A.T.S.S.'s

College of Business Studies and Computer Applications

Chinchwad, Pune 19

(Affiliated to Savitribai Phule Pune University, Recognized by Govt. of Maharashtra , Accredited by NAAC)

Workbook on

Basic Of Python Programming

Published By

ATSS CBSCA PUBLICATIONS

Address: C2, MIDC, Opp. Niramaya Hospital, Station, Chinchwad East,

Chinchwad, Pune, Maharashtra 411019

Ph. No.020-27472079, Mob. No. 9850886800,

Website: www.atsscollege.org

Printed at

SHREE KEDAR MUDRAN

Raka Palace, Basement Shop No. 4/5/6,

Chaphekar Chowk, Near Indian Overseas Bank,

Chinchwad, Pune-411033

Edition

I

Edited by

Ms. Gloriya Sarjerao Kardile

Mr. Vishal Popat Pawar

ISBN NO. 978-81-952723-0-3

Outcomes

By the end of the session participants will be able to:

- Explain Python features.
- Select appropriate data type and operators
- Installation IDE

Content

- What is Python?
- History of Python
- Version
- Application of Python
- Python Features
- Installation
- Familiar with Python working environment

History of Python

- It was created by **Guido van Rossum** in 1991 and further developed by the Python Software Foundation.
- when he was working in National Research Institute of Mathemetics and Computer Science (NRIM&CS) in Netherlands.
- He stared implementing it from 1989 and finished and released first working version of Python in 1991.

History of Python

- An important goal of Python's developers is keeping it fun to use.
- This is reflected in the language's name—a tribute to the British comedy group Monty Python.
- M*onty Python's Flying Circus* is a sketch comedy show.
- as a successor to the ABC language.
- Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter.
- He thought to create a scripting language as a “Hobby” in Christmas break in He studied all the languages like ABC (All Basic Code), C, C++, Modula-3, Smalltalk, Algol-68 and Unix Shell and collected best features.

What is Python?

- **Python** is an interpreted, high-level and general-purpose programming language.
- Python is a widely used general-purpose, high level programming language.
- It was designed with an **emphasis on code readability**, and its syntax allows **programmers to express their concepts in fewer lines of code.**
- Python is a programming language that lets you work quickly and integrate systems more efficiently.

What is Python?

- Python is dynamically typed and garbage-collected.
- It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming.
- Python is often described as a "**batteries included**" language due to its comprehensive standard library.

```
int  a=10;
```

```
a=“hello”
```

Core philosophy

- The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:[57]
- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Python Version

- https://www.python.org/doc/versions/
- https://en.wikibooks.org/wiki/Python_Programming/Version_history

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other

- If
- {
  - }

Python

```
If
   print(“hello”)
```

Who use Python?

- **Google** - Python is one of the key language used in google.
- **Philips** - Philips uses Python for the sequencing language
- **Quora** - Quora also chose Python for its development
- **NASA** - Johnson Space center uses Python in its Integrated Planning System as the standard scripting language

•**Walt Disney Feature Animation** - Walt Disney Feature Animation is also using Python to make their animation production system more efficient in scripting.

- **Instagram** - Instagram also uses Python for its backend
- **YouTube , DropBox, Pinterest**

Yahoo(Maps)) Mozilla, Dropbox Microsoft Cisco ,Spotify, YouTube,reddit etc.

https://dzone.com/articles/best-python-companies-work

Python Features

- Easy to Write
- Easy to Understand
- Object-Oriented
- Robust Standard Libraries
- Supports Various Programming Paradigms
- Support for Interactive Mode
- Dynamically Typed and Type Checking
- Databases and GUI Programming
- Extensible
- Portable
- Scalable
- Integrated
- Automatic Garbage Collection
- Free and Open Source

Feature

Easy-to-learn − python libraries use simple English phrases as it's keywords. Thus it's very easy to write code in python.

For eg:-

- Writing code for function doesn't use curly braces to delimit blocks of code.

Easy to Understand

- This is the most powerful feature of python language which makes it everyone's choice. As the keyword used here are simple English phrases thus it is very easy to understand.

Object-Oriented

Feature

Robust Standard Libraries

- The libraries of python are very vast that include various modules and functions that support various operations working in various data types such as regular expressions etc.

Supports Various Programming Paradigms

- With support to all the features of an object-oriented language, Python also supports the procedure-oriented paradigm. It supports multiple inheritances as well. This is

Feature

Support for Interactive Mode

- Python also has support for working in interactive mode where one can easily debug the code and unit test it lines by line. This helps to reduce errors as much as possible.

Automatic Garbage Collection

- Python also initiates automatic garbage collection for great memory and performance management. Due to this memory can be utilized to its maximum thus making the

Feature

Dynamically Typed and Type Checking

- This is one of the great feature of python that one need not declare the data type of a variable before using it. Once the value is assigned to a variable it's datatype gets defined Thus type checking in python is done at a run time, unlike other programming languages.

For eg-

- v=7;// here type of variable v is treated as an

Feature

Databases

- Database of an application is one of the crucial parts that also needs to be supported by the corresponding programming language being used.

GUI Programming

- Python being a scripting language also supports many features and libraries that allow graphical development of the applications

Feature

Extensible

- This feature makes use of other languages in python code possible. This means python code can be extended to other languages as well thus it can easily be embedded in existing code to make it more robust and enhance its features. Other languages can be used to compile our python code.

Portable

- A programming language is said to be portable if it allows us to code once and runs anywhere feature. Means, the platform where it has been coded and where it is going to run need not be the same. This feature allows one of the most valuable features of object-oriented languages- reusability. As a developer, one needs to code the solution and generated its byte code and need not worry about the environment where it is going to run.eg-one can run a code

Feature

Scalable

- This Language helps to develop various systems or applications that are capable of handling a dynamically increasing amount of work. These type of applications helps a lot in the growth of the organization as they are strong enough to handle the changes upto some extent.

Free and Open Source

- One can easily download it and use it as required as well as share it with others. Thus it gets improved every day.

Integrated

- Python can be easily integrated with other

Feature

- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Application Python

Web Applications

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python.

Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS.

Application of Python

Python's standard library supports many Internet protocols:

- HTML and XML
- JSON
- E-mail processing.
- Support for FTP, IMAP, and other Internet protocols.
- Easy-to-use socket interface.

Application of

Scientific and Numeric Computing

There are numerous libraries available in Python for scientific and numeric computing.

SciPy is a collection of packages for mathematics, science, and engineering.
Pandas is a data analysis and modeling library.
IPython is a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing.

Application

Scientific and Numeric Computing

machine learning, data mining and deep learning.

libraries that exist already such as Pandas, Scikit-Learn, NumPy and so many more.

Application

Data Science and Data Visualization

- You study the data you have, perform operations and extract the information required.
- Libraries such as Pandas, NumPy help you in extracting information.
- You can even visualize the data libraries such as Matplotlib, Seaborn, which are helpful in plotting graphs and much more.
- This is what Python offers you to become a Data

Desktop GUI

- Python can be used to program desktop applications.
- It provides the Tkinter library that can be used to develop user interfaces.
- There are some other useful toolkits such as the wxWidgets, Kivy, PYQT that can be used to create applications on several platforms.

Business Applications

- Business Applications are different than our normal applications covering domains such as e-commerce, ERP and many more.

- They require applications which are scalable, extensible and easily readable and Python provides us with all these features.

Audio and Video Applications

- Video and audio applications such as TimPlayer, Cplay have been developed using Python libraries and they provide better stability and performance compared to other media players.

Application

Image Processing and Graphic Design Applications:

Python has been used to make 2D imaging software such as Inkscape, GIMP, Paint Shop Pro and Scribus.

Application

Games

•Python has various modules, libraries and platforms that support development of games.

•PySoy is a 3D game engine supporting Python 3, and PyGame provides functionality and a library for game development.

•Games such as Civilization-IV, Disney’s Toontown Online, Vega Strike etc. have been built using Python.

Python IDE

- ***Windows***: There are many interpreters available freely to run Python scripts like IDLE (Integrated Development Environment) that comes bundled with the Python software downloaded from **http://python.org/**.

- ***Linux***: Python comes **preinstalled** with popular Linux such as Ubuntu and Fedora. To check which version of Python you're running, type "python" in the terminal emulator. The interpreter should start and print the version number.

Python IDEs

1. Online Compiler from Programiz
2. IDLE (Free) - When you install Python, IDLE is also installed by default.
3. Sublime Text 3- Sublime Text is a popular code editor that supports many languages

including Python. It's fast, highly customizable and has a huge community

4. Atom-Atom is an open-source code editor developed by Github that can be used for Python

development (similar Sublime text).

5. Thonny- Thonny is a Python dedicated IDE that comes with Python 3 built-in. Once you

6. PyCharm- PyCharm is an IDE for professional developers. It is created by JetBrains, a

company known for creating great software development tools.

7. Visual Studio Code- Visual Studio Code (VS Code) is a free and open-source IDE created

by Microsoft that can be used for Python development.

8. Vim- Vim is a text editor pre-installed in macOS and UNIX systems. For Windows,

Python IDEs

9. Spyder - Spyder is an open-source IDE usually used for scientific development.
10. Jupyter Notebook - open-source software that allows you to create and share live code,

visualizations, etc.

11. Eclipse + PyDev - Eclipse is a popular IDE that can be used for Python development using

PyDev plugin.

Anaconda Navigator is a desktop graphical user interface (GUI) that allows you to launch
applications and easily manage conda packages, environments, and channels without using
command-line commands. Navigator can search for packages on Anaconda Cloud or in a local
Anaconda Repository. It is available for Windows, macOS, and Linux.

Link -> https://docs.anaconda.com/anaconda/navigator/

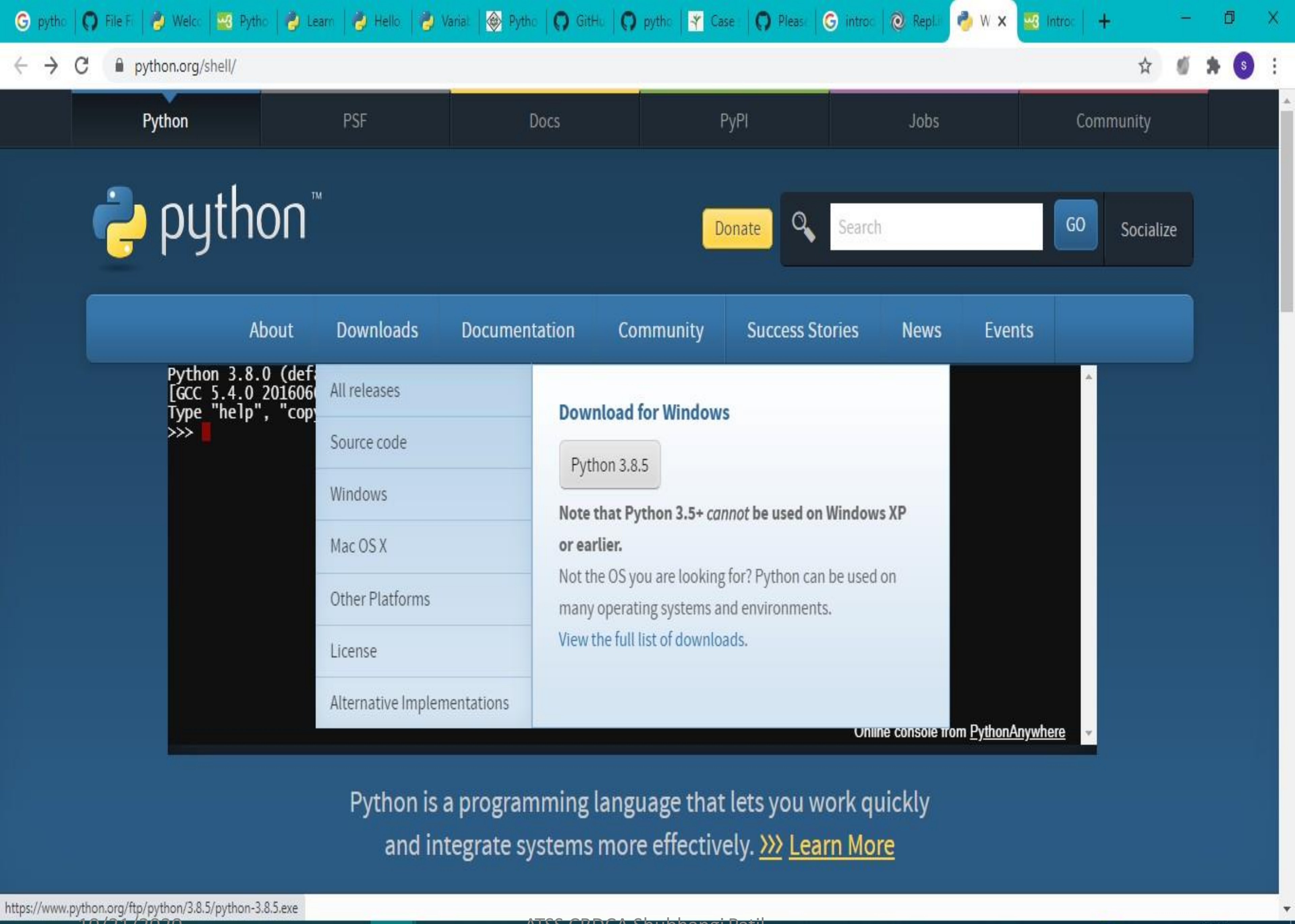
What applications can I access using Navigator?

- JupyterLab Jupyter Notebook Spyder PyCharm
- VSCode Glueviz Orange 3 App RStudio
- Anaconda Prompt (Windows only) Anaconda PowerShell (Windows only)
- Advanced conda users can also build their own Navigator applications

Downloading and installing Python

Windows

- Installing Python on Windows is a lot like installing any other program.
- Go to the official Python website.
- Move your mouse over the blue Downloads button, but don't click it. Then click the button that downloads the latest version of Python.
- Run the installer.
- Make sure that the launcher gets installed and click Install Now



```
Python 3.8.0 (def
[GCC 5.4.0 201606
Type "help", "cop
>>>
```

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.

Online console from PythonAnywhere

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

https://www.python.org/ftp/python/3.8.5/python-3.8.5.exe

Online Interpreter

- https://repl.it/languages/python3
- https://www.onlinegdb.com/online_python_compiler

Running Python

- Next we'll learn to run Python on a PowerShell or terminal. There are several other ways to run Python, but if you learn this way now it's going to make things easier later.

Windows

- Open a PowerShell from your start menu or start screen.
- Type py and press Enter.

Getting started with Python

Launch Python.

- The >>> means that Python is ready and we can enter a command.
- The basic idea is really simple: we enter a command, press Enter, enter another command, press Enter and keep going.

Software requirements

- Anaconda jupyter lab
- https://anaconda.org/anaconda/jupyter
- Open Anaconda Navigator
- Launch JupyterLab
- Select notebook

print

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

DAY -2

- The print() function produces a more readable output, by omitting the enclosing quotes and by printing escaped and special characters:

```
Print(‘shubha\ngi’)
```

Raw string

If you don’t want characters prefaced by \ to be interpreted as special characters, you can use *raw strings* by adding an r before the first quote:

- print(r'C:\some\name') *# note the r before the quote*

```
C:\some\name
```

Print

- word='word‘‘
- Sentence="This is sentence"
- Paragraph=„ „ „is multiline „ „ „

Python Identifiers

- **Starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).**
- Python does not allow **punctuation characters such as @, $, and % within identifiers.**
- Python is a **case sensitive programming language .**
- Identifiers are unlimited in length.

