

Module - 1

# Introduction to Python with Data Structure



## Units for Discussion

A diagram for Unit 1 consisting of a light blue semi-circular arc with a black outline. Inside the arc, the text 'Python Basics' is centered. Below the arc, a dark blue rectangular box contains the text 'Unit - 1'. Two small white circles are positioned at the base of the arc, one on each side.

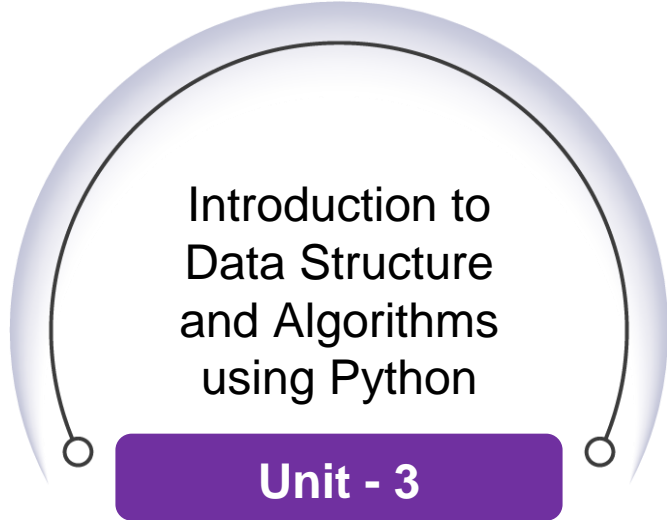
Python Basics

**Unit - 1**

A diagram for Unit 2 consisting of a light red semi-circular arc with a black outline. Inside the arc, the text 'Python Advance' is centered. Below the arc, a dark red rectangular box contains the text 'Unit - 2'. Two small white circles are positioned at the base of the arc, one on each side.

Python Advance

**Unit - 2**

A diagram for Unit 3 consisting of a light purple semi-circular arc with a black outline. Inside the arc, the text 'Introduction to Data Structure and Algorithms using Python' is centered. Below the arc, a dark purple rectangular box contains the text 'Unit - 3'. Two small white circles are positioned at the base of the arc, one on each side.

Introduction to  
Data Structure  
and Algorithms  
using Python

**Unit - 3**

Unit - 1

# Python Basics



# DISCLAIMER

The content is curated from online/offline resources and used for educational purpose only.



Quora

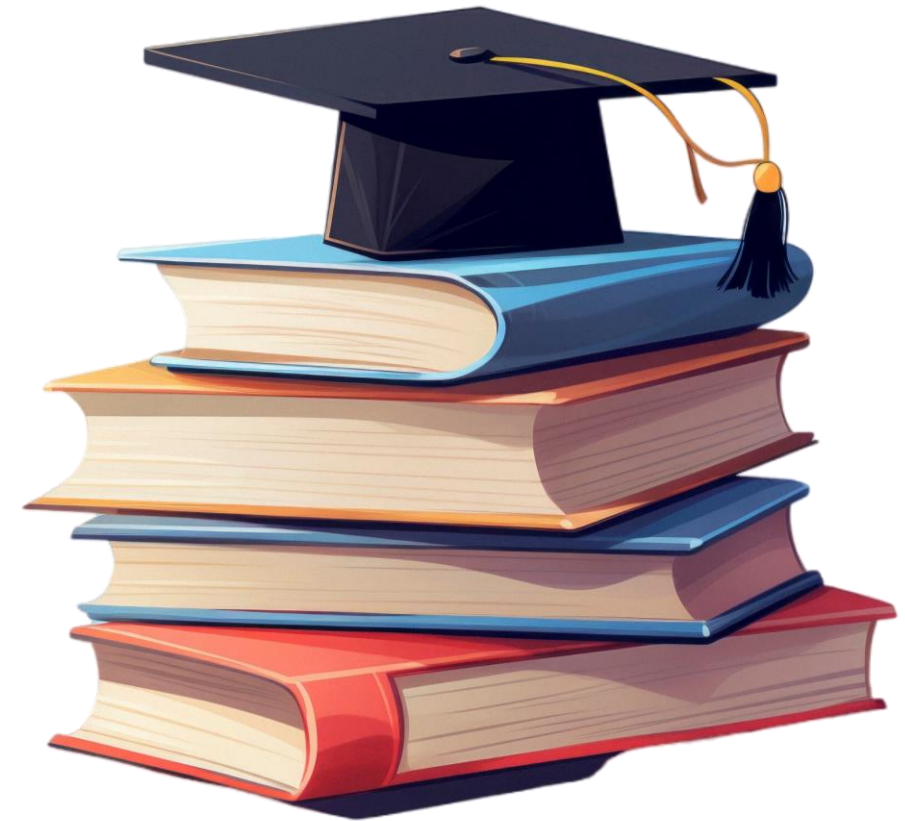


UBER

## Learning Objectives

You'll learn in this unit

- Introduction to Python Programming Language
- Python IDE, different types of IDEs in Python
- Anaconda installation on windows/Linux
- Overview of Jupyter notebook
- Python variables
- Python built-in Data Types
- Operators in Python
- Different Data Structure
- Setting up Virtual Environment



Source :



## How does a computer program work?

- A program makes a computer usable. Without a program, a computer, even the most powerful one, is nothing more than an object. Similarly, without a player, a piano is nothing more than a wooden box.
- Computers are able to perform very complex tasks, but this ability is not innate. A computer's nature is quite different.
- It can execute only extremely simple operations. For example, a computer cannot understand the value of a complicated mathematical function by itself, although this isn't beyond the realms of possibility in the near future.
- Contemporary computers can only evaluate the results of very fundamental operations, like adding or dividing, but they can do it very fast and can repeat these actions virtually any number of times.

