

One Step In Changing Education Chain...

**RED & WHITE**<sup>®</sup>  
GROUP OF INSTITUTES

## DEMO MATERIAL



**CORE  
PYTHON**

## Table of Contents

1.	<b>INTRODUCTION TO PYTHON.....</b>	<b>3</b>
■	WHAT IS PYTHON LANGUAGE?.....	3
■	HISTORY OF PYTHON.....	3
■	PYTHON VERSIONS .....	4
■	INSTALLATION OF PYTHON .....	5
■	PYTHON SHELL & IDLE .....	11
■	FIRST PYTHON PROGRAM / TYPES OF MODE TO RUN PROGRAM.....	13
2.	<b>FUNDAMENTALS OF PYTHON .....</b>	<b>18</b>
■	I/O (INPUT/OUTPUT) FUNCTIONS.....	18
■	DATA-TYPES IN PYTHON.....	20
■	VARIABLES IN PYTHON .....	21
■	OPERATORS IN DART.....	22
■	TYPE CASTING CONSTRUCTORS .....	24
■	id() & type() FUNCTIONS .....	26
3.	<b>STRING IN DETAIL.....</b>	<b>28</b>
■	STRING DATATYPE .....	28
■	STRING FORMATTING.....	29
■	STRING MANIPULATION.....	33
■	TYPES OF COMMENTS .....	34

#1

## INTRODUCTION TO PYTHON

### WHAT IS PYTHON LANGUAGE?

1. पायथन एक प्रोग्रामिंग language है जिसका उपयोग web-development, software development, mathematics और system scripting के लिए किया जाता है।
2. पायथन english भाषा जैसी सरल syntax अपनाता है जिससे यह language सीखने में बहुत ही सरल पड़ती है।
3. पायथन की सरल syntax की वजह से developers दूसरी कोई प्रोग्रामिंग languages की बराबरी में बहुत ही कम lines में code लिख सकते हैं।
4. पायथन interpreter system पे रन होती है, यानि की जैसे जैसे code लिखाता जाता है वैसे तुरंत ही वो code रन भी होता जाता है। इस तरह prototyping बहुत ही तेज़ हो सकता है।
5. पायथन को procedural और object-oriented ऐसे दोनों तरीकों से उपयोग में लिया जा सकता है।
6. पायथन scope बनाने के लिए indentation (white space) पर आधार रखता है, जैसे की loops, functions और classes के scope। और कोई प्रोग्रामिंग language इस हेतु के लिए curly-brackets { } का उपयोग करती है।

### HISTORY OF PYTHON

1. पायथन का implementation CWI, Netherland December 1989 में Guido Van Rossum द्वारा शरू किया गया था।



GUIDO VAN ROSSUM

2. February 1991 में Guido Van Rossum ने पायथन का पहला code, version 0.9.0 publish किया था।
3. पायथन language, जो ABC प्रोग्रामिंग language पर से बानी हुई language है जो की अपने आप में Exception handling और Amoeba operating system के साथ compatible थी।
4. पायथन basically 2 प्रोग्रामिंग languages पर से influenced हुई है:
  - A. ABC language.
  - B. Modula-3
5. 1994 में पायथन version 1.0 को रिलीज़ किया गया था।
6. हाल में पायथन का latest version 3.8 चल रहा है।

## PYTHON VERSIONS

1. पायथन के basically 2 versions main गिने जाते हैं:

**Python 2.x** और **Python 3.x**

`x = 1,2,3,...`

2. बहुत समय से चल रहे Python 2.x की lifecycle का end 2020 में already हो चूका है। यानि की अब से Python 2.x के लिए कोई भी प्रकार का update या maintenance support नहीं किया जाने वाला।
3. Python 3.x हल में पायथन का latest और fast version है।

Python Version	Release Date
Python 1.0	January 1994
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
<b>Python 2.7</b>	July 3, 2010
Python 3.0	December 3, 2008
<b>Python 3.8</b>	October 14, 2019