Comments

Comments are text that don't do anything when they're run.

- They can be created by typing a # and then some text after it
- It is useful when our code would be hard to understand without them.
- can be used to explain Python code.
- can be used to make the code more readable.
- can be used to prevent execution when testing code.

- Example

```
#This is a comment
print("Hello, World!")

print("Hello, World!") #This is a comment
```

Multi Line Comments

```
#This is a comment
 #written in
 #more than just one line
 print("Hello, World!")
```

Multi Line Comments

```
"""
 This is a comment
 written in
 more than just one line
 """
 print("Hello, World!")
```

Python as a calculator

- >>> 17 + 3

20

- >>> 17 – 3

14

- >>> 17 * 3

51

- >>> 17 / 3

5.666666666666667

- >>>

- >>> 4+2+1 7

Things are calculated in the same order as in math.

The parentheses (and) also work the same way.

- >>> 1 + 2 * 3 # 2 * 3 is calculated first

o/p 7

- >>> (1 + 2) * 3 # 1 + 2 is calculated first
  9

QUIZ

Q.1 In the Python statement x = a + 5 - b:

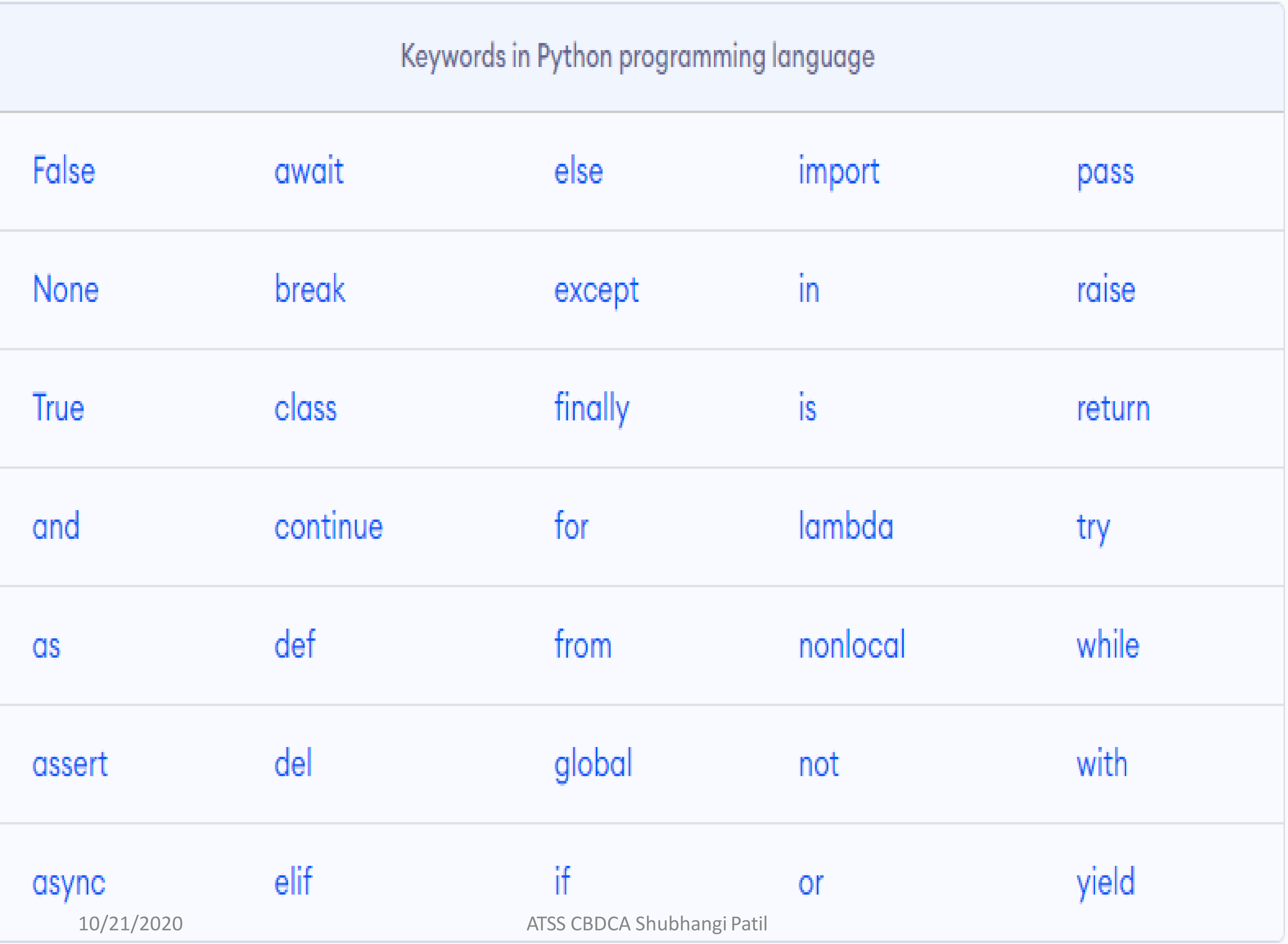
•a and b are _______

•a + 5 - b is _______

A. operands, an equation
B. operators, a statement
C. terms, a group
D. operands, an expression

Keywords

- reserved words
- cannot use a keyword as a variable name, function name or any other identifier.
- All the Python keywords contain lowercase letters only



Keywords in Python programming language

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

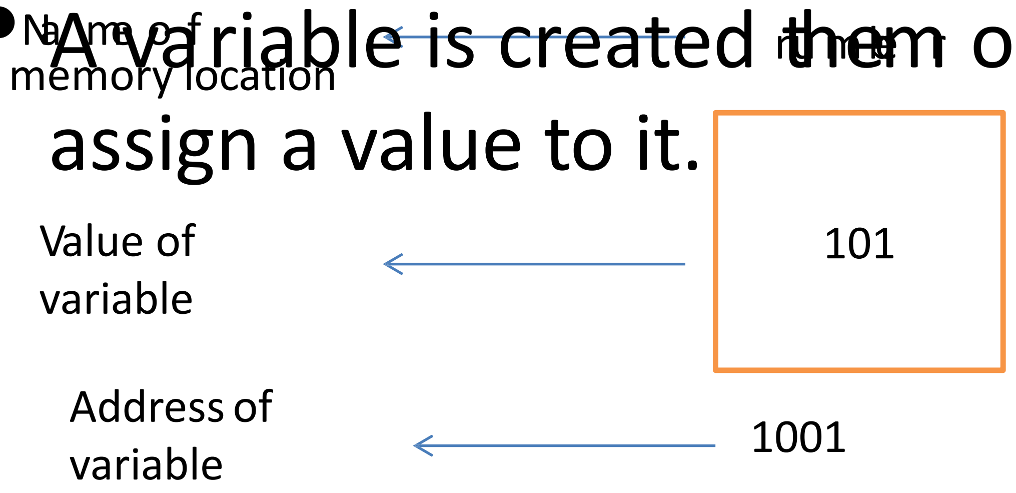
```
import keyword
>>> print(keyword.kwlist)
 ['False', 'None', 'True', 'and', 'as', 'assert',
   'async', 'await', 'break', 'class', 'continue', 'def',
   'del', 'elif', 'else', 'except', 'finally', 'for', 'from',
   'global', 'if', 'import', 'in', 'is', 'lambda',
   'nonlocal', 'not', 'or', 'pass', 'raise', 'return',
   'try', 'while', 'with', 'yield']
```

What it is?

=

Creating Variables

- Variables are containers for storing data values.
- Unlike other programming languages, Python has no command for declaring a variable.
- A variable is created them oment you first assign a value to it.



Variable Declaration

- **declaration happens automatically when you assign a value to a variable.**

- **Assign Value to Multiple Variables**

```
x, y, z=10,”ram”,20
```

•can assign the *same* value to multiple variables in one line

Values and types

- A **value** is one of the basic things a program works with, like a letter or a number. Some values we have seen so far are 2, 42.0, and 'Hello, World!'.
- These values belong to different **types**: 2 is an **integer**, 42.0 is a **floating-point number**, and 'Hello, World!' is a **string**, so-called because the letters it contains are strung together.
- If you are not sure what type a value has, the interpreter can tell you:
- >>> type(2)
- <class 'int'>
- >>> type(42.0)

- <class 'float'>

Data types

- Data types are nothing but the **different types of input data accepted** by a programming language, for defining, declaring, storing and performing mathematical & logical values/ operations.

The standard data types of python

- **Numbers:** Number data type is used to stores numeric values.
- **String:** String data type is used to stores the sequence of characters.
- **Tuple**
- **List**
- **Set**
- **Dictionary**

Boolean Values

- if an expression is True or False.
- When we evalutae expression some times it gives True and false .
- When you compare two values, the expression is evaluated and Python returns the Boolean answer:

-
  ```
  print(10 > 9)
  print(10 == 9)
  print(10 < 9)
  ```

-
  ```
  x = "Hello"
  y = 15
  ```

Note

- Any string is True, except empty strings.
- Any number is True, except 0.
- Any list, tuple, set, and dictionary are True, except empty ones.

1. Numbers

- When a number is assigned to a variable Number class object is created.
- **Consider an example:**

var a = 100, var b = 200 # var a and var b number are assigned and these are objects of number.

- The Number can have 4 types of numeric data:

int :

int stores integers

eg a=100, b=25, c=526, etc.

long:

long stores higher range of integers

eg a=908090999L, b=-0x1990999L, etc.

input function

- The input() function allows user input.

Syntax

input(*prompt)*

prompt: A String, representing a default message before the input.

Example:

```
x = input('Enter your name:')
#Output:
```

float:

float stores floating-point numbers eg a=25.6, b=45.90, c=1.290,

32.54e100

etc.

complex:

complex stores numbers eg a= 3 + 4j, b=2 + 3j,

- print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
-

- sep: string inserted between values, default a space.
- end: string appended after the last value, default a newline.

print function

Example:

```
print("Hello", end="\t")
print("Welcome to Python!")
#Output:
#Hello Welcome to Python!
```

Escape Characters

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

```
txt = “hello “shubhangi “ how r u "
```

Escape SEQUENCE

\\Backslash (\)

- print("\\")

\'Single quote (')

- print('\'')

\"Double quote (")

print("\"")

\bASCII Backspace (BS)

- print("Hello \b World!")

\nASCII Linefeed (LF)

- print("Hello \n World!")

\rASCII Carriage Return (CR)

print("Hello \r World!")

\tASCII Horizontal Tab (TAB)

- print("Hello \t World!")

\vASCII Vertical Tab (VT)

- print("Hello \v World!")

Boolean

- Booleans represent one of two values:

  True or False.

- You can evaluate any expression in Python, and get one of two answers, True or False.

String

- The string can be defined as the sequence of characters represented in the quotation marks.

- In python, the string can be quoted by single, double, or triple quotes.

- **1. Numbers**
- When a number is assigned to a variable Number class object is created.
- **Consider an example:** var a = 100, var b = 200 # var a and var b number are assigned and these are objects of number. The Number can have 4 types of numeric data:

- What happens if you use + between two strings, like "hello" + "world"? How about "hello" * "world"?
- What happens if you use + between a string and an integer, like "hello" + 3? How about "hello" * 3?
- What happens if you use + between a float and an integer, like 0.5 + 3? How about 0.5 * 3?

Summary

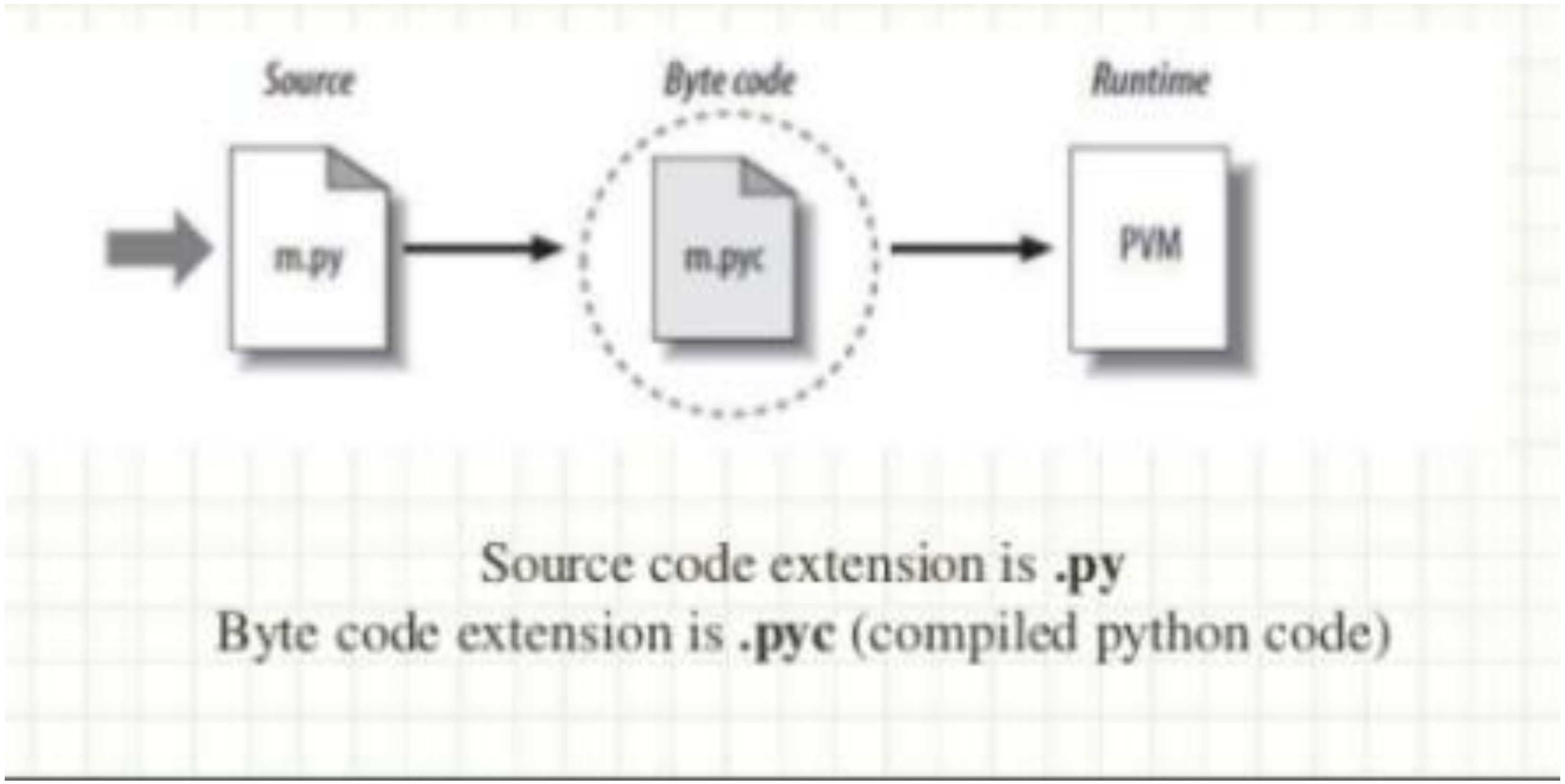
- Error messages are our friends.
- We can enter any Python commands to the interactive >>> prompt, and it will echo back the result.
- +, -, * and / work in Python just like in math.
- Pieces of text starting with a # are comments and pieces of text in quotes are strings.
- You can use single quotes and double quotes however you want.

