        continue

    if(passenger=="SP"):
        print("Declare emergency in the airport")
        break

    if(passenger=="A" or passenger=="C"):
        print("Proceed with normal security check")


    print("Check the person")
    print("Check for cabin baggage")
```

Control-Flow: While Loop



While syntax in python
Condition which needs to be checked
Colon symbol marks the start of the while block of statements
while is a keyword in python
Statements to be executed when the condition is true
All statements inside while block must have same level of indentation
Note the level of indentation for the statement outside the while block

```
while(condition):
    statement 1
    .....
    statement n
statement outside while block
```

- While loop statements in Python are used to repeatedly execute a certain statement as long as the condition provided in the while loop statement stays true.
- While loops let the program control to iterate over a block of code.

Syntax of While Loop in Python:

```
while test_expression:
    body of while
```

Control-Flow: **While Loop**

```
a = 0
while a<10:
    a = a+1
    print(a)
a = 1

while a<5:
    print("condition is true")
    a=a+1
else:
    print("condition is false now")

a = 10
while True:
    a = a-1
    print(a)
    if a<7:
        break
        print('Done.')
```

```
a = 1
while a <5:
    a += 1
    if a == 3:
        break
    print(a)

a = 1

while a <5:
    a += 1
    if a == 3:
        continue
    print(a)
```

Assignment No. 2

Write a python program to find largest of three numbers