## INSTALLATION OF PYTHON

### A. INSTALLATION PROCESS ON MACOS

**STEP - 1**

First go to this link: <https://www.python.org/>

**STEP - 2**

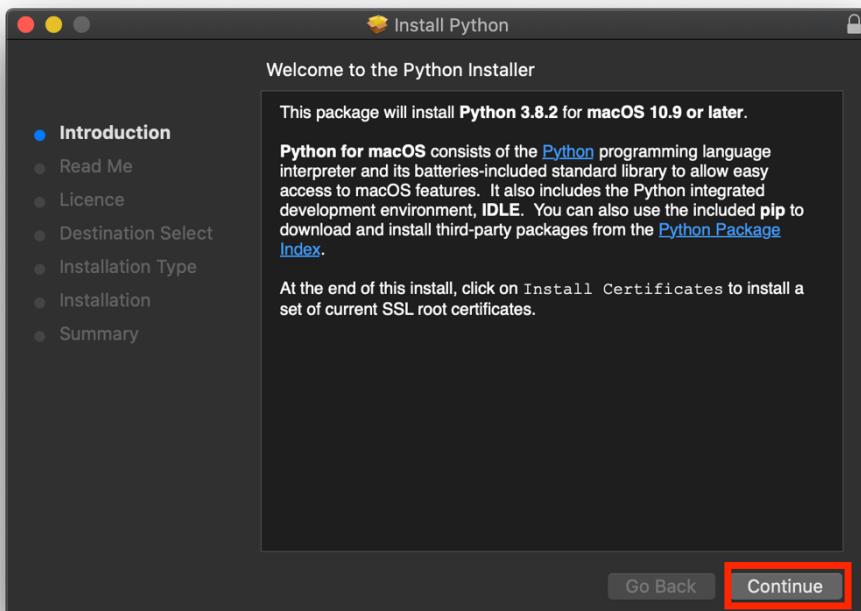
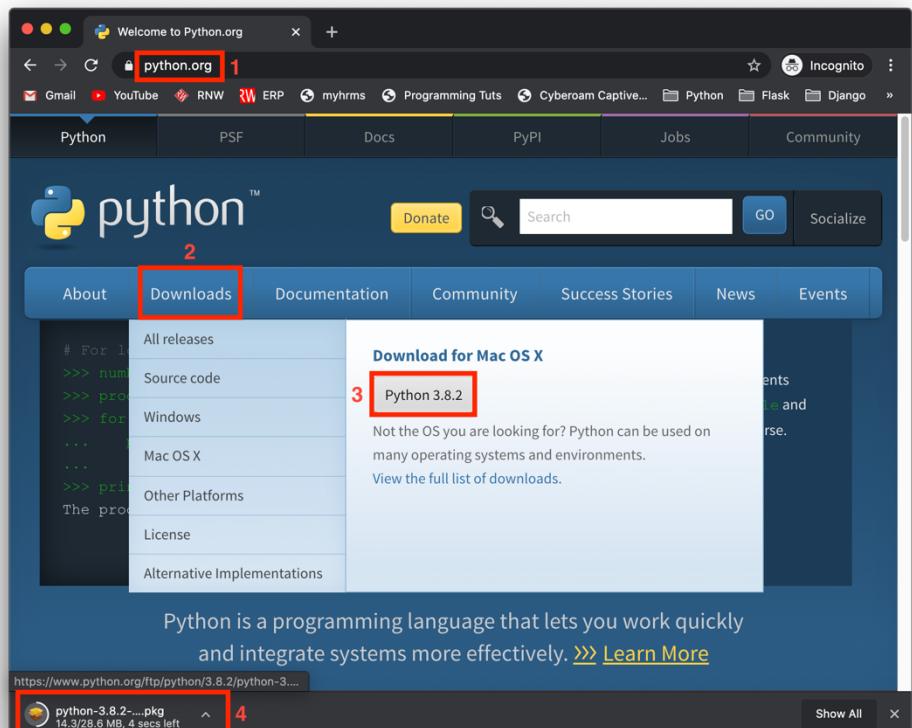
Hover on **Downloads** button from navigation bar.

**STEP - 3**

Click on **Python 3.8.2** button.

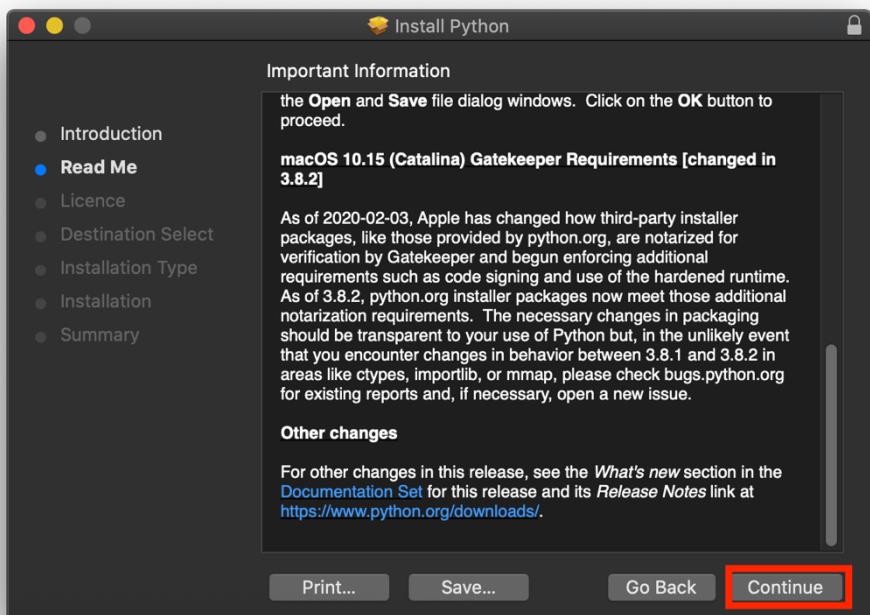
**STEP - 4**

Download will start.