Indentation

- Python uses indentation for blocks, instead of curly braces. Both tabs and spaces are supported, but the standard indentation requires standard Python code to use four spaces.
- For example:

```
x = 1
if x == 1:    # indented four
    spaces
    print("x is 1.")
```

Python Code Execution


Source
Byte code
Runtime
m.py
m.pyc
PVM
Source code extension is .py
Byte code extension is .pyc (compiled python code)

Source code converted into byte code,it is automatic complied,but then it is interpreted.

Unicode is also an encoding technique that provides a unique number to a character. While ASCII only encodes 128 characters, the current Unicode has more than 100,000 characters from hundreds of scripts.

# Program to find the ASCII value of the given character

c = 'p' print("The ASCII value of '" + c + "' is", ord(c))

ASCII values using the **chr()** function

```
bool(5)
abs(-5)
eval("5+2-3")
```

Type Casting

Integer

```
x = int(1)   # x will be 1
   y = int(2.8) # y will be 2
   z = int("3") # z will be 3
```

Float

```
x = float(1)     # x will be 1.0
   y = float(2.8)   # y will be 2.8
   z = float("3")   # z will be 3.0
   w = float("4.2") # w will be 4.2
```

Strings:

```
x = str("s1") # x will be 's1'
```

Python Operators

- Operators are used to perform operations on variables and values.

$$a + b = c$$

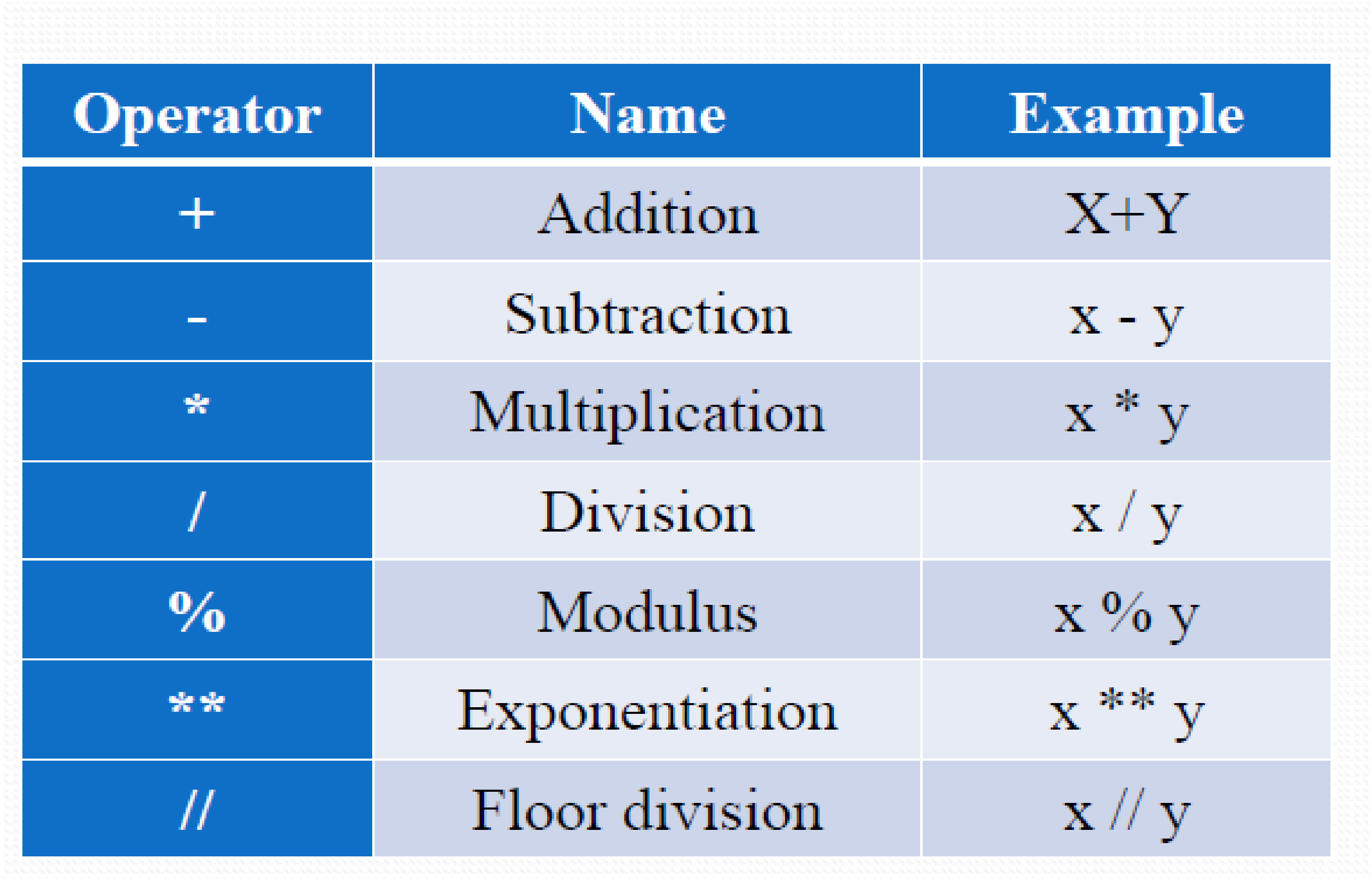
Python Operators

Python divides the operators in the following groups:

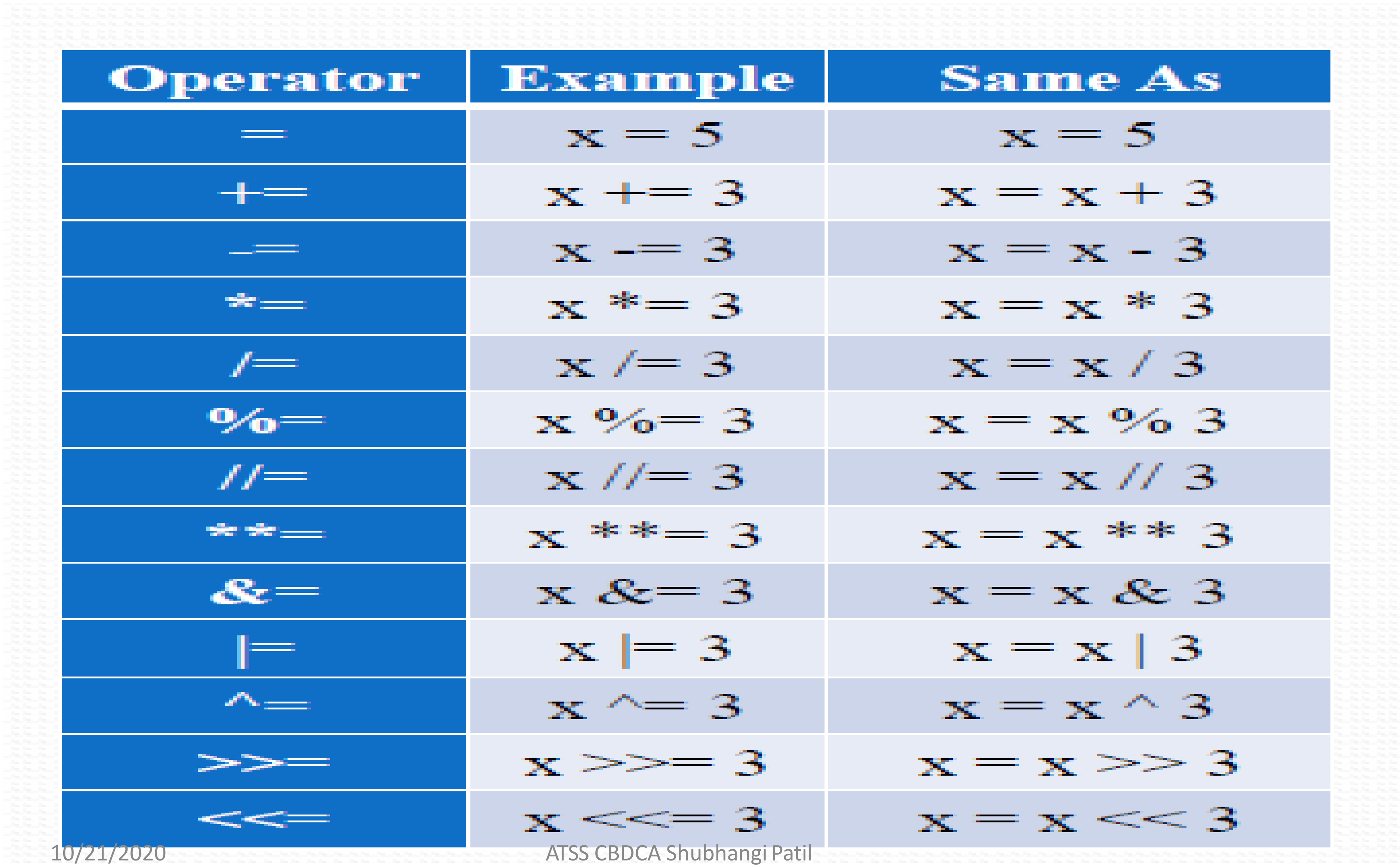
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- **Identity operators**
- **Membership operators**
- Bitwise operators

Operators

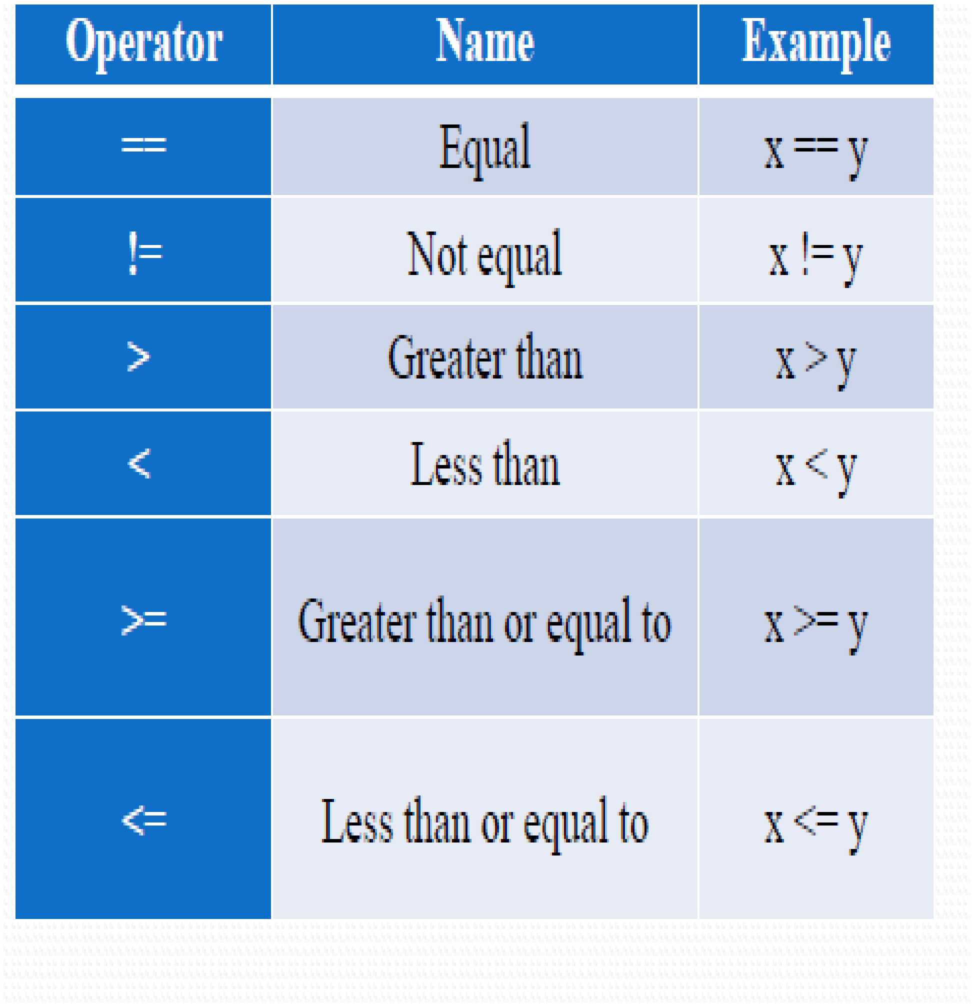
Arithmetic	Assignmewn t	Comparisio n	Logical	Identity	Membership	Bitwise
+	+=	>	and	is	in	&
_	_=	<	or	Is not	not in	|
*	*=	>=	not			!
/	/	<=				^
%	%=	==				<<
**	**=	!=				>>
//	//=					



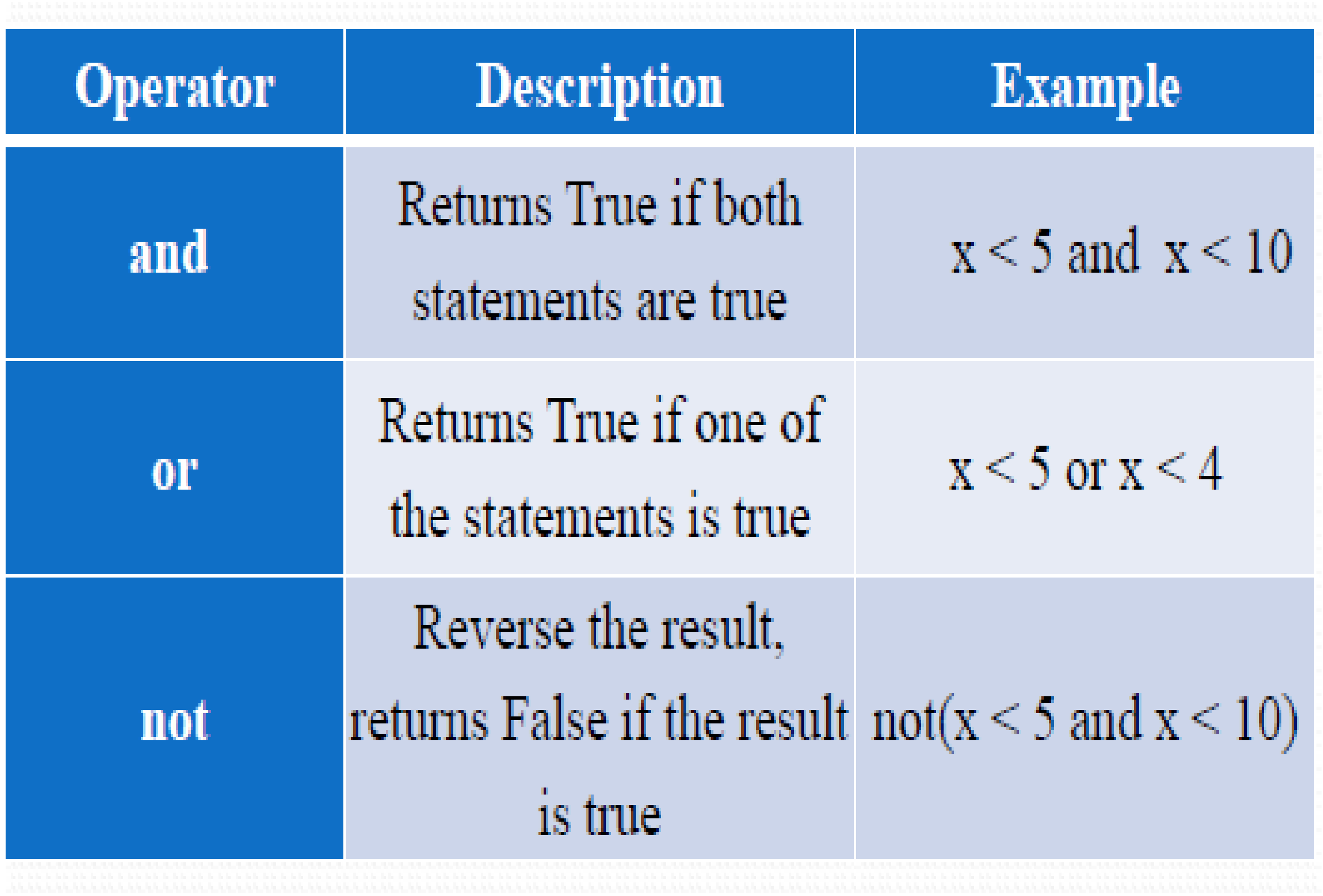
Operator	Name	Example
+	Addition	X+Y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y



Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

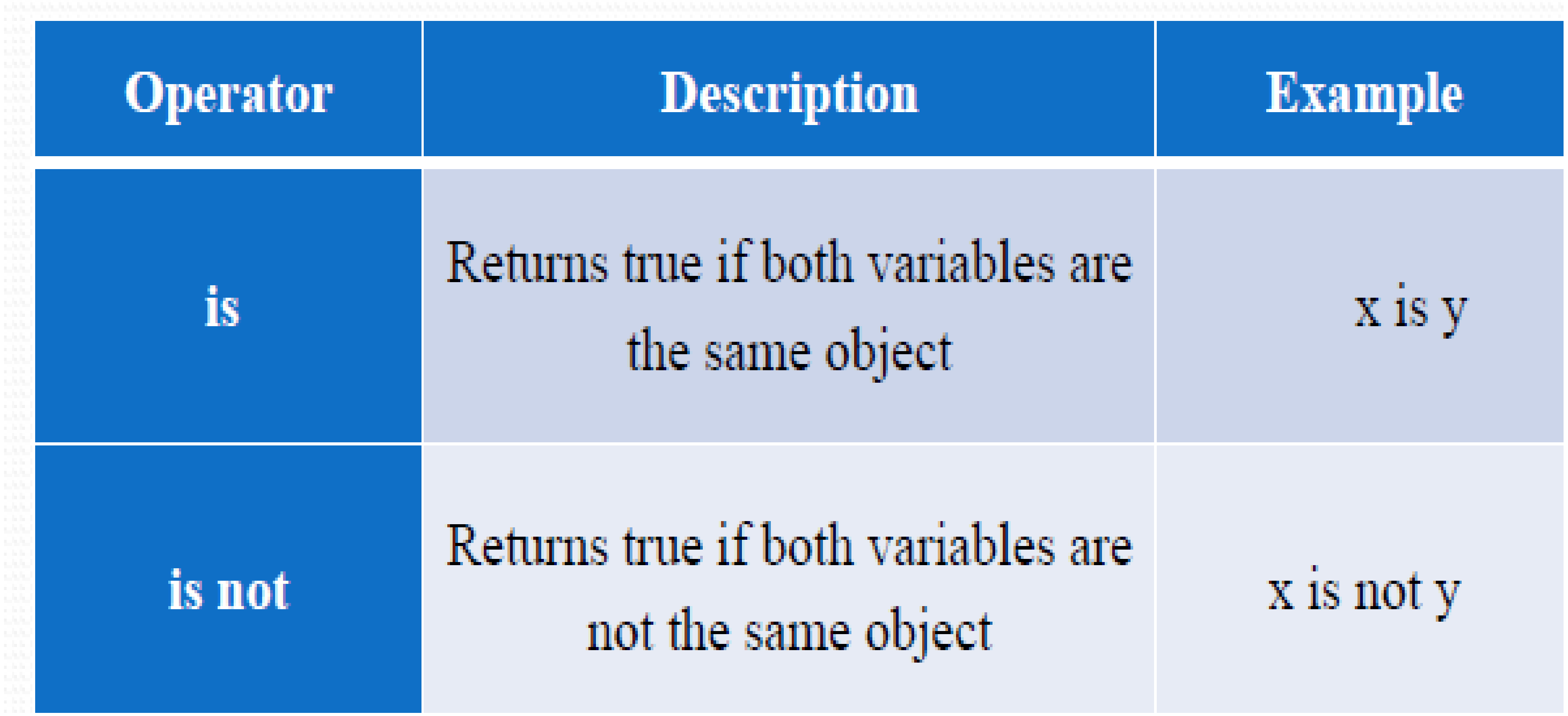


Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y



Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

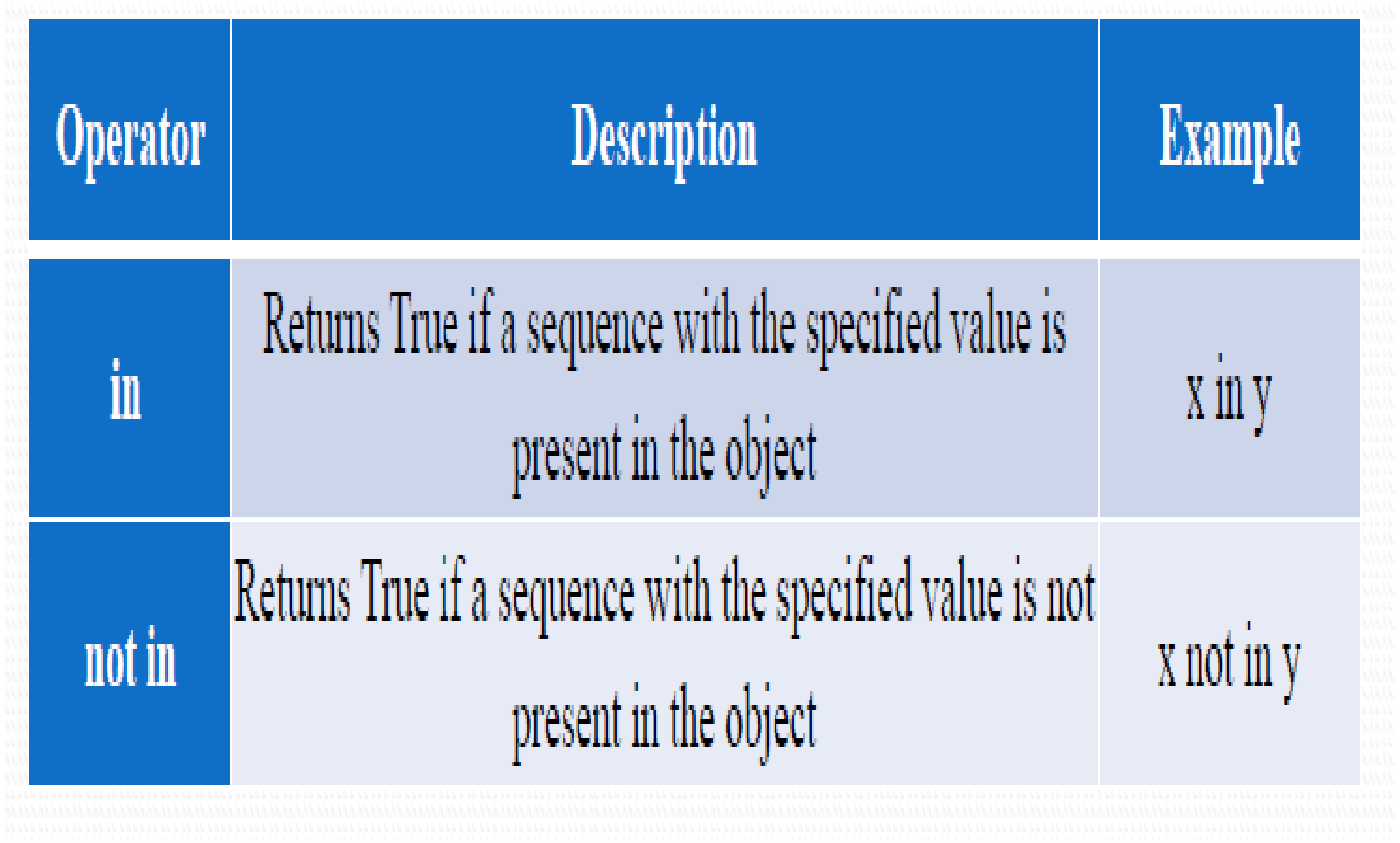
Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location



Operator	Description	Example
is	Returns true if both variables are the same object	x is y
is not	Returns true if both variables are not the same object	x is not y

```
>>> x = 5
>>> type(x) is int
True
>>> type(x) is not float
True
>>> y = 3.23
>>> type(y) is not float
 False
>>> type(y) is int
```

```
x =10
y =10
z = x
print(x is z)
# returns True because z is the same object as x
print(x is y)
# returns False because x is not the same object as
  y, even if they have the same content
print(x == y)
# to demonstrate the difference betweeen "is" and
```



Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

x = ["apple", "banana"]

print("banana" in x)

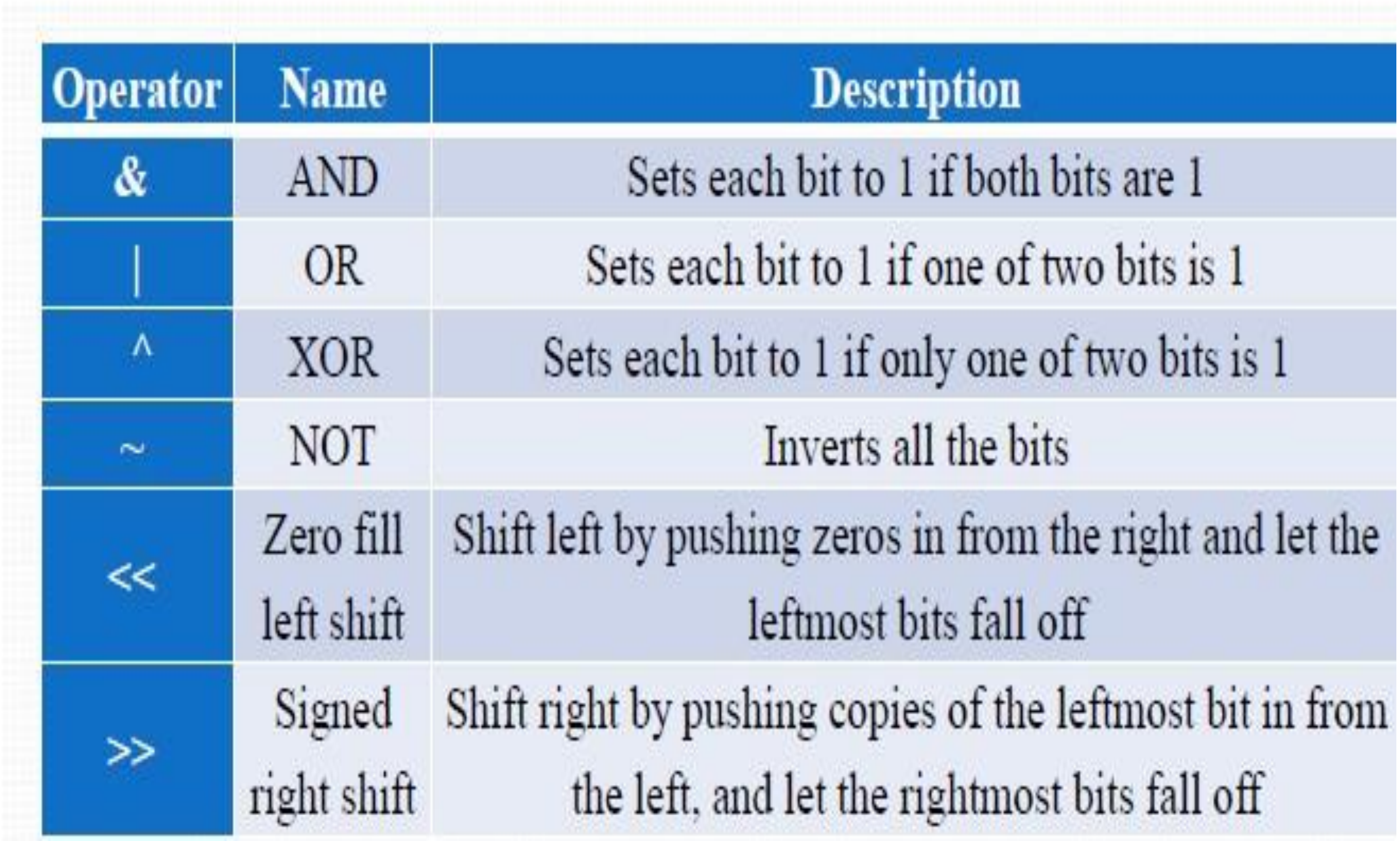
# returns True because a sequence with the value "banana" is in list

EX-2

= ["apple", "banana"]

print("pineapple" not in x)

# returns True because a sequence with the



Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

- print("a" in "apple")
- True

What is Decision making ?