## Natural languages vs. programming languages

Aspect	Natural Language	Programming Language
<b>Definition</b>	A language used by humans for everyday communication.	A formal language comprising a set of instructions used to produce various kinds of output, typically used to program computers.
<b>Structure</b>	Flexible, often ambiguous, relies on context and grammar	Rigid, strictly defined syntax and grammar
<b>Usage</b>	Human communication	Writing software, algorithms, and controlling hardware
<b>Complexity</b>	Highly complex with a vast vocabulary and rules	Simpler, with a limited set of keywords and rules
<b>Error Handling</b>	Humans can often infer meaning from context and correct errors	Requires explicit error handling and debugging
<b>Processing</b>	Processed by the human brain	Processed by computers using compilers or interpreters
<b>Error Tolerance</b>	High, humans can understand and adapt to errors	Low, syntax errors must be corrected for the program to run



## Machine language vs. high-level language

### Machine Language

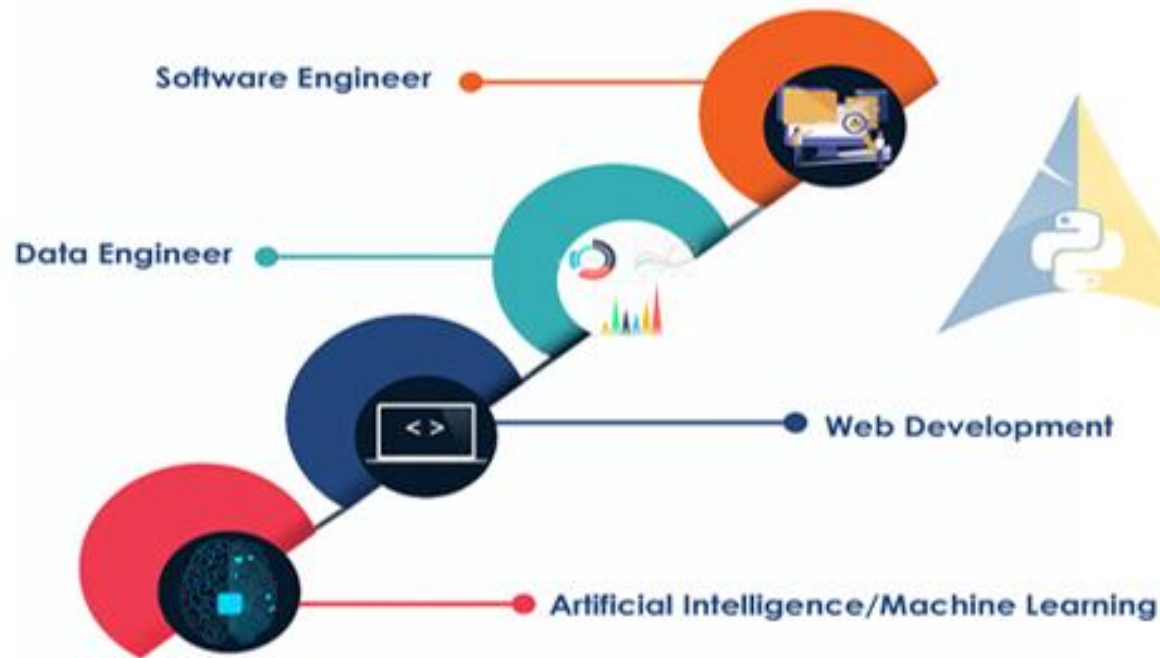
- Alphabet of machine language
- Simplest set of symbols for commands
- Computer's mother tongue

### High- Level Language

- Bridge between humans and computers
- More complex than machine language, simpler than natural language
- Uses readable symbols, words, and conventions
- Enables complex commands
- Source code written in high-level languages
- Source file contains the source code

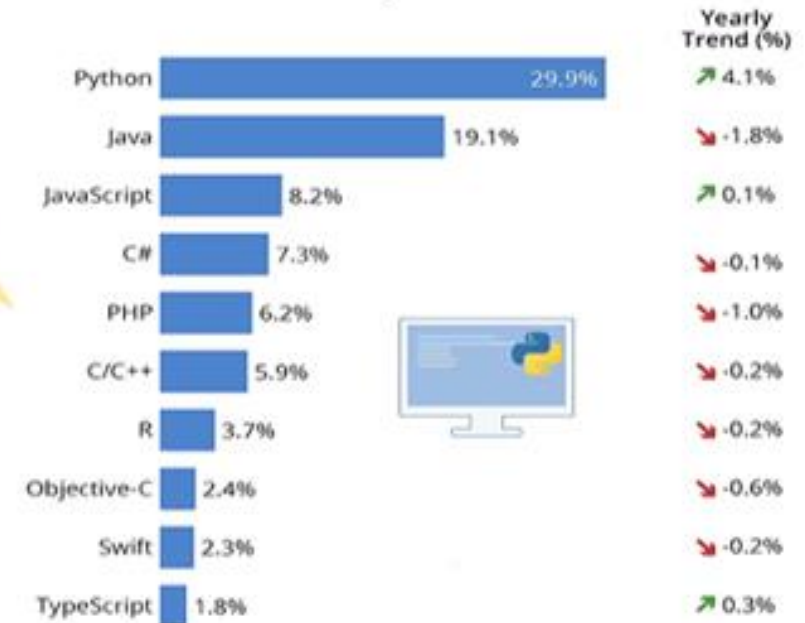
# Why Do We Need To Learn Python?