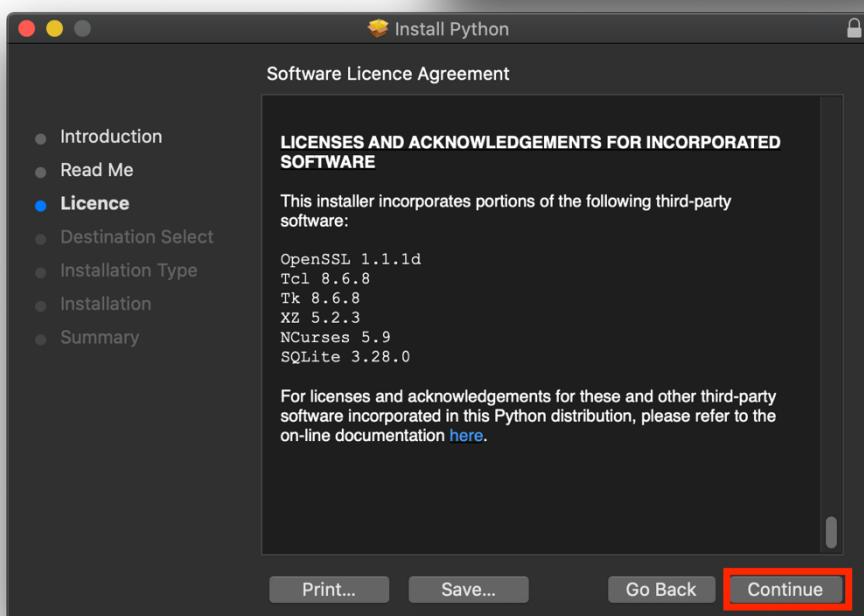


**STEP - 5**

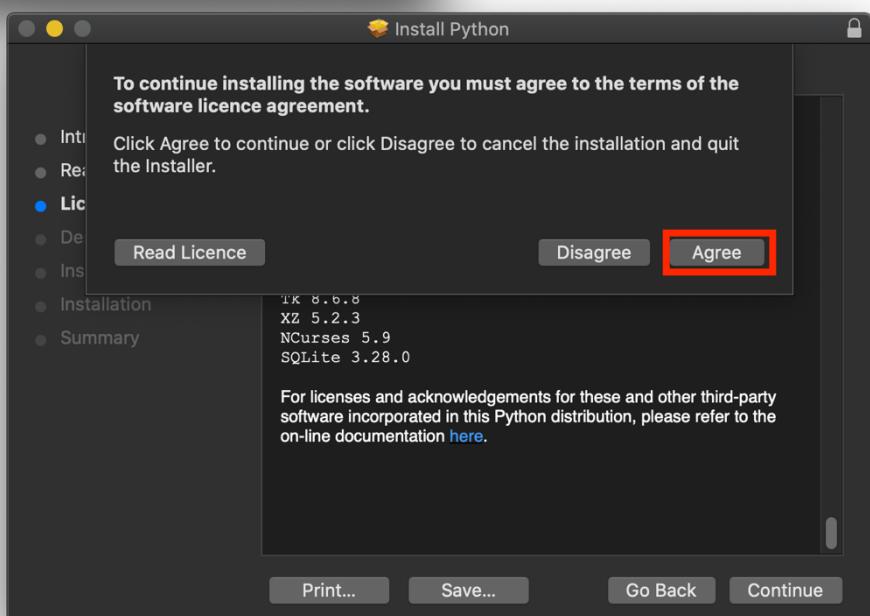
Now open downloaded file and click on **Continue** button.



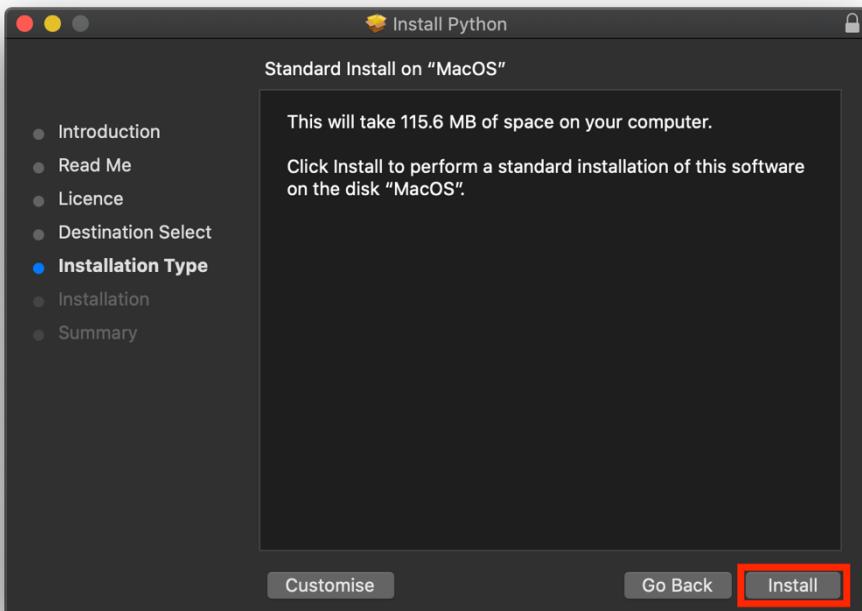
Click on **Continue** button again.



After reading all licence agreement, click on **Continue** button.