- Decision making in programming is same as daily life
- decisions checking of conditions occurring while execution of the program
- specifying actions taken according to the conditions
- Decision structures produce TRUE or FALSE outcome after evaluating expressions
- You need to determine which action to take and which statements to execute if outcome is

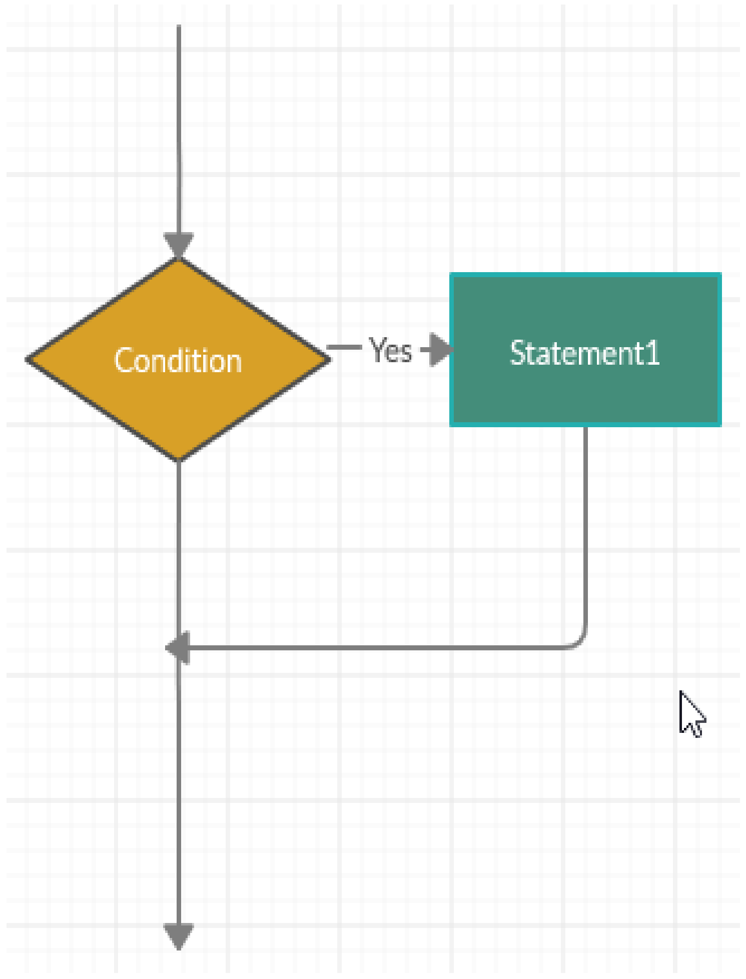
- Comparison operators and Logical operators
- **Comparison operators are used to compare values. It returns either True or False according to the condition.**

Decision Making Statements in Python

- if statements
- if else statements
- elifstatements
- nested if conditions
- single statement conditions

Note: Indentation

Python relies on indentation to define scope in the code. In C/C++/Java often use curly-brackets for this purpose



Syantax:

if **<test expr>:**

  <statement>

Example : Check employee age less than 18 then display message “Not allowed” if there are multiple statements then maintain indentation

```
if <test expr>:
   <statement1>
   <statement2>
   <statement3>
<statement4> #outside if
```

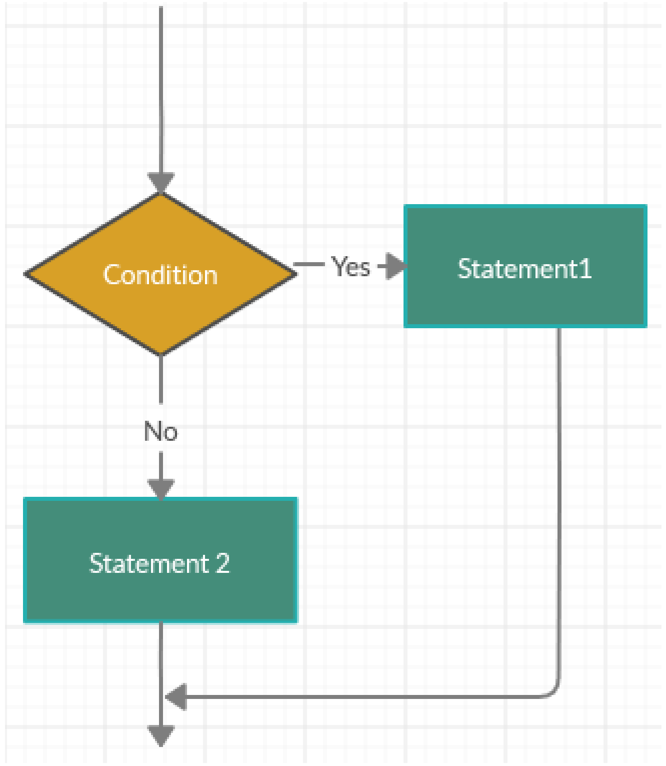
If statement

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

$\downarrow$

Indentation

if else



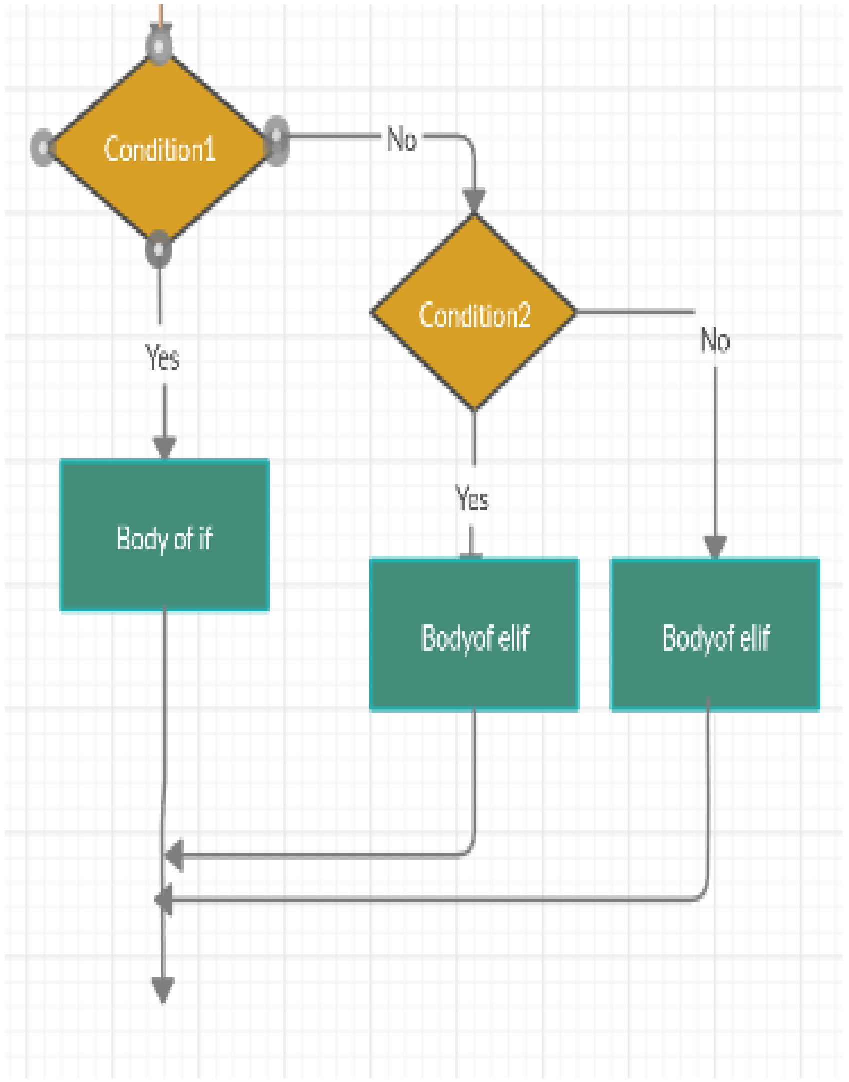
```
if <test expr>:
    <statement1>
else:
    <statement 2>
```

Example: Check given number is odd or even

else without the elif:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

elif



```
if <test expr>:
    <statement1>
elif< test expr> :
    <statement 2>
else:
    <statement 3>
```

Example : Check given number is positive, negative or zero

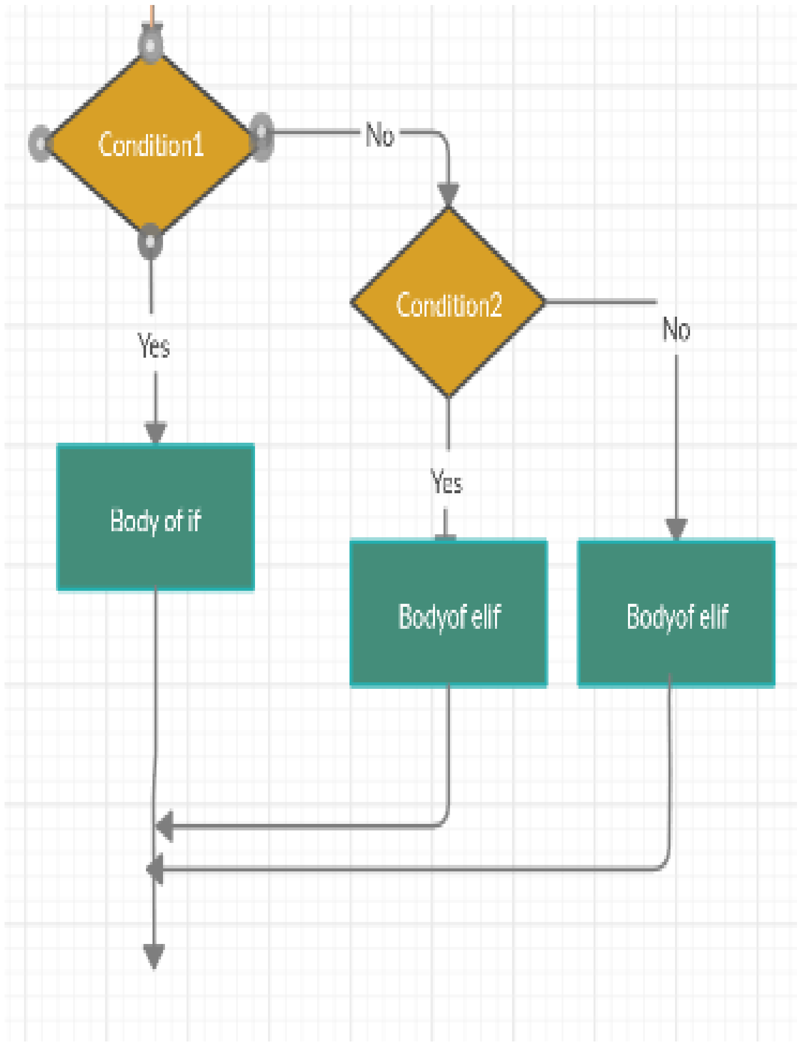
else

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

elif

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

$\longrightarrow$ if this not true than try this



```
if <test expr>:
      if < test expr> :
               <statement 1>
      else:
               <statement 2>
else:
      <statement3>
```

Example

Check given number is positive, negative or zero

```
n=int(input("enter number:"))
if n>=0:
      if n>0:
               print (n, "is positive Number")
      else:
               print (n, " is zero")
else:
      print (n, " is negative Number")
```

Short Hand If

- If you have only one statement to execute, you can put it on the same line as the if statement.

Example

One line if statement:

```
if a > b: print("a is greater than b")
```

Short Hand If ... Else

- one statement to execute, one for if, and one for else

```
a = 2
b = 330
print(a) if a > b else print(b)
```

This is known as **Ternary Operators**, or **Conditional Expressions**.

```
if (num1 >= num2) and (num1 >= num3):
     largest = num1
 elif (num2 >= num1) and (num2 >= num3):
     largest = num2
else: largest = num3
   print("The largest number is", largest)
```

Quiz

logical or operator

```
x = 10
y = 0
z = -12
if x > 0 or y > 0:
   print(“ x or y is greater than 0")
else:
   print("No number is greater than 0")
if x > 0 and y >= 0:
   print(“x and y are greater than or equal to 0")
```

Quiz

Consider list of Friends=“Deepa”, “Rupa”,”Hemant”, “Nisha”, “Devid”

What is output of following?

A.'DEEPA' inFriends

B.'Deepa'in friends

C.'Deepa' inFriends

D.'Rohit' not in Friends

Quiz time

- What is the error in following code?

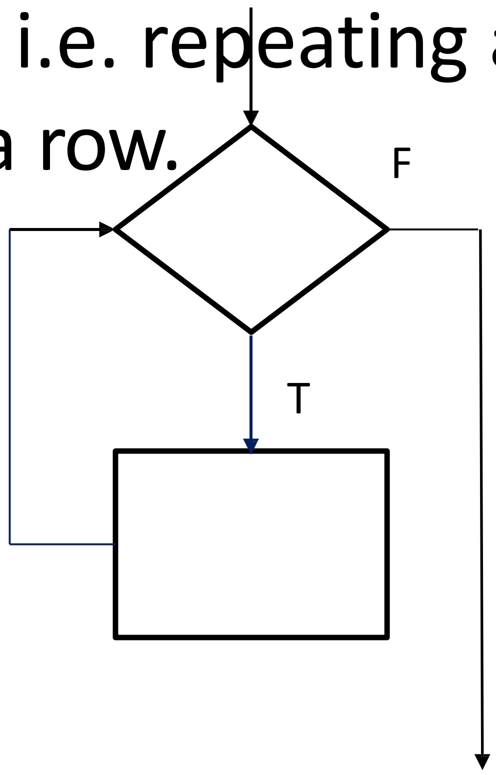
```
if ( x> 0):
   print("number is positive")
else x< 0:
   print("number is negative")
else :
   print ("number is zero")
```

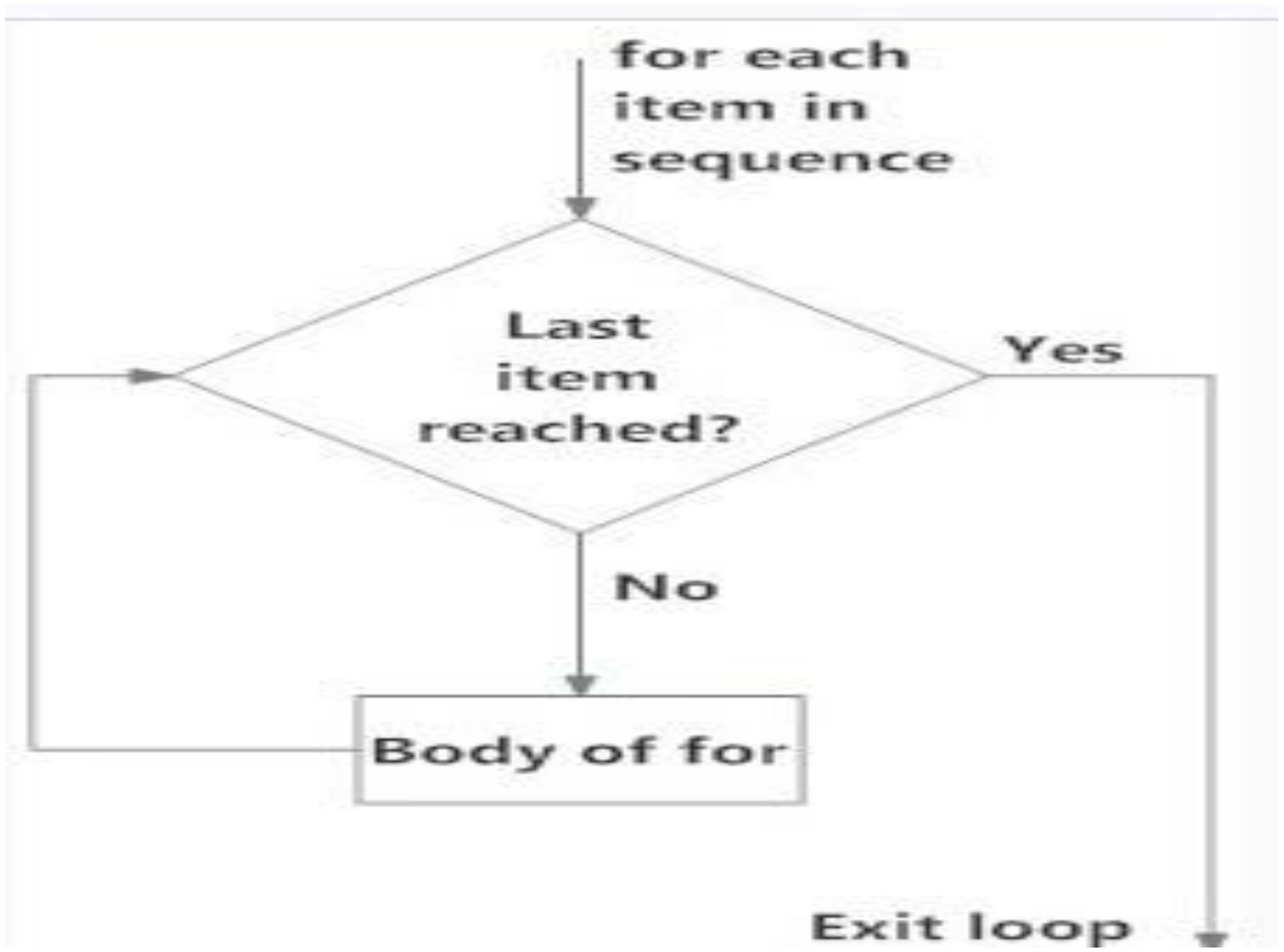
Control Statement

- For repetitive tasks computers are often used to automate it.
- Iteration is repeated execution of a set of statements.
- **There may be a situation when you need to execute a block of code several number of times.**
- **A looping statement allows us to execute a statement or group of statements multiple times.**