## Python in various sectors



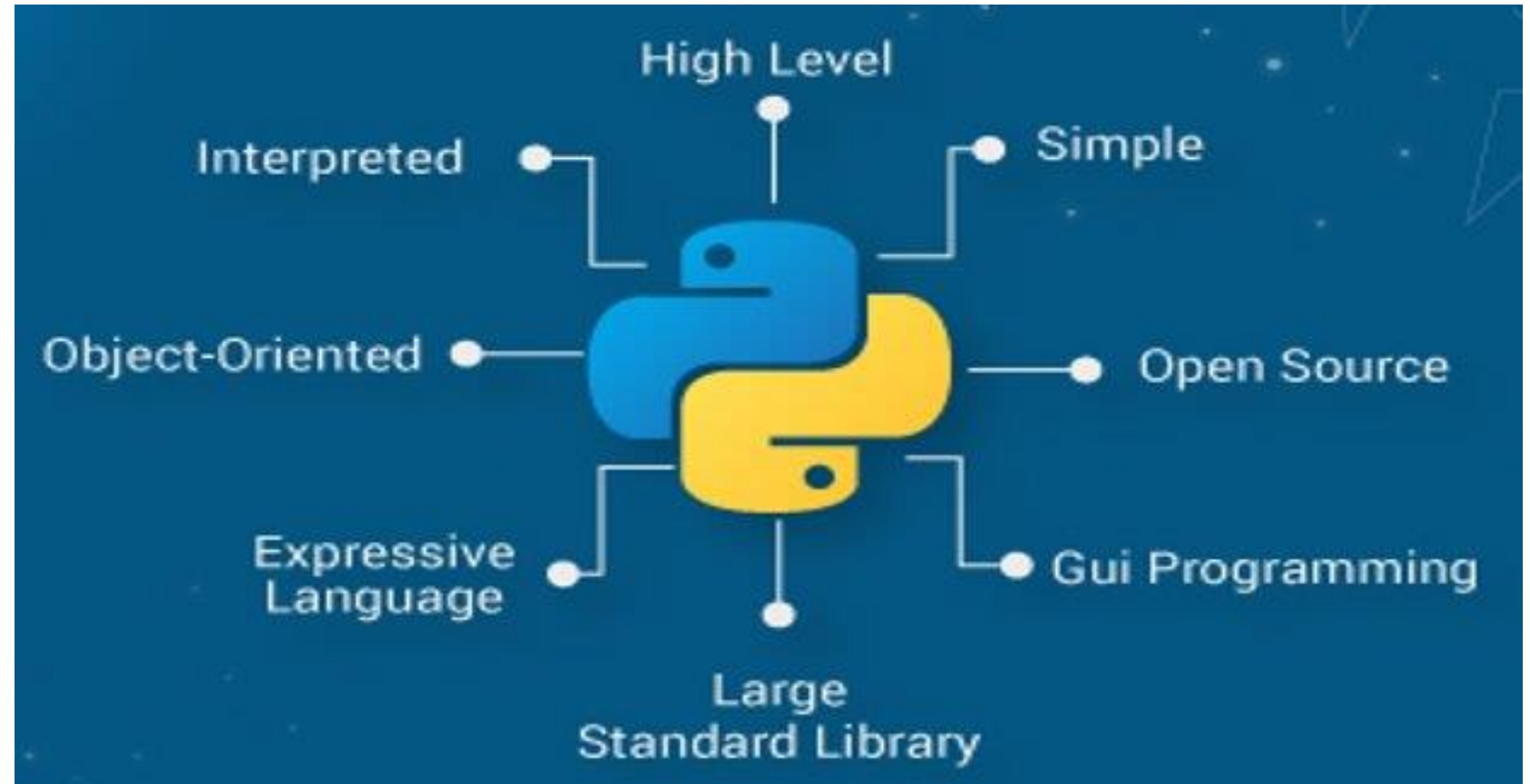
## Python Remains Most Popular Programming Language

Popularity of each programming language based on share of tutorial searches in Google



Source :

## What is Python?



## Interpretation vs Compilation

### Interpretation:

- The process where the source code is translated and executed line by line by an interpreter at runtime, without producing a separate machine code file.
- Examples include Python, JavaScript...

### Compilation:

- The process where the entire source code is translated into machine code before execution. This machine code is typically stored in an executable file.
- Examples include C, C++...



Source :

## Key Differences

### Translation Process:

- Compiled Languages: The compiler translates the whole program at once, creating an executable.
- Interpreted Languages: The interpreter translates and executes code line by line.

### Execution Speed:

- Compiled Languages: Generally faster because the entire program is translated into machine code ahead of time.
- Interpreted Languages: Slower due to line-by-line translation during execution.

### Error Detection:

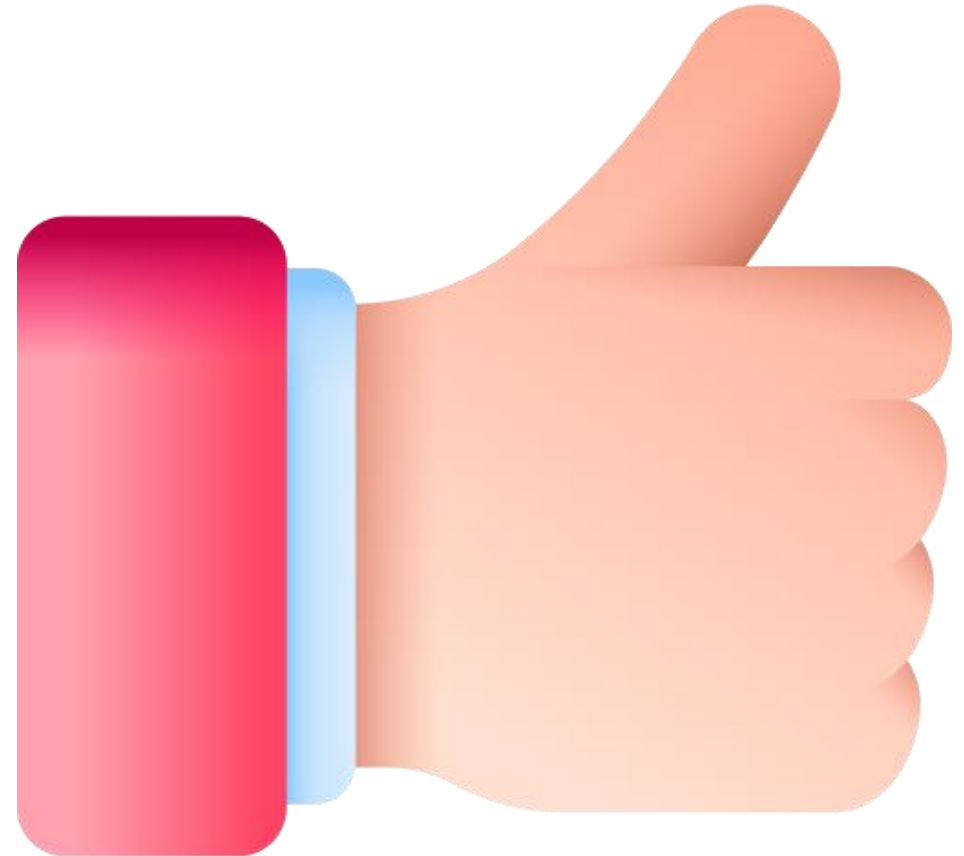
- Compiled Languages: Errors are detected at compile-time, before execution.
- Interpreted Languages: Errors are detected at runtime, potentially causing failures during execution.