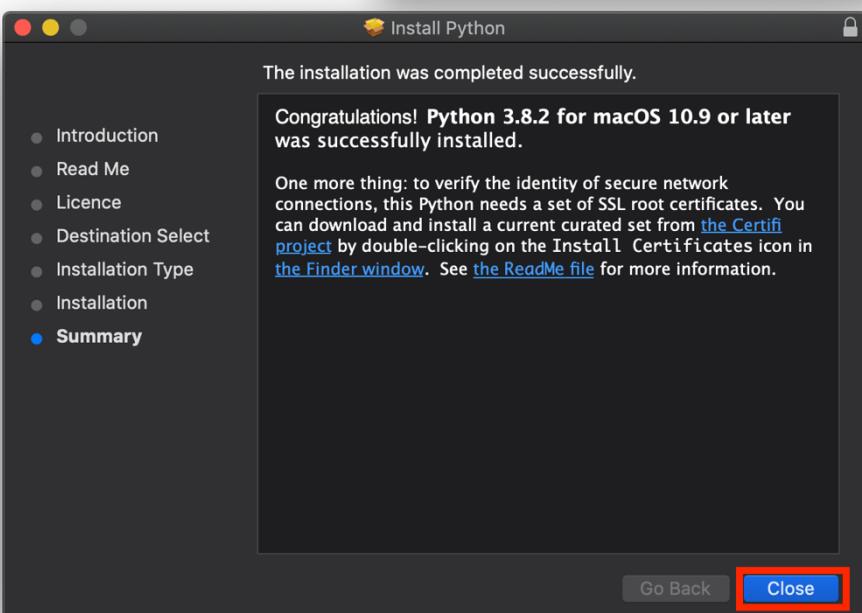
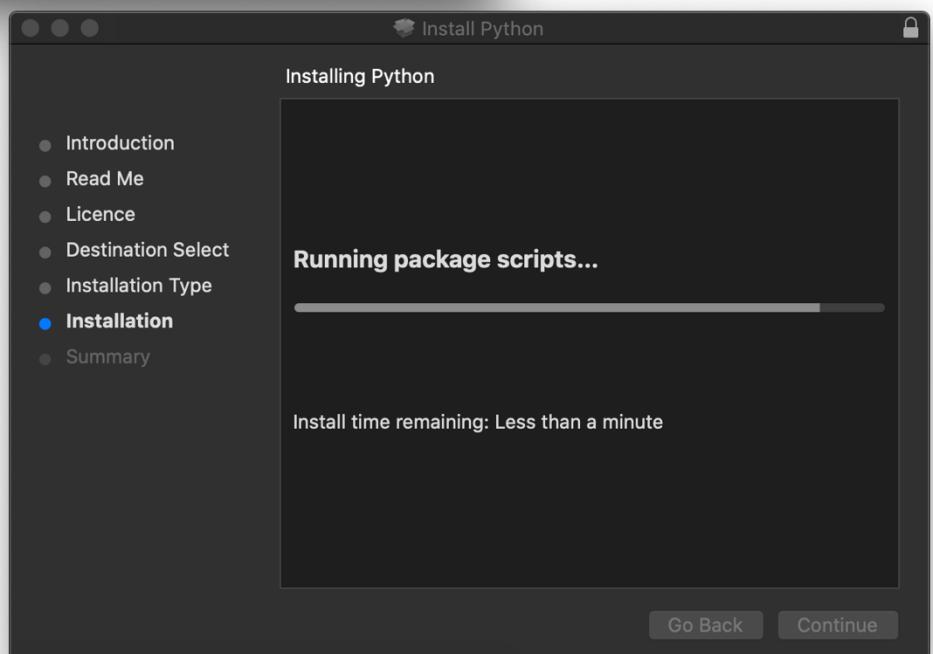


Click on **Agree** button.



## STEP - 9

Click **Install** button.



## STEP - 11

Now click on **Close** button.  
Python installation successfully done.

**STEP - 12**

Now open your **Terminal** and write command

**python3 –version**

Press Enter and that shows up installed version of Python.

```
[milankathiriya@Milans-MacBook-Pro ~ % python3 --version
Python 3.8.2
milankathiriya@Milans-MacBook-Pro ~ % ]
```

**B. INSTALLATION PROCESS ON WINDOWS****STEP - 1**

First go to this link: <https://www.python.org/>

**STEP - 2**

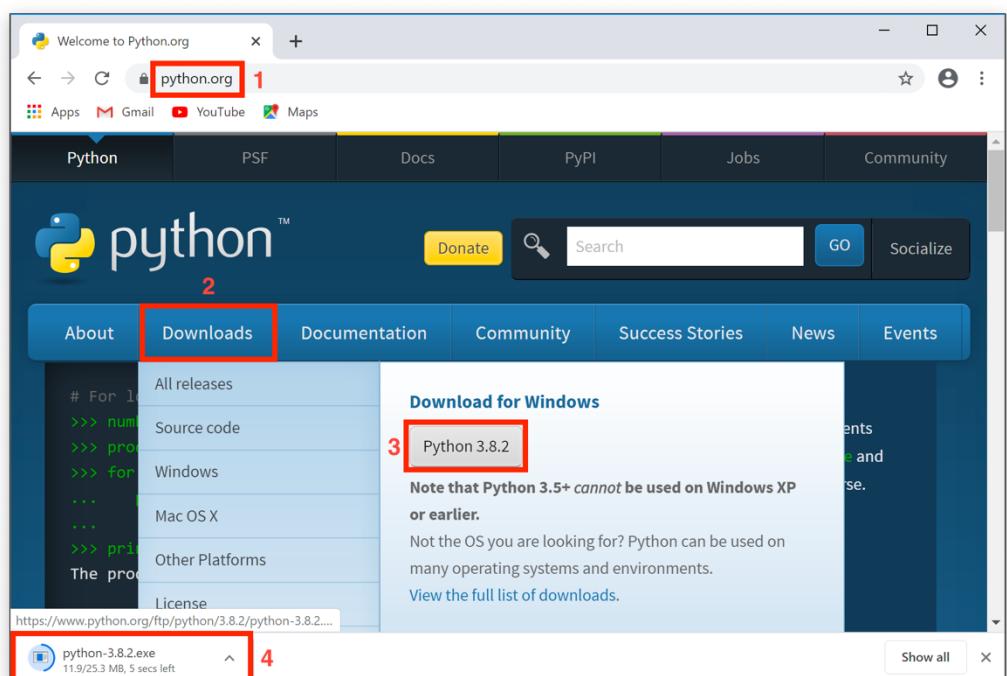
Hover on **Downloads** button from navigation bar.