Iteration

- Iteration used for looping, i.e. repeating a piece of code multiple times in a row.
  - for loop
  - while loop




for each item in sequence
Last item reached?
Yes
No
Body of for
Exit loop

For loop

Syntax:

```
for item in sequence:
    statements(s)
```

Range

The range()function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops **before a specified number.**

Syntax

range(start, end, step)

#start (default is 0) and step (default is 1)are optional

Example:

x =range(8) -> 0,1,2,3,4,5,6,7x =range(3,8) -> 3,4,5,6,7

Example:

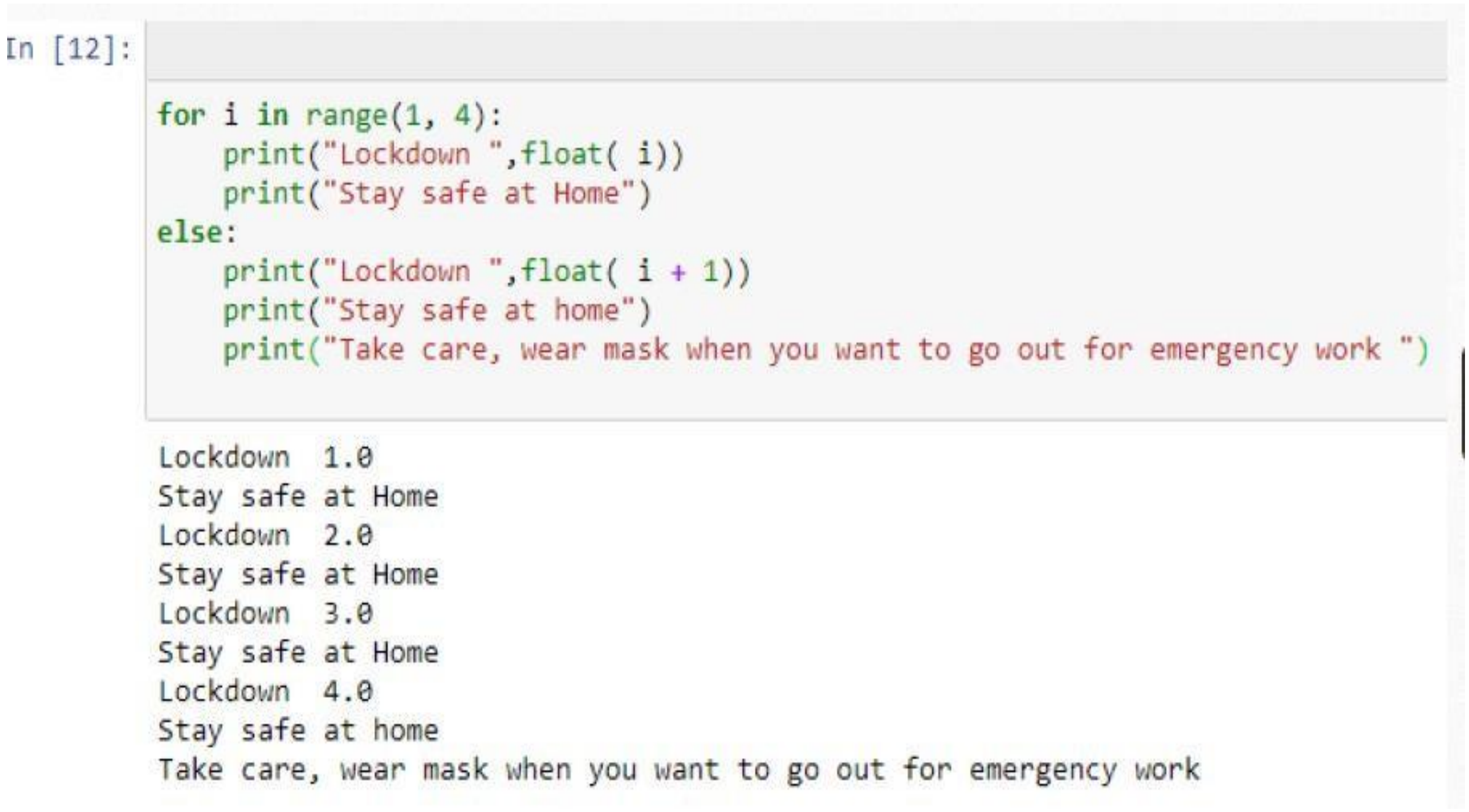
```
lst = [1,2,3,4,8]
for num in lst:
    if num >5:
        print ("list contains numbers greater than 5")
        break
else:
    print("list contains numbers less than 5");
```

Output:

```
list contains numbers greater than 5
```

```
for I in range(1, 4):
    print("Lockdown ",float( i))
    print("Stay safe at Home")
else:
    print("Lockdown ",float( i+ 1))
    print("Stay safe at home")
    print("Take care, wear mask when you want
    to go out for emergency work ")
```

Output



```
In [12]:
for i in range(1, 4):
    print("Lockdown ",float( i))
    print("Stay safe at Home")
else:
    print("Lockdown ",float( i + 1))
    print("Stay safe at home")
    print("Take care, wear mask when you want to go out for emergency work ")
```

```
Lockdown  1.0
Stay safe at Home
Lockdown  2.0
Stay safe at Home
Lockdown  3.0
Stay safe at Home
Lockdown  4.0
Stay safe at home
Take care, wear mask when you want to go out for emergency work
```

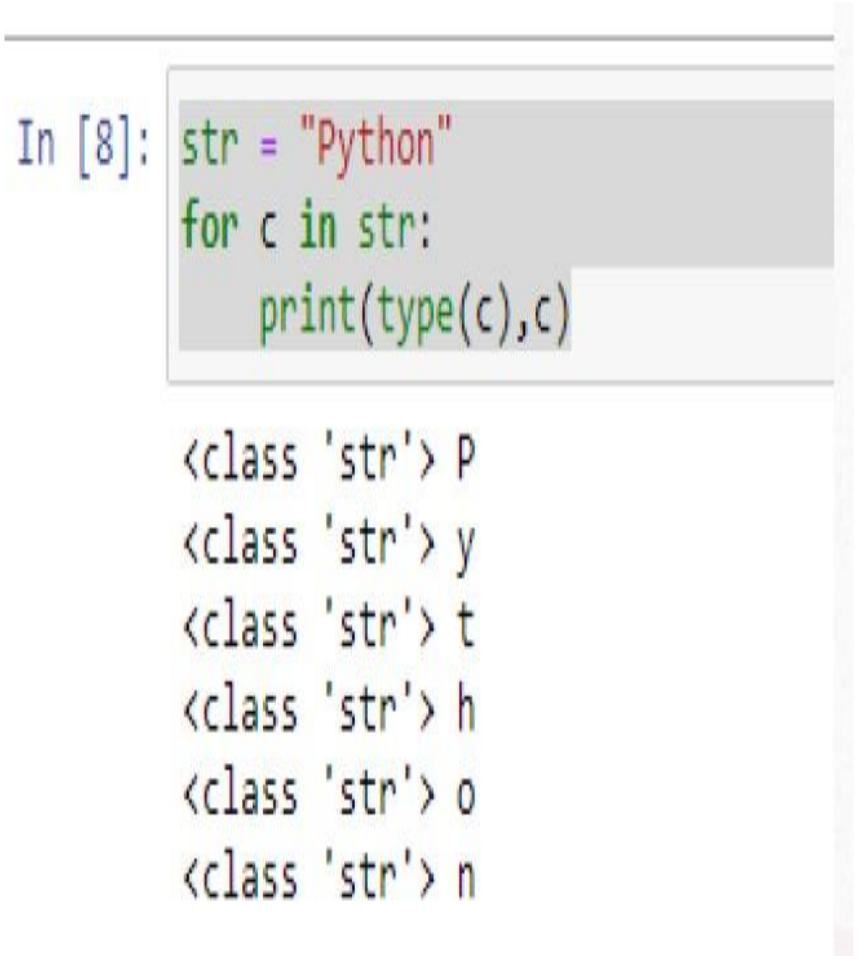
Factorial of no.

```
if num < 0:
    print("Sorry, factorial does not exist for negative
    numbers")
 elif num == 0:
    print("The factorial of 0 is 1")
 else:
    for i in range(1,num + 1):
    factorial = factorial*i
    print("The factorial of",num,"is",factorial)
```

```
n = int(input("How many terms? "))
a, b = 0, 1
count = 0
if n <= 0:
  print("Please enter a positive integer")
elif n == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(a)
else:
  print("Fibonacci sequence:")
  while count < n:
     print(a)
     f = a + b
     # update values
     a = b
     b = f
     count += 1
```

Looping Through a String

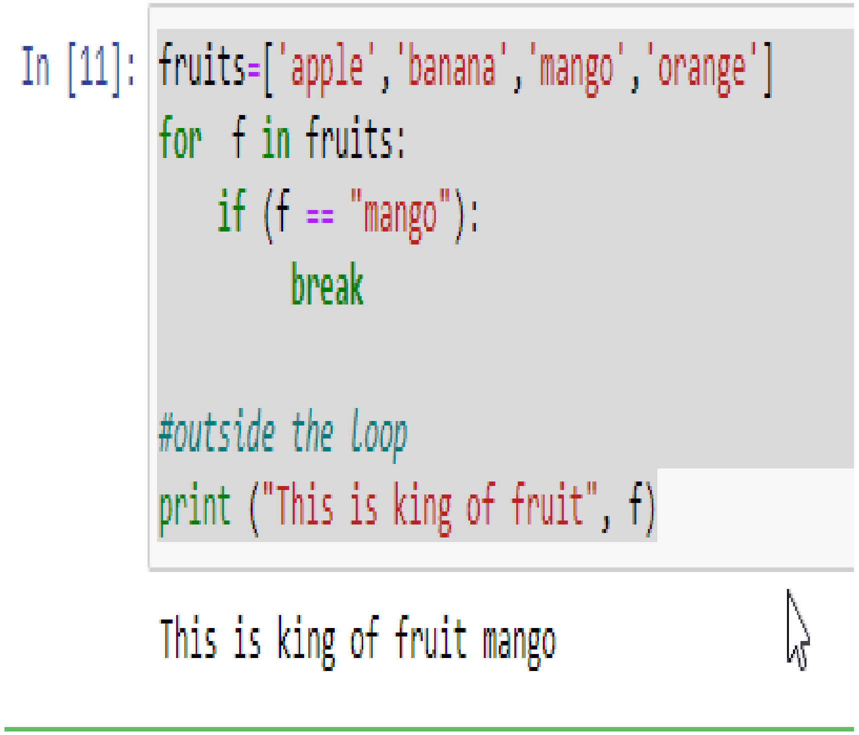
- **Since a string is simply a sequence of characters, theforloop iterates over each character automatically .**



```
In [8]: str = "Python"
        for c in str:
            print(type(c),c)
```

```
<class 'str'> P
<class 'str'> y
<class 'str'> t
<class 'str'> h
<class 'str'> o
<class 'str'> n
```

Break statement



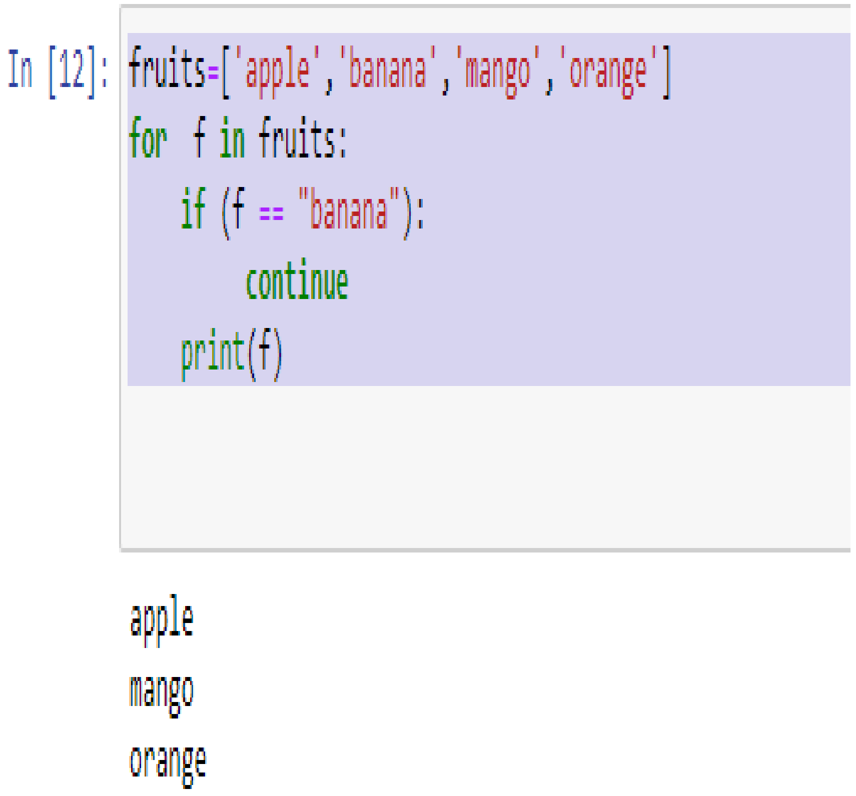
```
In [11]: fruits=['apple','banana','mango','orange']
         for  f in fruits:
             if (f == "mango"):
                 break

         #outside the loop
         print ("This is king of fruit", f)
```

```
This is king of fruit mango
```

- With the break statement we can stop the loop before it has looped through all the items:

Continue

- With the continue statement we can stop the current iteration of the loop, and continue with the next



```
In [12]: fruits=['apple','banana','mango','orange']
         for  f in fruits:
             if (f == "banana"):
                 continue
             print(f)
```

```
apple
mango
orange
```