### Portability:

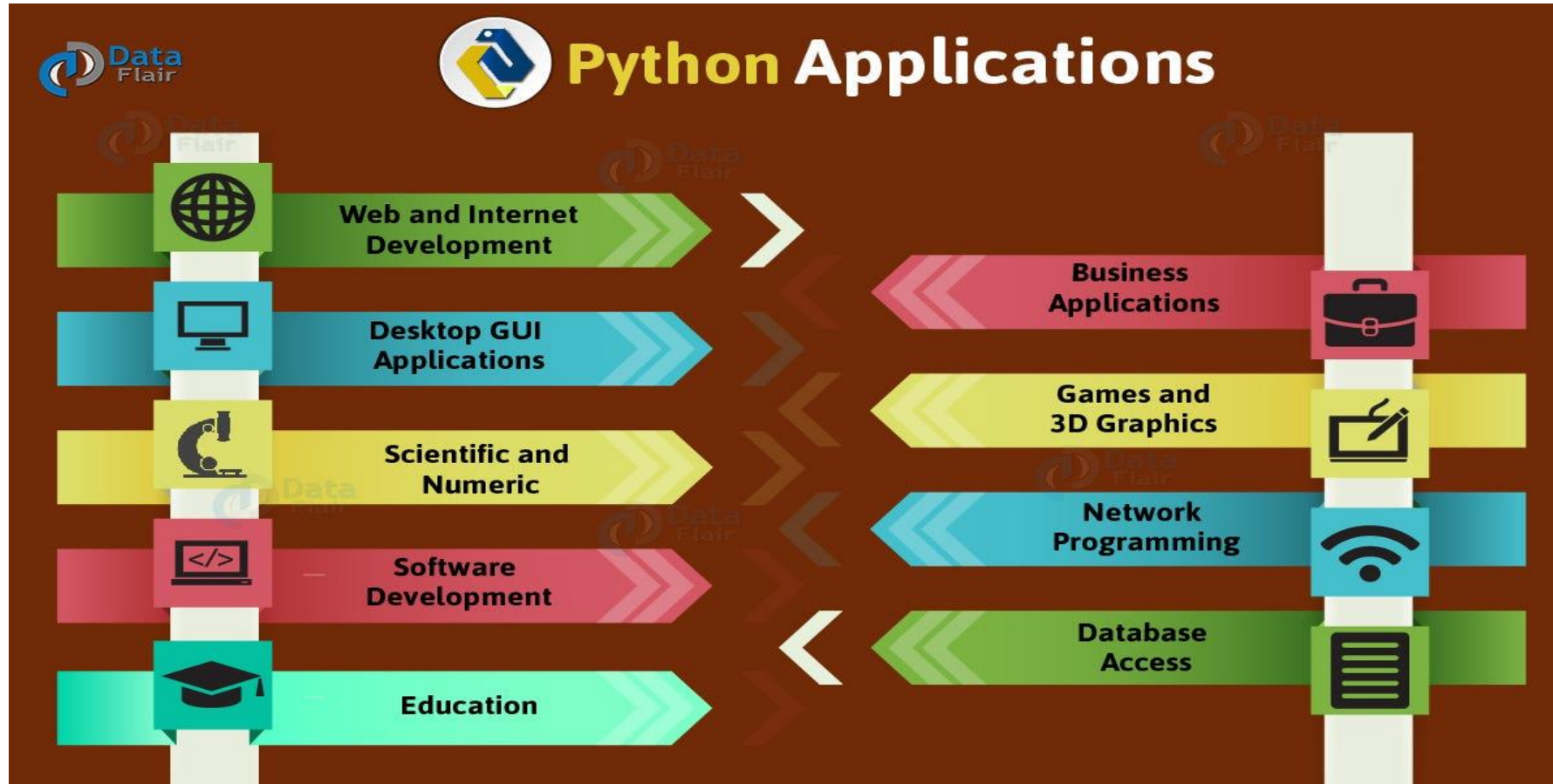
- Compiled Languages: Less portable because the compiled code is platform-specific.
- Interpreted Languages: More portable since the source code is interpreted by platform-specific interpreters.

## Advantages Of Python

- Presence of third-party modules.
- Extensive support libraries
- Open source and large active community base
- Versatile, easy to read, learn and write
- User-friendly data structures (dict, tuple, set, list etc.)
- High-level language
- Dynamically typed language
- Object-oriented and procedural programming language
- Portable and interactive
- Ideal for prototypes – provide more functionality with less coding



# Applications of Python



Source :



## What Is IDE, Different Types Of Ides In Python

IDE stands for Integrated Development Environment. IDE is a software package that consists of several tools for developing and testing the software. A software developer throughout the Software Development Lifecycle has to use many tools such as libraries, editors, compiling and testing tools.

### Features of Python IDEs

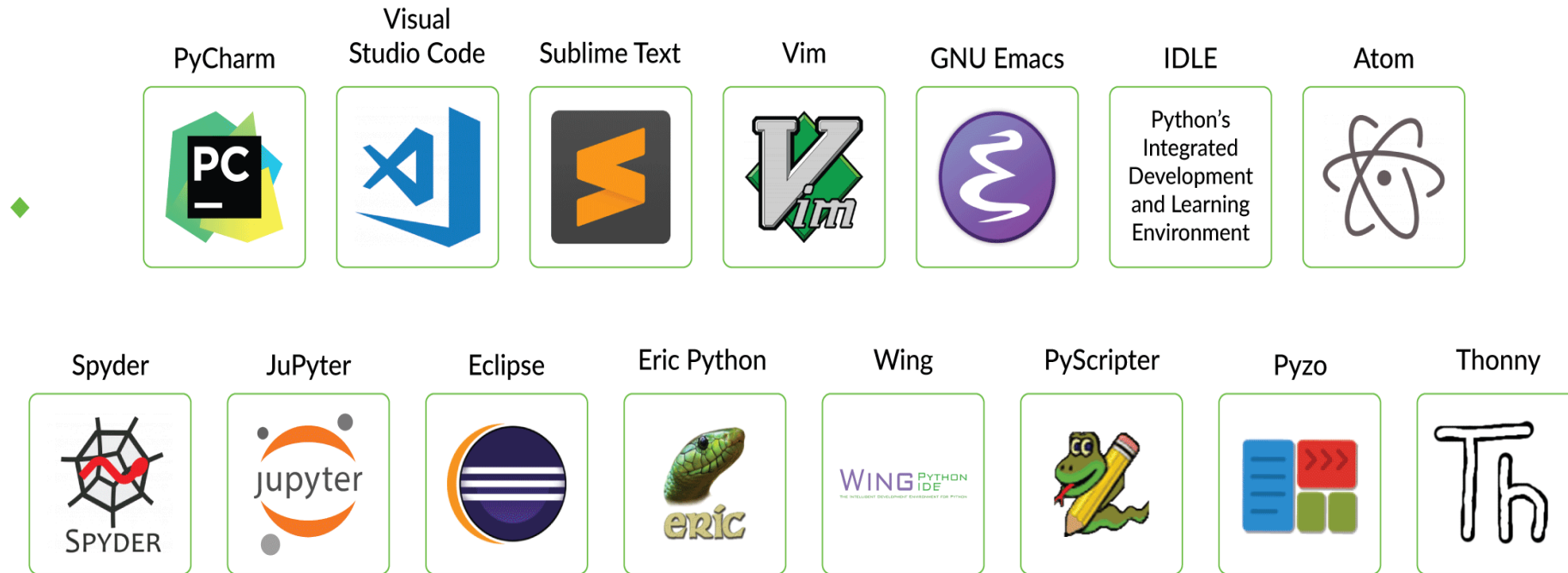
The two distinctive features of Python IDE are given below:

- It provides a plethora of shortcut editing functions that are language-specific.
- It is very user-friendly and fast in terms of functioning.