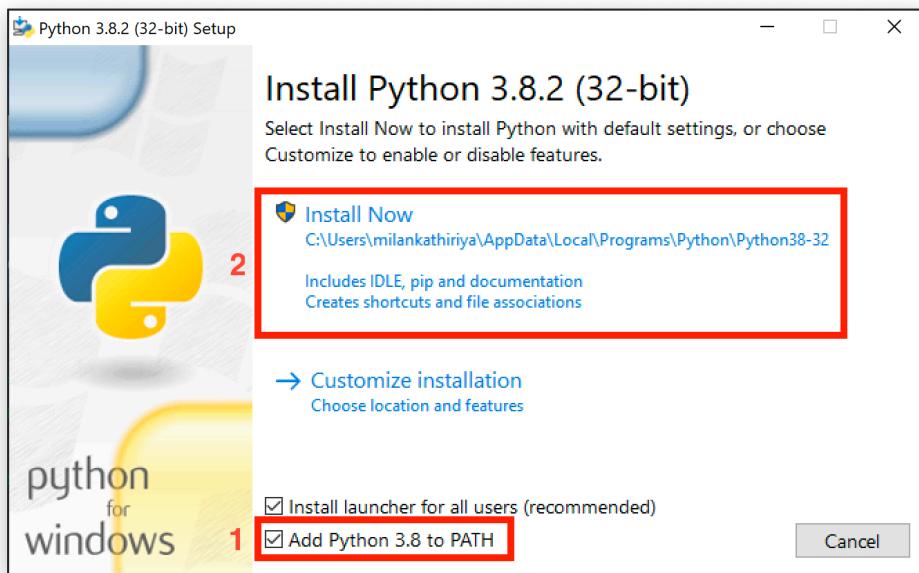
**STEP - 3**

Click on **Python 3.8.2** button.

**STEP - 4**

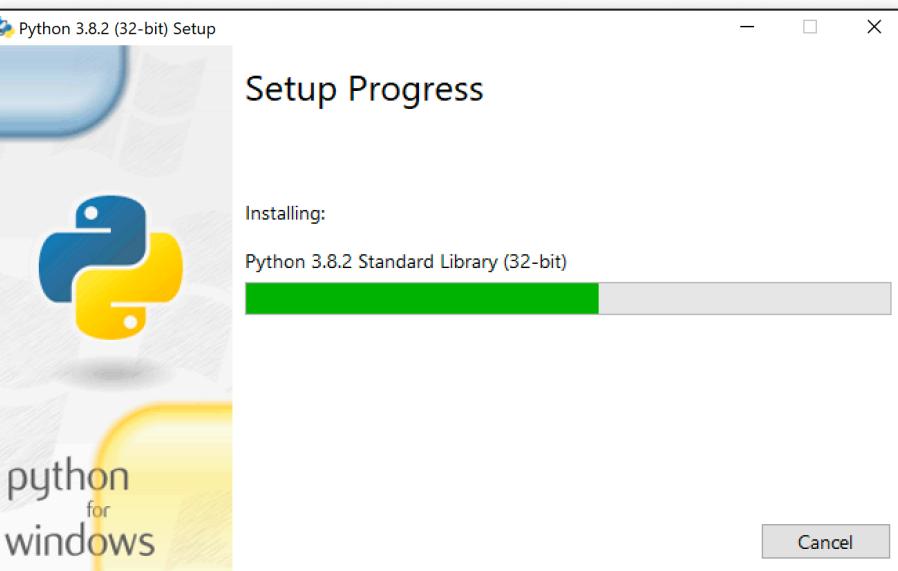
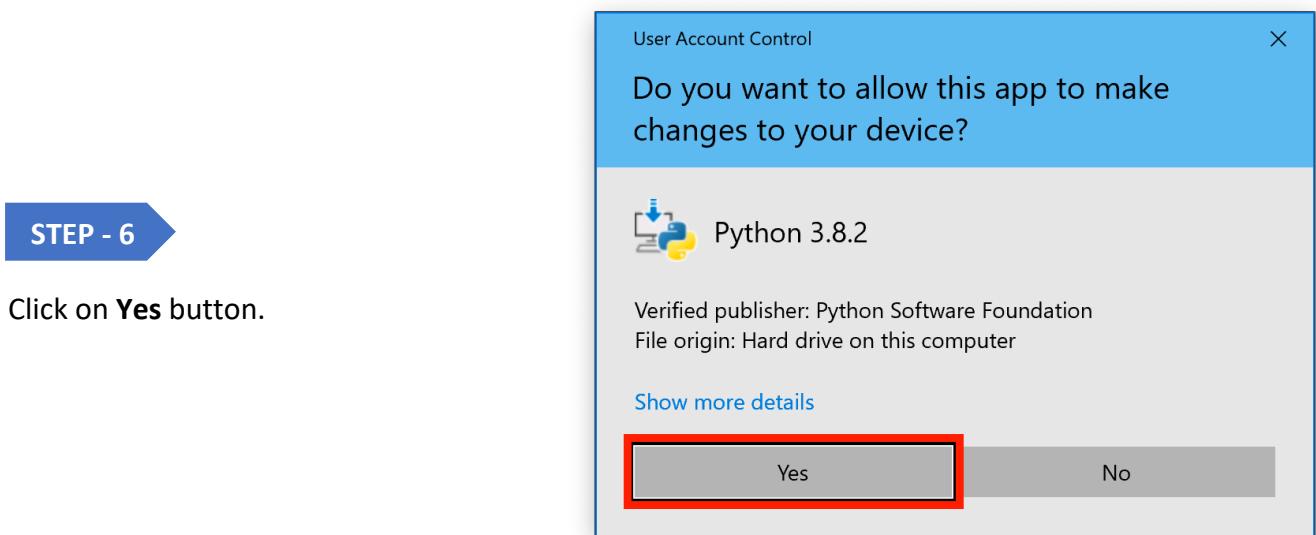
Download will start.