Looping Through a String

While loop

Syntax

```
while condition:
    #body _of_while
```

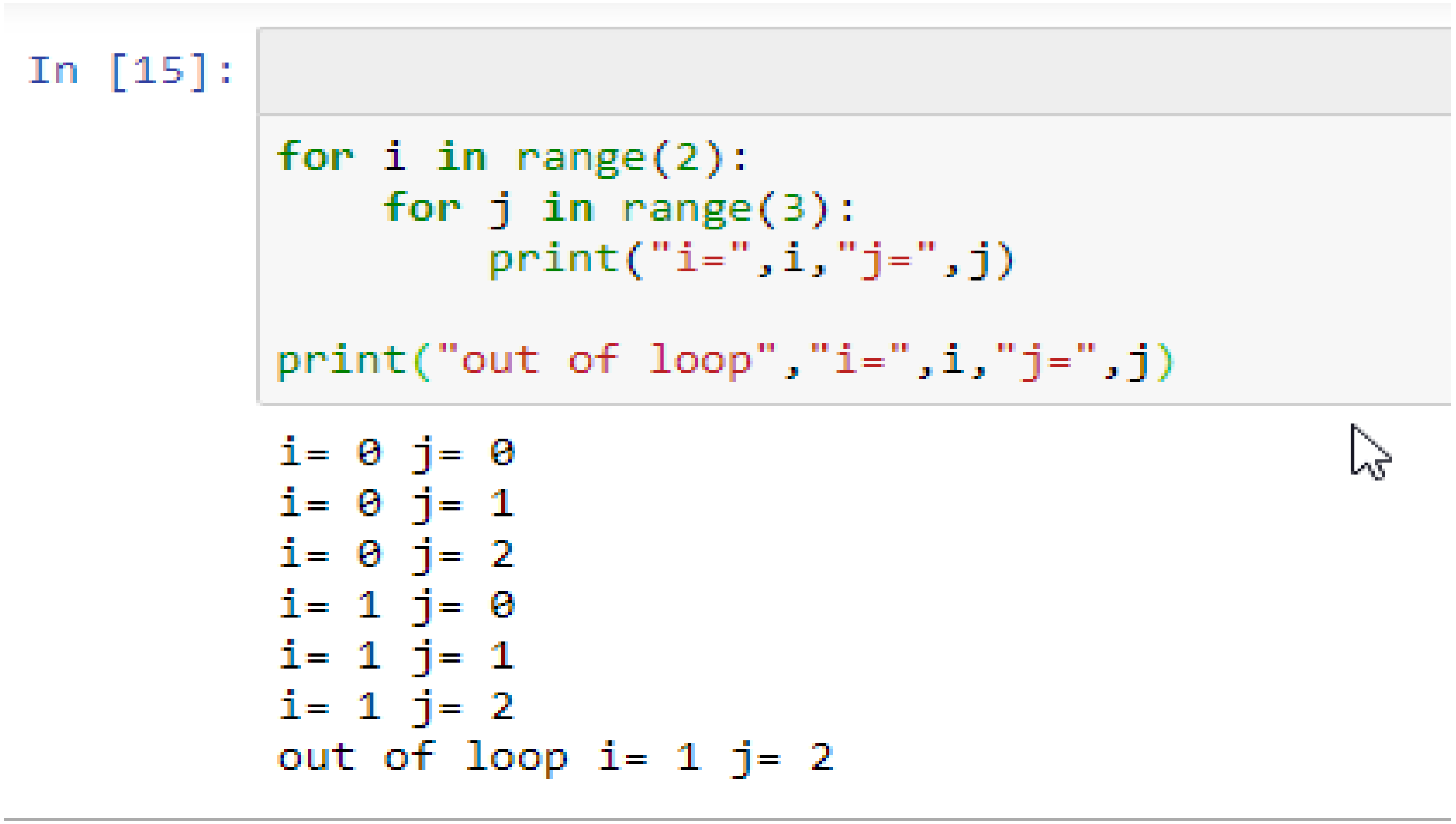
Infinite loop

```
while True:
    print("Hello world")
```

Use of break

```
count = 0
while True:
    print(count)
    count += 1
if count >= 5:
```

Nested loop



```
In [15]:
```

```python
for i in range(2):
    for j in range(3):
        print("i=",i,"j=",j)

print("out of loop","i=",i,"j=",j)
```

```
i= 0 j= 0
i= 0 j= 1
i= 0 j= 2
i= 1 j= 0
i= 1 j= 1
i= 1 j= 2
out of loop i= 1 j= 2
```

Example:

```
num = 1
# loop will repeat itself as long as
# num < 10 remains true
while num < 10:
    print(num)
    #incrementing the value of num
    num = num + 3
```

Output:

```
1
4
7
```

```
t = (‘Hi', ‘hello', ‘I ', ‘am', ‘good‘)
```

Tuples

Tuples

- Create tuples
- Indexing
- Access components
- Slicing
- Built in functions
- Combine multiple tuples

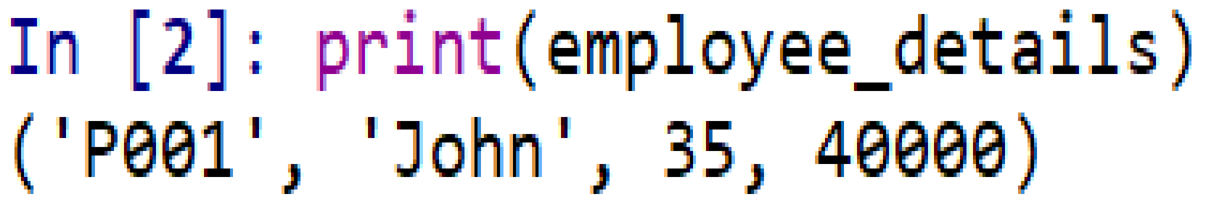
Tuples

•an ordered collection of objects. Tuples are identical to lists in all respects.

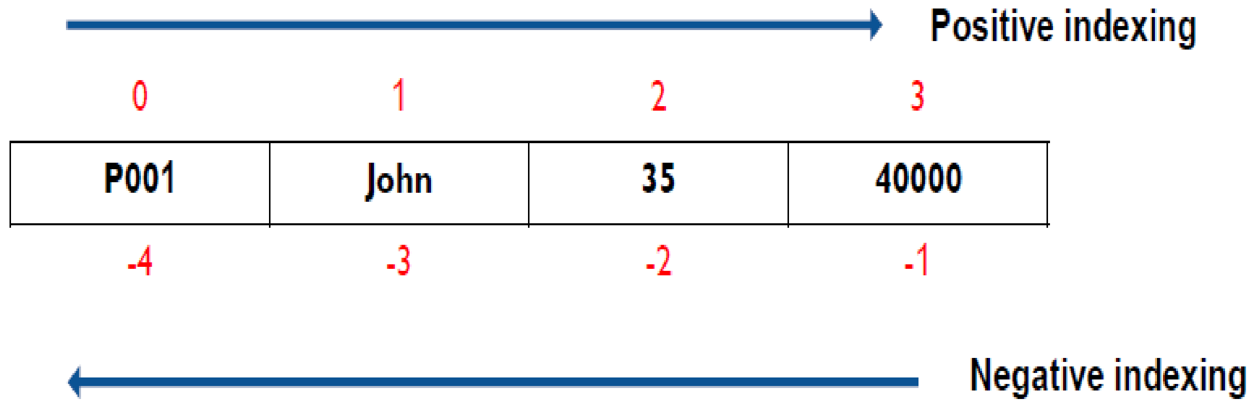
- Tuples are defined by enclosing the elements in parentheses (()) instead of square brackets ([]).
- Allows duplicate members.
- Allow Positive and negative indexing.
- Tuples are immutable.

  Once created they cannot be modified

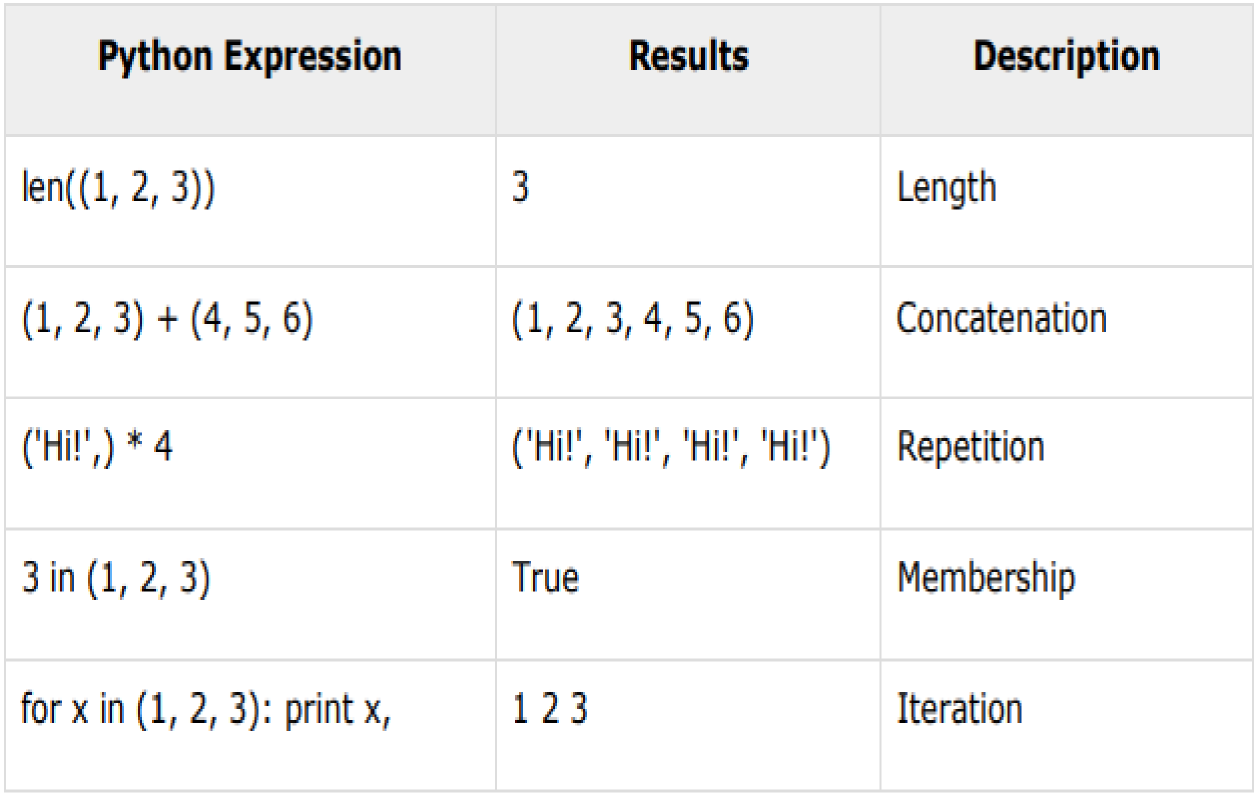
Indexing



```
In [2]: print(employee_details)
('P001', 'John', 35, 40000)
```



Basic Tuples Operations



Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

How to create?

```
t=12,33,"ram",44.5
 t[0]
t1 = ()
print (t1)
t2 = ('tuple', False, 3.2, 1)
print (t2)
```

How to create?

```
tuplex = 4, 7, 3, 8, 1
 print (tuplex)
 tuplex = 4
print (tuplex)
tuplex = tuple()
print (tuplex)
```

Print

```
tuplex = 4, 7, 3, 8, 1
print (tuplex)
print(tuplex[1])
print(tuplex[-1])
print(tuplex[1:3])
```

In

```
tuplex = ("w", 3, "r", "e", "s", "o",
          "u", "r", "c", "e")
print("r" in tuplex)
```

List to tuple

```
listx = [5, 10, 7, 4, 15, 3]
print(listx)
tuplex = tuple(listx)
print(tuplex)
```

Remember

tuples are immutable, so you can not add new elements

using merge of tuples with the + operator you can add an element and it will create a new tuple

Remember - Append

- tuplex = (4, 6, 2,8, 3, 1)

  print(tuplex)
- tuplex = tuplex + (9,)

  print(tuplex)

- t=(10,20,30)
- print(t)
- print(id(t))
- t= t + (5,)
- print(t)
- print(id(t))

Remember - Append

```
listx = list(tuplex)
listx.append(30)
tuplex = tuple(listx)
print(tuplex)
```

```
# Accessing tuple elements using slicing
my_tuple = ('p','r','o','g','r','a','m','i','z')
# elements 2nd to 4th
# Output: ('r', 'o', 'g')
print(my_tuple[1:4])
 # elements beginning to 2nd
# Output: ('p', 'r')
print(my_tuple[:-7])
# elements 8th to end
 # Output: ('i', 'z')
 print(my_tuple[7:])
# elements beginning to end
# Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
```

Slice

```
tuplex = (2, 4, 3, 5, 4, 6, 7, 8, 6, 1)
 _slice = tuplex[3:5]
print(_slice)
_slice = tuplex[:6]
print(_slice)
```

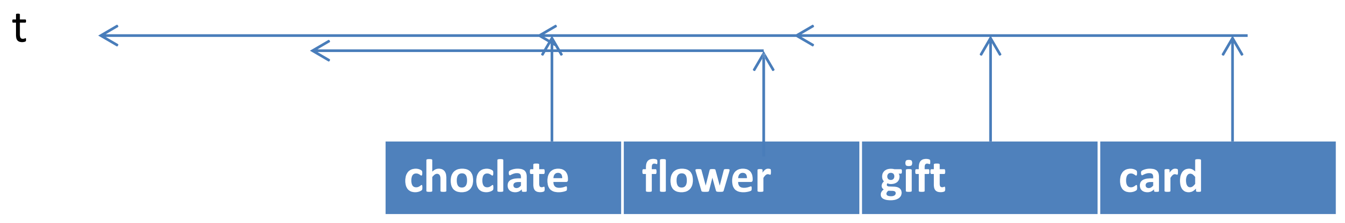
Slice (Cont…)

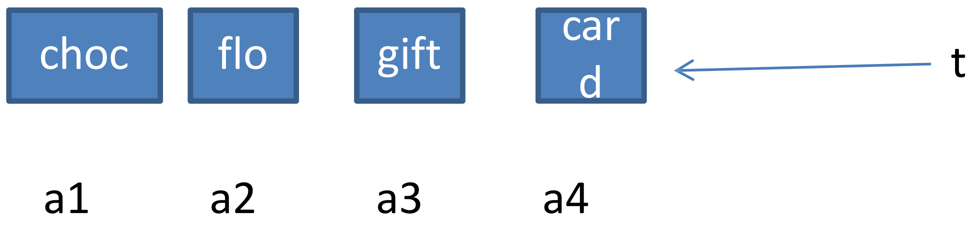
```
_slice = tuplex[5:]
print(_slice)
_slice = tuplex[:]
print(_slice)
_slice = tuplex[-8:-4]
print(_slice)
```

Built-in Tuple Functions

Sr.No.	Function with Description
1	cmp(tuple1, tuple2)Compares elements of both tuples.
2	len(tuple)Gives the total length of the tuple.
3	max(tuple)Returns item from the tuple with max value.
4	min(tuple)Returns item from the tuple with min value.
5	tuple(seq)Converts a list into tuple.