## Some Popular IDEs For Python



Source :

## Python Setup – Getting Started With Programming

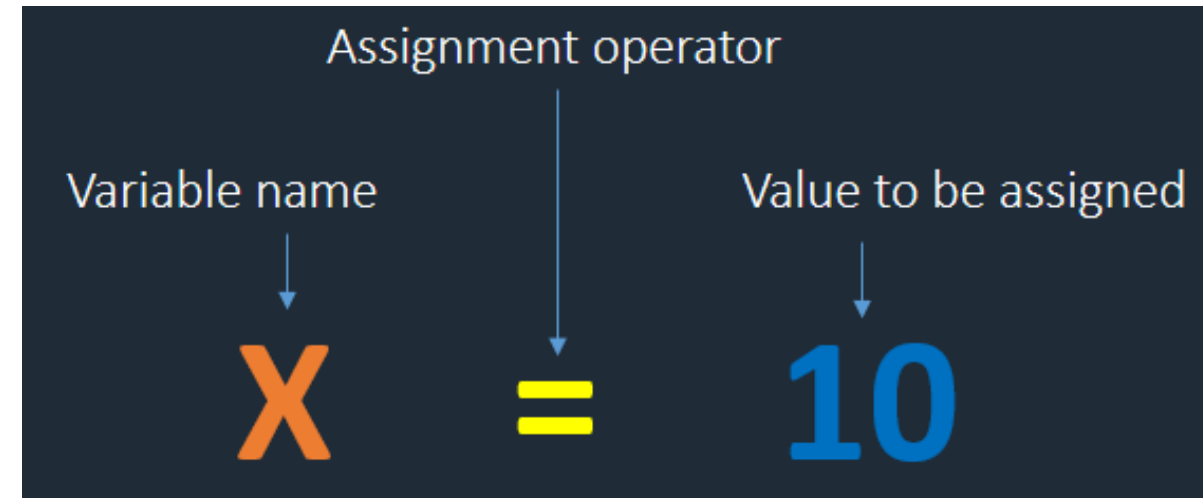


Lab - 1

# Installation of Anaconda Navigator

## Python Variables

- A variable is a container for a value. It can be assigned a name; you can use it to refer to it later in the program. Based on the data type of the variable.
- The interpreter allocates memory and decides what can be stored in the reserved memory.



Source :

## Rules For Variable Names:

### Not allowed naming conventions are:

- Variable name cannot start with numbers (e.g. -> 0\_a, 1\_test )
- White spaces (blanks) cannot be present in the variable name (e.g. -> a b, my variable). The most common practice is to fill the gap with an underscore character.
- Special sign/symbol (“+” and “-”) cannot be used in variable name (e.g. -> a+b , x-y )
- **Python Reserve keyword** cannot be used for naming a variable. (e.g. -> if, for, while etc)
- Variables names are case-sensitive. (e.g.-> a and A is different)

## Assigning Values To Variables

```
count = 100           #Integer Assignment  
kilometers = 1000.0   # Float Assignment  
name = 'Rahul'        # String Assignment
```

```
print(count,kilometers,name)
```

100 1000.0 Rahul

```
a,b,c=1,2,"john"  
print(a,b,c,sep=',')
```

1,2,john

## Keywords In Python

Python reserves 33 keywords in 3.3 versions for its use. Keywords are case sensitive in python. You can't use a keyword as variable name, function name or any other identifier name. Here is the list of keywords in python.

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

Source :

## Comments In Python

Writing the comments in Python is very easy. The comments in Python start with the '#' mark. In python we use three types of comments:

1. **Single line comments**
2. **Inline comments**
3. **Multi-line comments**

```
# This is a Single Line comment:  
print("hello world")
```

hello world

```
# This is inline comment:  
print("Hello world") # this is hello world program
```

Hello world

```
# This is multiline comment:  
# This is comment  
# This is comment  
# This is comment  
print("Hello world")
```

Hello world

Source :

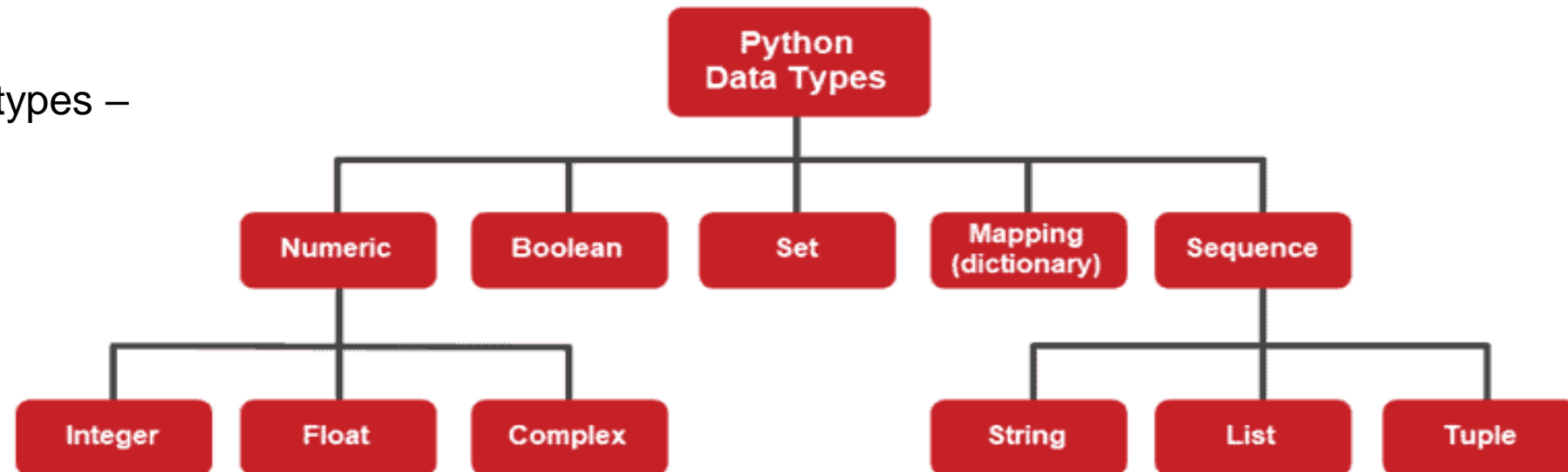


## Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary



Source :