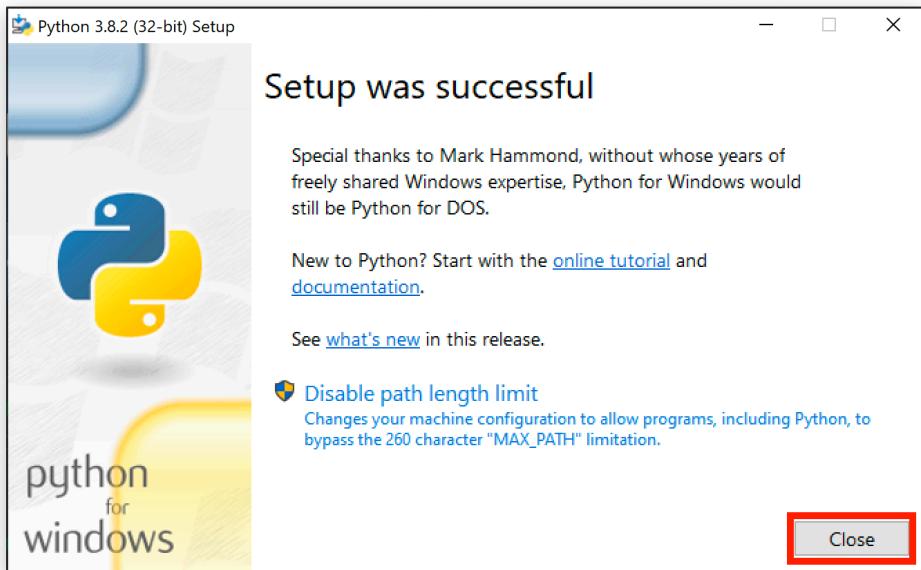
**STEP - 5**

First check **Add Python 3.8.2 to PATH**

Then, Click on **Install Now**

**STEP - 7**

Setup is now installing.



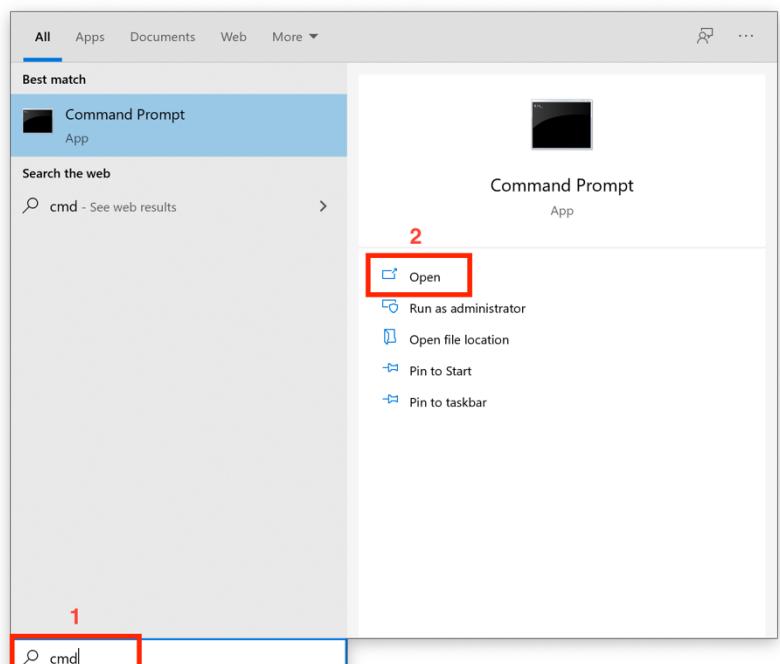
### STEP - 8

Now, click on **close** button.

Python installation successfully done.

### STEP - 9

Search **cmd** in search bar and click **Open**.



```
Microsoft Windows [Version 10.0.18363.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\milankathiriya>python --version
Python 3.8.2

C:\Users\milankathiriya>
```

### STEP - 10

Now open your **cmd** and write command  
**python –version**

Press Enter and that shows up installed version of Python.