Packing, and Unpacking





Sets

Set

- Create sets
- Modify components
- Set operations
  - Union
  - Intersection
  - Difference
  - Symmetric
  - difference

Set

- Set is a collection of distinct objects
- Do not hold duplicate items
- Stores elements in no particular order
- Created using curly braces { }

SET

- A set is created by placing all the items (elements) inside curly braces {}.
- **separated by comma** or **by using the built-in function set()**.
- It can have any number of items and they may be of different types (integer, float, tuple, string etc.).

But a set cannot have a mutable element, like list, set or dictionary, as its element.

Set is an unordered and unindexed collection of items in Python

Set

A set can be created in two ways.

- First, you can define a set with the built-in set() function:

x = set(<iter>)

Access Items

- You cannot access items in a set by referring to an index or a key.

```
>>> x = (['foo', 'bar', 'baz', 'foo', 'qux'])
>>> x {'qux', 'foo', 'bar', 'baz'}
>>> x = (('foo', 'bar', 'baz', 'foo', 'qux'))
>>> x {'qux', 'foo', 'bar', 'baz'}
```

- To remove an item in a set, use the remove(), or the discard() method.
- **Note:** Show an error if item does not exist
- **Note:** If the item to remove does not exist,
- pop(), method to remove an item, but this method will remove the *last* item. Remember that sets are unordered, so you will not know what item that gets removed.
  discard() will **NOT** raise an error.

Methods For Modifying Sets

x.add(<elem>)

```
>>> x = {'foo', 'bar', 'baz'}
>>> x.add('qux')
>>> x {'bar', 'baz', 'foo', 'qux'}
```

- x.remove(<elem>)
- >>> x = {'foo', 'bar', 'baz'}
- >>> x.remove('baz')
- >>> x
- {'bar', 'foo'}

x.pop()

Removes a random element from a set.

x.pop() removes and returns an arbitrarily chosen element from x. If x is empty, x.pop() raises an exception:

```
my_set.add(5)
```

Duplicate? Using add()?

```
my_set.update([2,3,4])

my_set.update([4,6,7], {1,8,9})
```

my_set.discard(4)

my_set.remove(6)

Then difference?

my_set.pop() my_set.clear()

A = {1, 2, 3, 4, 5}

B = {4, 5, 6, 7, 8}

c=A|B A.union(B)

```
A = {1, 2, 3, 4, 5}

B = {4, 5, 6, 7, 8}

print(A & B)  B.intersection(A)
```

- A = {1, 2, 3, 4, 5}
- B = {4, 5, 6, 7, 8}
- print(A ^ B)

- **A={1,2,3} , B={1,2,3,4,5} , C={1,2,4,5}**
- **print(A.issubset(B))**
- **print(C.issubset(B))**

print(A.issubset(C))

Set Method

```
#Create a new empty set
 x = set()
print(x)
 #Create a non empty
set n = set([0, 1, 2, 3, 4])
print(n)
```

- Shallow copy is a bit-wise copy of an object. A new object is created that has an exact copy of the values in the original object.

- A={1,2,3,4,5} B={1,2,3}
- print(A.issuperset(B))
- True
- print(B.issuperset(A))
- False

- **https://realpython.com/python-beginner-tips/#make-it-stick11 Beginner Tips for Learning Python ProgrammingbyKrishelleHardson-HurleyTable of ContentsMake It StickTip #1: Code EverydayTip #2: Write It OutTip #3: Go Interactive!Tip #4: Take BreaksTip #5: Become a Bug Bounty Hunter**

- **Make It CollaborativeTip**
- **#6: Surround Yourself With Others Who Are Learning**
- **Tip #7: TeachTip**
- **#8: Pair ProgramTip**
- **#9: Ask "GOOD" Questions(G: Give context the problem,O: Outline the things O: Offer your best guess D: Demo)Make SomethingTip #10: Build Something, AnythingTip #11: Contribute to Open**

Python Strings

A string is a sequence of characters.

Ex. 1

“ Hi, how are you? ”

“ 1+1 ”

‘ I ate 4 bananas ‘

“ !@#$%^ & *() ”

Ex.2

```
# all of the following are equivalent
str1 = 'Hello'
print(str1)
str2 = "Hello"
print(str2)
Str3 = '''Hello'''
print(str3)
# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
        the world of Python"""
print(str3)
```

Python Strings

Escape Characters

Following table is a list of escape or non-printable characters that can be represented with backslash notation.

An escape character gets interpreted; in a single quoted as well as double quoted strings.

Backslash notation	Description
\b	Backspace
\n	Newline
\t	Tab

Python Strings

String special operations

Operator	Description
+	Concatenation - Adds values on either side of the operator
*	Repetition - Creates new strings, concatenating multiple copies of the same string
[]	Slice - Gives the character from the given index
[:]	Range Slice - Gives the characters from the given range
In	Membership - Returns true if a character exists in the given string
not in	Membership - Returns true if a character does not exist in the given string
%	Format - Performs String formatting

Python Strings

- **Single quotes, Double quotes, Triple quotes**

- Python string functions are very popular.
- There are two ways to represent strings in python. String is enclosed either with single quotes or double quotes.
- Both the ways (single or double quotes) are correct depending upon the requirement. Sometimes we have to use quotes (single or double quotes) together in the same string, in such cases, we use single and double quotes alternatively so that they can be distinguished.

Example #1:

Check below example and analyze the error –

#Gives Errorprint

('It's python')

Python Strings

Here is a complete list of all the built-in methods to work with strings in Python.
Python has a set of built-in methods that you can use on strings.
Note: All string methods returns new values. They do not change the original string.

Method	Description
capitalize()	Capitalize the first letter of string
endswith()	Returns true if the string ends with the specified value
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet

Python Strings

Method	Description
capitalize()	Capitalize the first letter of string
endswith()	Returns true if the string ends with the specified value
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdigit()	Returns True if all characters in the string are digits

Python Strings

Method	Description
islower()	Returns True if all characters in the string are lower case
isupper()	Returns True if all characters in the string are upper case
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title

Python List

- Python lists are the data structure that is capable of holding different type of data.
- Python lists are mutable i.e., Python will not create a new list if we modify an element in the list.
- Different operation like insertion and deletion can be performed on lists.
- A list can be composed by storing a sequence of different type of values separated by commas.
- A python list is enclosed between square([]) brackets.
- The elements are stored in the index basis with starting index as 0.
- List is one of the most frequently used and very versatile datatype used in Python.

Python List

How to create a list?

In Python programming, a list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.

It can have any number of items and they may be of different types (integer, float, string etc.).

example:

```
List1 =[];                          /* empty list
List2=[1,2,3,4];                    /* List of Integer
List3=*‘x’,‘y’,‘z’+;                /* List of Character
List4=[12.5,11.6];                  /* List of  Float numbers
List5=*‘abc’,‘xyz’+;                /* List of  String
List6=*‘Chirag’,22,2000.3,‘c’+;     /* List  with mixed Data type
```

Python List

Basic List Operations

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership

Python List

Built-in List functions

The following Python functions can be used on lists.

1. len(s)

Returns the number of items in the list.

The len() function can be used on any sequence (such as a string, bytes, tuple, list, or range) or collection (such as a dictionary, set, or frozen set).

Example :

```
a = ["bee", "moth", "ant"]
print(len(a))
```

Output :

```
3
```

Python List

range(stop) or range(start, stop[, step])

Example:

```
print(list(range(10)))
print(list(range(1,11)))
print(list(range(51,56)))
print(list(range(1,11,2)))
```

Output :

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[51, 52, 53, 54, 55]
[1, 3, 5, 7, 9]
```

Python List

Using Lists as stacks and Queues

Example:

```
>>> stack = *‘1’,’2’,’3’+
>>> stack.append(6)
>>> stack.append(7)
>>> print(stack)
Output :
[1, 2, 3, 6, 7]

>>> stack.pop()
Output : 7
>>>print(stack)
 Output : [3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
3
>>>print(stack)
 Output : [1, 2]
```

Python List

Using Lists as stacks and Queues cntnd..

```
# Python code to demonstrate Implementing
# Queue using list
queue = ["A", "B", "C"]
queue.append("Y")
queue.append("Z")
print(queue)
# Removes the first item
print(queue.pop(0))
print(queue)
# Removes the first item
print(queue.pop(0))
print(queue)
```

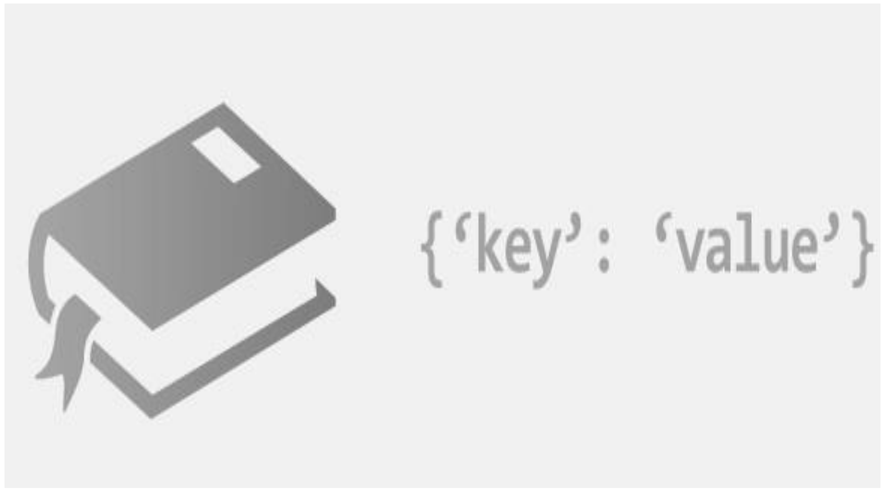
Output:

```
['A', 'B', 'C', 'Y', 'Z']
A
['B', 'C', 'Y', 'Z']
B
['C', 'Y', 'Z']
```

Python Dictionary

What is dictionary in Python?

- Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair.
- Dictionaries are optimized to retrieve values when the key is known.



Python Dictionary

In Python the function dict( ) creates a new dictionary with no items.

```
d1=dict()
print(d1)
{ }
```

Two curly braces with no elements is an empty dictionary.

```
>>>dict
<type 'dict'>
```

Python Dictionary

Creating and accessing values in a dictionary

Example :

```
>>> d=dict({1:'a',2:'b'})
>>> print(d)
{1: 'a', 2: 'b'}
>>> print(d.keys())   #keys() method the find the keys
dict_keys([1, 2])

>>> val=d.values()    #values method to find values
>>> print(val)
dict_values(['a', 'b'])
```

Python Dictionary

Again, The keys are not in any order . To traverse the keys in sorted order one can use the built-in function **sorted():**

```
>>> for key in sorted(shalmali):
            print(key,shalmali[key])
```

```
address pune
age 20
name shalmali
```

Python Dictionary

Properties of Dictionary Keys

Method	Description
clear()	Remove all items form the dictionary.
copy()	Return a shallow copy of the dictionary.
fromkeys(seq[, v])	Return a new dictionary with keys from seq and value equal to v (defaults to None).
get(key[,d])	Return the value of key. If key doesnot exit, return d(defaults to None).
items()	Return a new view of the dictionary's items (key, value).
keys()	Return a new view of the dictionary's keys.
pop(key[,d])	Remove the item with key and return its value or d if key is not found. If d is not provided and key is not found, raises KeyError.
popitem()	Remove and return an arbitary item (key, value). Raises KeyError if the dictionary is empty.
setdefault(key[,d])	If key is in the dictionary, return its value. If not, insert key with a value of d and return d (defaults to None).
values()	Return a new view of the dictionary's values

Python Dictionary

Update() method :

```
>>> d1={1:'a',2:'c'}
>>> d={3:'f'}
>>> d1.update(d)
>>> print(d1)
```

Output:

```
{1: 'a', 2: 'c', 3: 'f'}
```

Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.
- In Python a function is defined using the def keyword:

Example

```
def  my_function():
  print("Hello from a function")
```

Python Functions

- **Calling a Function**

To call a function, use the function name followed by parenthesis:

Example

```
def  my_function():
  print("Hello from a function")

my_function()
```

Output :

```
Hello from a function
```

Python Functions

Scope of variable

Local Scope

- A variable created inside a function is available inside that function:

```
def myfunc():
  x = 300
  print(x)
myfunc()
```

Output:

```
 300
```

Python Functions

Example:

```
list=[1,2,3]
def changeme(list):
  list.append([10,20,30])
  print("values inside function are:",list)
changeme(list)
print("values outside function",list)
```

Output:

```
values inside function are: [1, 2, 3, [10, 20, 30]]
values outside function [1, 2, 3, [10, 20, 30]]
```

Python Functions

```
factorial(3)              # 1st call with 3
3 * factorial(2)          # 2nd call with 2
3 * 2 * factorial(1)      # 3rd call with 1
3 * 2 * 1                 # return from 3rd call as number=1
3 * 2                     # return from 2nd call
6                         # return from 1st call
```

Python Functions

Treatment of Input and Output Arguments

Example:

```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

Output:

```
apple
banana
cherry
```

Python Files & Directories

- In Python programming, File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).
- File is of text file, music File or binary file or video file, All these files are categorized into types

Creation of new text file

```
f= open("hello.txt","w")
```

Python Files & Directories

Modes	Description
R	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
W	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
A	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.

Python Files & Directories

Example 1

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")                    #if already exist gives error
```

Output : a new empty file is created!

Create a new file if it does not exist:

```
f = open("myfile.txt", "w")
```

Example of open method:

```
# Open a file
fobj = open("shalmali.txt", "w")
print ("Name of the file: ", fobj.name)
# Close opened file
fobj.close()
```

Output:

```
Name of the file: shalmali.txt
```

Python Files & Directories

- **Operations on files (open, close, read, write)**

Different File operations can be performed as follows :-

1. Opening an Existing File
open( ) with “w” or “r” or “a” attribute

2.Reading Data from file read( )
or readline( )

3. Writing data into file
write( )

4.Appending Data into file open( )
with “a” attribute

5.Closing a file Close( )

Python Files & Directories

Traversing the current directory using **os.listdir()**

Example:

```
import os
os.listdir('.')
```

Output:

```
['.config', 'sample_data']
```

Python Files & Directories

How to find path of file

```
import os
os.listdir()
os.path.abspath('shalmali.txt')
os.path.realpath('shalmali.txt')
```

Output:

```
/content/shalmali.txt
```

Python Files & Directories

- **Removing Files and Directories**

Syntax of remove( ) Method:

```
            os.remove(file_name)
```

Following is the example to delete an existing file ks.txt:

```
import os
# Delete file a.txt
os.remove(“a.txt")
```

Output:

```
import os
os.remove('a.txt')
f=open('a.txt','r')
```

Output:

```
FileNotFoundError: [Errno 2] No such file or directory: 'a.txt'
```

Python Files & Directories

- **The rmdir( ) Method**

Syntax:
os.rmdir('dirname')

Example:
Import os
# This would remove "/tmp/test" directory.
os.rmdir("/home/ks")

Python Files & Directories

- **Renaming a File**

Syntax of rename() Method:

os.rename(current_file_name, new_file_name)
The rename*()* method takes two arguments, the current filename and the new filename.

```
Import os
# Rename a file from shalmali.txt to kiran.txt
os.rename( "shalmali.txt", “atss.txt" )
```

Output :
Hence shalmali.txt file renamed with atss.txt