## Standard Data Types

Example	Data Type
<code>x = "Hello World"</code>	str
<code>x = 20</code>	int
<code>x = 20.5</code>	float
<code>x = 1j</code>	complex
<code>x = ["apple", "banana", "cherry"]</code>	list
<code>x = ("apple", "banana", "cherry")</code>	tuple
<code>x = range(6)</code>	range
<code>x = {"name" : "John", "age" : 36}</code>	dict
<code>x = {"apple", "banana", "cherry"}</code>	set
<code>x = frozenset({"apple", "banana", "cherry"})</code>	frozenset

## Input() in Python

- In Python, Using the input() function, we take input from a user, and using the print() function, we display output on the screen.
- Using the input() function, users can give any information to the application in the strings or numbers format.

```
# take three values from user
name = input("Enter Employee Name: ")
salary = input("Enter salary: ")
company = input("Enter Company name: ")

# Display all values on screen
print("\n")
print("Printing Employee Details")
print("Name", "Salary", "Company")
print(name, salary, company)
```

## Python Numbers

- Number data types store numeric values. Number objects are created when you assign a value to them.

```
a = 10
```

```
a
```

```
10
```

```
# Multiple Assignment
```

```
b,c,d = 11,12,13
```

```
print(a,b,c,d)
```

```
10 11 12 13
```

## Complex numbers

- A complex number consists of an ordered pair of real floating-point numbers denoted by  $x + yj$ , where  $x$  and  $y$  are the real numbers and  $j$  is the imaginary unit.

```
In [18]: a=32.3e18  
         type(a)
```

```
Out[18]: float
```

```
In [19]: b=3.14j  
         type(b)
```

```
Out[19]: complex
```

```
In [20]: print(a,b,sep='\n')
```

```
3.23e+19
```

```
3.14j
```

## Python Strings

- 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.
- Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 to the end.
- The plus (+) sign is the string concatenation operator and the asterisk (\*) is the repetition operator.

```
str1 = 'Python is a programming language'
```

```
print(str1)
```

```
Python is a programming language
```

```
print(str1[0])
```

```
P
```

```
print(str1[2:10])
```

```
thon is
```

```
print(str1[6:18])
```

```
is a progra
```

## String formatting

- Strings in Python can be formatted with the use of format() method which is very versatile and powerful tool for formatting of Strings.
- Format method in String contains curly braces {} as placeholders which can hold arguments according to position or keyword to specify the order.
- A string can be left (<), right (>) or center(^) justified with the use of format specifiers, separated by colon (:). Integers such as Binary, hexadecimal, etc. and floats can be rounded or displayed in the exponent form with the use of format specifiers.

```
# Default order
String1 = "{} {} {}".format('Python', 'Programming', 'Language')
print("Print String in default order: ")
print(String1)
```

```
Print String in default order:
Python Programming Language
```

## String formatting

### Escape sequence:

\n – starts printing since the next line.

```
# Positional Formatting
String1 = "{1} {0} {2}".format('Python', 'Programming', 'Language')
print("\nPrint String in Positional order: ")
print(String1)
```

```
Print String in Positional order:
Programming Python Language
```

```
# Keyword Formatting
String1 = "{a} {e} {c}".format(a='Python', e='Programming', c='Language')
print("\nPrint String in order of Keywords: ")
print(String1)
```

```
Print String in order of Keywords:
Python Programming Language
```

## F-String

- The idea behind f-strings is to make string interpolation simpler, to create an f-string, prefix the string with the letter “f”.
- The string itself can be formatted in such the same way that you would with str.format(). F-strings provide a concise and convenient way to embed python expressions inside string literals for formatting.

### Example:

```
# program on f-string
val = 'Prince'
print(f"{val} is the student of my class.")

name = 'Ravi'
age = 23
print(f"Hello, My name is {name} and I'm {age} years old.")
```

```
Prince is the student of my class.
Hello, My name is Ravi and I'm 23 years old.
```



There are some basic string methods for various built-in data types:

methods	Description
<b>upper()</b>	Convert to upper case
<b>lower()</b>	Convert to lower case
<b>title()</b>	Convert the first letter of each word to caps
<b>capitalize</b>	Convert first letter to caps
<b>swapcase()</b>	Toggle case
<b>isalpha()</b>	Return true if alphabet
<b>isdigit()</b>	Return true if digit
<b>isupper()</b>	Return true if upper case string
<b>islower()</b>	Return true if string in title case
<b>istitle()</b>	Returns true if lower case string
<b>strip()</b>	Remove blank space from the both sides
<b>split()</b>	Break the string into list of words
<b>join()</b>	Join two string with given separator

## Example:

```
s="I love my India "  
print("Actual String : ",s)  
print("Length is : ",len(s))  
s=s.strip()  
print("Stripped String : ",s)  
print("Length is : ",len(s))  
print("Upper Case : ",s.upper())  
print("Lower Case : ",s.lower())  
print("Title Case : ",s.title())  
print("Capitalize Case : ",s.capitalize())  
t=s.split()  
print(t)  
k=" ".join(t)  
print(k)
```

```
Actual String : I love my India  
Length is : 17  
Stripped String : I love my India  
Length is : 15  
Upper Case : I LOVE MY INDIA  
Lower Case : i love my india  
Title Case : I Love My India  
Capitalize Case : I love my india  
['I', 'love', 'my', 'India']  
I love my India
```

## Different Types Of Operators In Python

Python divides the operators in the following groups:

Arithmetic operators:	<code>+, -, *, /, %, **, //</code>
Assignment operators:	<code>=, +=, -=, *=, /=, %=, //=, **=</code> etc
Comparison operators:	<code>==, <u>!=</u>, &gt;, &lt;, &gt;=, &lt;=</code> etc.
Logical operators:	<code>and, <u>or</u>, not</code> .
Identity operators:	<code>is, is not</code> .
Membership operators:	<code>in, not in</code> .
Bitwise operators:	<code>&amp;,  , ^, ~, &lt;&lt;</code> etc

## Arithmetic operator

### Example:

```
# addition, sub, mul, div: +, -, *, / etc  
x = 5  
y = 3  
  
print(x + y)
```

8

## Assignment operator

### Example: 1

```
x = 5  
x += 3  
  
print(x)
```

8

### Example: 2

```
x = 5  
x *= 3  
  
print(x)
```

15

## Comparison operator

### Example: 1

```
x = 5
y = 3

print(x == y)

# returns False because 5 is not equal to 3
```

False

### Example: 2

```
x = 5
y = 3

print(x != y)

# returns True because 5 is not equal to 3
```

True

## Logical Operator

### Example:

```
x = 5  
  
print(x > 3 and x < 10)  
  
# returns True because 5 is greater than 3 AND 5 is less than 10
```