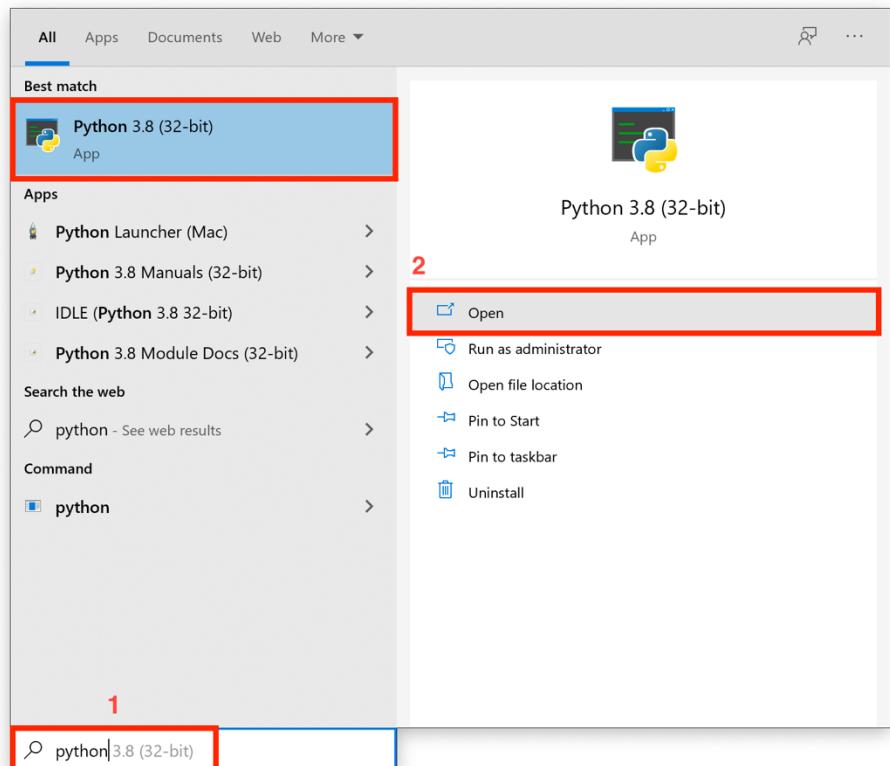
## PYTHON SHELL & IDLE

- पायथन हमें **Shell** और **IDLE (Integrated Development Environment)** जैसे built-in platform प्रदान करते हैं जिसमें हम पायथन का coding कर सकते हैं।
- पायथन को install करने के साथ ही shell और IDLE भी built-in साथ में ही install हो जाते हैं।

### A. Opening SHELL

#### STEP - 1

Search **python** in search bar and click on **Open** to open Python Shell.