True

```
x = 5  
  
print(x > 3 or x < 4)  
  
# returns True because one of the conditions are true (5 is greater than 3, but 5 is not less than 4)
```

True

Lab - 2

# Perform Basic Operations of Python Programming



## Python List

- List are mutable datatypes(changeable).
- Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

### Syntax:

```
List1 = [ "Physics", "Chemistry", "Maths"]
```



## Python List

- For example – Create a list and perform slicing, update operations

```
L=[1,2,3,"abes",12.5]
```

```
L
```

```
[1, 2, 3, 'abes', 12.5]
```

```
L[0:3]
```

```
[1, 2, 3]
```

```
L[0]=10
```

```
L
```

```
[10, 2, 3, 'abes', 12.5]
```

## Python List

- **Split method:**

```
# write a program to take input a sentence and split it into word
```

```
text = input("Enter the word: ")
```

```
Enter the word: I love Python the most
```

```
text.split()
```

```
['I', 'love', 'Python', 'the', 'most']
```

## Python List

- Concatenation:

```
list1 = [ 'abcd', 786 , 2.23, 'john', 70.2 ]  
small = [123, 'john']
```

```
print(small * 2)           # Prints list two times  
print(list1 + small)      # Prints concatenated lists
```

```
[123, 'john', 123, 'john']  
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

## Tuples

- Tuples are Immutable datatypes(unchangeable)
- A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated. Tuples can be thought of as read-only lists.
- For example – Create a tuple and prints slicing, concatenation etc., operations.

```
tuple1 = ( 'abcd', 786 , 2.23, 'john', 70.2 )  
tuple2 = (123, 'john')
```

```
print(tuple1)                # Prints the complete tuple  
  
( 'abcd', 786, 2.23, 'john', 70.2)
```

# Tuples

```
print(tuple1[0])          # Prints first element of the tuple
```

abcd

```
print(tuple1[1:3])        # Prints elements of the tuple starting from 2nd till 3rd
```

(786, 2.23)

```
print(tuple1[2:])         # Prints elements of the tuple starting from 3rd element
```

(2.23, 'john', 70.2)

```
print(tuple2 * 2)         # Prints the contents of the tuple twice
```

(123, 'john', 123, 'john')

```
print(tuple1 + tuple2)    # Prints concatenated tuples
```

('abcd', 786, 2.23, 'john', 70.2, 123, 'john')

---

## Tuples

As we discussed tuple is not changeable that's why we can't update anything in tuple

```
tuple1 = ( 'pqrs', 123 , 2.14, 'python', 98.6 )  
list1 = [ 'pqrs', 123 , 2.14, 'python', 98.6 ]
```

```
tuple1[2] = 1000    # Invalid syntax with tuple  
list1[2] = 1000     # Valid syntax with list
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-67-ea927ce875cf> in <module>  
----> 1 tuple1[2] = 1000    # Invalid syntax with tuple  
      2 list1[2] = 1000     # Valid syntax with list
```

```
TypeError: 'tuple' object does not support item assignment
```

## Python Dictionary

- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs.
- A dictionary key can be almost any Python type but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- **For example** – Create the blank dictionary named dict1 and dict2. Update the element in dict1 and dict2, also print the keys and values from dict1 and dict2.

### #Syntax of dictionary

dict = {"Keys" : "values"}

```
dict1 = {}  
dict1['one'] = "This is Python"  
dict1[2] = "This is Java"
```

```
print(dict1)
```

```
{'one': 'This is Python', 2: 'This is Java'}
```

```
dict2 = {'name': 'rohit', 'id': 1234, 'dept': 'technical'}
```



## Python Dictionary

- For example –

```
print(dict1['one'])      # Prints value for 'one' key
```

This is Python

```
print(dict1[2])          # Prints value for 2 key
```

This is Java

```
print(dict2)             # Prints complete dictionary
```

```
{'name': 'rohit', 'id': 1234, 'dept': 'technical'}
```

```
print(dict2.keys())      # Prints all the keys
```

```
dict_keys(['name', 'id', 'dept'])
```

```
print(dict2.values())    # Prints all the values
```

```
dict_values(['rohit', 1234, 'technical'])
```

## Python Set

Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces {}.

- Unordered
- Unchangeable
- Not allowed duplicate values
- Example- For duplicates:

```
a = {1,2,3,4,5,1,1,2,3,4}  
print(a)
```

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

## Type casting

In Python, Type Casting is a process in which we convert a literal of one type to another. Inbuilt functions `int()`, `float()` and `str()` shall be used for typecasting.

- `int()` can take a float or string literal as argument and returns a value of class 'int' type.
- `float()` can take an int or string literal as argument and returns a value of class 'float' type.
- `str()` can take a float or int literal as argument and returns a value of class 'str' type.
- For example:

```
#integer
n = 100

#float
f = float(n)
print(f)
print(type(f))
```

```
100.0
<class 'float'>
```

```
#string
s = str(n)
print(s)
print(type(s))
```

```
100
<class 'str'>
```

## Slicing in Python

### Indexing in Python

- Indexing is the process of accessing an element in a sequence using its position in the sequence (its index).
- In Python, indexing starts from 0, which means the first element in a sequence is at position 0, the second element is at position 1, and so on.
- To access an element in a sequence, you can use square brackets [] with the index of the element you want to access.

#### example:

```
my_list = ['apple', 'banana', 'cherry', 'date']
```

```
print(my_list[0]) # output: 'apple'
```

```
print(my_list[1]) # output: 'banana'
```

## LISTS - INDEXING AND SLICING

List Slicing

```
a[2:5]
```

'spam'	'egg'	'bacon'	'tomato'	'ham'	'lobster'
0	1	2	3	4	5
List Indices					

Source :

## Slicing in Python

Slicing is the process of accessing a sub-sequence of a sequence by specifying a starting and ending index. In Python, you perform slicing using the colon : operator.

**The syntax for slicing is as follows:**

```
sequence[start_index:end_index]
```

where start\_index is the index of the first element in the sub-sequence and end\_index is the index of the last element in the sub-sequence (excluding the element at the end\_index). To slice a sequence, you can use square brackets [] with the start and end indices separated by a colon.

**For example:**

```
my_list = ['apple', 'banana', 'cherry', 'date']  
print(my_list[1:3]) # output: ['banana', 'cherry']
```

Lab - 3

## Perform Different Data Structures of Python Programming

## Setting up Virtual Environment in Anaconda

### What is a Virtual Environment?

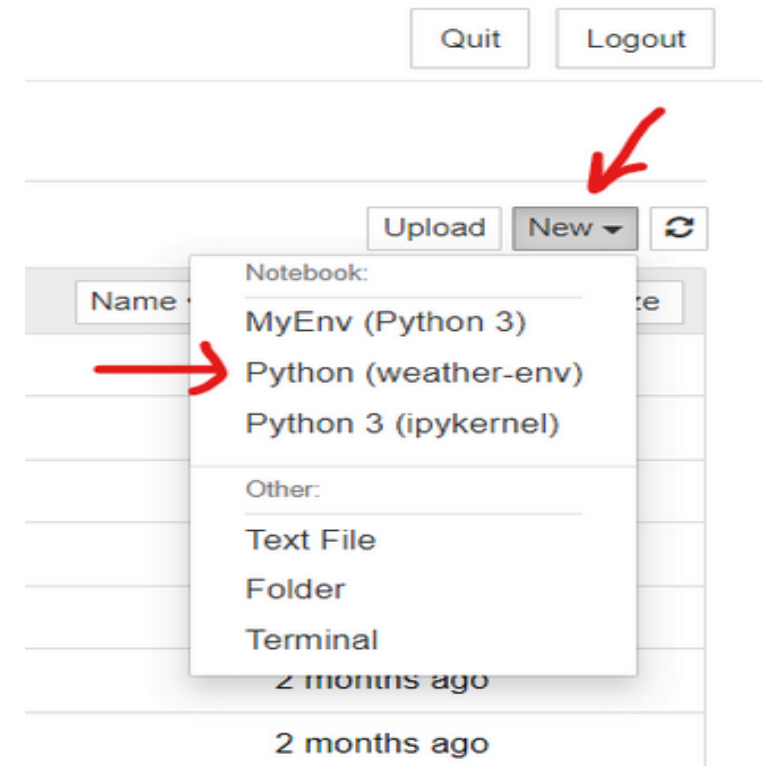
- A **virtual environment** is an isolated environment that allows you to manage dependencies for a specific project without interfering with other projects.
- It ensures that each project can have its own set of dependencies, regardless of what dependencies every other project has.

### Benefits

- Isolation: Separate dependencies for each project.
- Consistency: Avoids version conflicts.
- Reproducibility: Easy to replicate environments.

### Key Tools

- virtualenv: Create isolated Python environments.
- ipykernel: Use virtual environments as Jupyter kernels.



Lab - 4

## Setting up Virtual Environment



## Conclusion

- Python is a High-level, interpreted, object-oriented language and the most popular programming language used in many real-time applications.
- Lists, Dictionaries, Tuple and Sets are different types of python data structure.
- Python is also an object-oriented language, and it allows us to develop applications using an Object-Oriented approach.
- Python has various modules that helps in connecting and communicating with databases.
- List are mutable(Changeable) and tuples are immutable(Unchangeable).
- Set is a collection which is unordered, unchangeable and unindexed. Not allows the duplicate members.
- Python is dynamically typed language that means we do not need to define the type of variable during assigning like: `int a = 2`
- Extensive support of libraries.



## Reference

- <https://www.edureka.co/blog/variables-and-data-types-in-python/>
- [https://www.tutorialspoint.com/python/python\\_variable\\_types.htm](https://www.tutorialspoint.com/python/python_variable_types.htm)
- <https://data-flair.training/blogs/python-variables-and-data-types/>
- <https://www.programiz.com/python-programming/variables-datatypes>
- <https://www.tutorialspoint.com/numpy>
- <https://numpy.org/>
- [https://www.tutorialspoint.com/python\\_data\\_science](https://www.tutorialspoint.com/python_data_science)
- <https://www.geeksforgeeks.org/generating-random-number-list-in-python/>
- [https://www.w3schools.com/python/numpy/numpy\\_random.asp](https://www.w3schools.com/python/numpy/numpy_random.asp)
- <https://www.geeksforgeeks.org/numpy-arrays-in-python/>
- <https://data-flair.training/blogs/numpy-statistical-functions/>
- <https://www.w3schools.in/python-tutorial/decision-making/>
- <https://www.geeksforgeeks.org/python-broadcasting-with-numpy-arrays/>
- <https://pythonexamples.org/python-type-casting/>

## Reference

- <https://realpython.com/>
- [https://www.tutorialspoint.com/python/python\\_variable\\_types.htm](https://www.tutorialspoint.com/python/python_variable_types.htm)
- [https://www.tutorialspoint.com/python/python\\_variable\\_types.htm](https://www.tutorialspoint.com/python/python_variable_types.htm)
- <https://www.edureka.co/blog/variables-and-data-types-in-python/>
- <https://data-flair.training/blogs/python-variables-and-data-types/>
- <https://www.programiz.com/python-programming/variables-datatypes>
- [Jupyter Notebook: An Introduction – Real Python](#)
- <https://net-informations.com/python/iq/how.htm>
- <https://docs.python.org/3/faq/general.html#what-is-python>



Let's Start

## Quiz

1. What will be the output of following code ?

```
i = 1  
while True:  
    if i%3 == 0:  
        break  
    print(i)  
  
    i += 1
```

- a) 1 2 3
- b) Error
- c) 1 2
- d) none of the mentioned

**Answer: B**

Error



## Quiz

**2. What is the method inside the class in python language?**

- a) Object
- b) Function
- c) Attribute
- d) Argument



**Answer: B**  
Function

## Quiz

3. Amongst which of the following is / are the application areas of Python programming?

- a) Web Development
- b) Game Development
- c) Artificial Intelligence and Machine Learning
- d) All of the above

**Answer: D**

All of the above



## Quiz

4. What are the values of the following Python expressions?

$2^{(3^2)}$   
 $(2^3)^2$   
 $2^3^2$

- a) 512, 64, 512
- b) 512, 512, 512
- c) 64, 512, 64
- d) 64, 64, 64

**Answer: A**  
512, 64, 512





## Quiz

5. What will be the output of the following Python code?

- `l = [1, 0, 2, 0, 'hello', '', []] list(filter(bool, l))`
- a) `[1, 0, 2, 'hello', '', []]`
- b) Error
- c) `[1, 2, 'hello']`
- d) `[1, 0, 2, 0, 'hello', '', []]`



**Answer: C**  
**[1, 2, 'hello']**

# Thank You