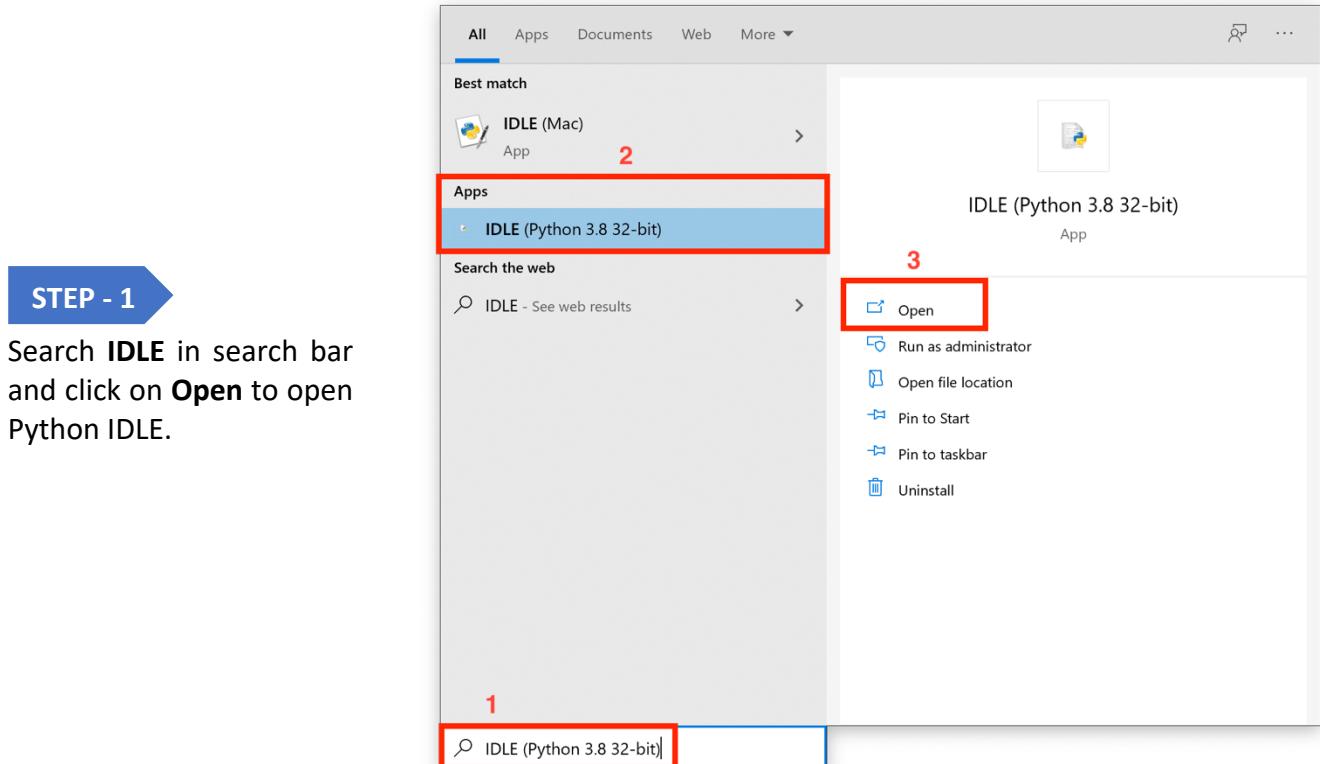


```
Python 3.8 (32-bit)
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

#### STEP - 2

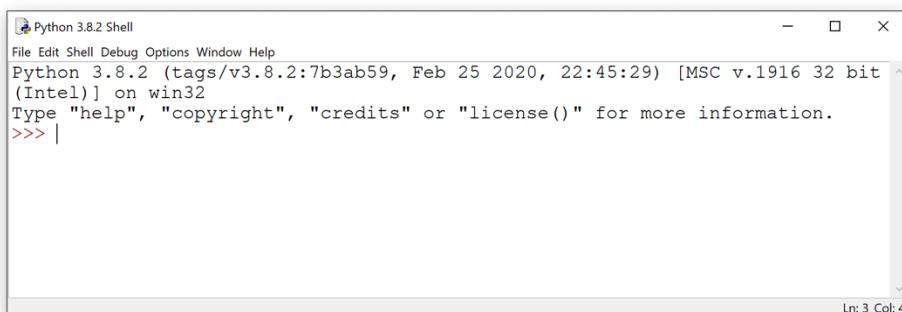
In shell window, **>>>** indicates interactive **interpreter** mode.

## B. Opening IDLE



### STEP - 1

Search **IDLE** in search bar and click on **Open** to open Python IDLE.



### STEP - 2

In IDLE window, **>>>** indicates interactive **interpreter mode**.

- IDLE को open करते ही वो by-default interactive interpreter mode में open होता है | पर IDLE को script mode में भी उपयोग में लिया जा सकता है |

## FIRST PYTHON PROGRAM / TYPES OF MODE TO RUN PROGRAM

- पायथन प्रोग्राम को रन करने के लिए 2 methods हैं:

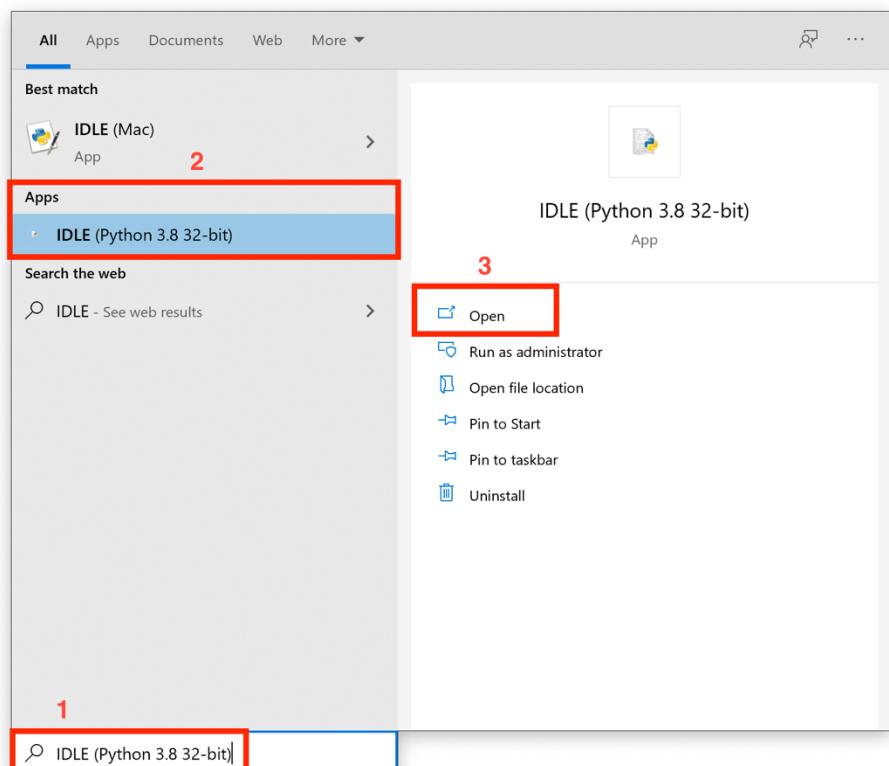
- Using Interpreter Mode
- Using Script Mode

### A. Using Interpreter Mode

- Interpreter mode** का मतलब प्रोग्राम **लाइन-बाय-लाइन** चलाना है।
- यानी की जैसे जैसे हम एक एक लाइन लिखते जायेंगे वैसे वैसे वह लाइन रन होती रहेगी।
- जैसे एक लाइन का coding करके दूसरी लाइन में गए की तुरंत ही वह लाइन का output हमें देखने को मिलेगा।
- Interpreter mode में हम पूरा प्रोग्राम एक ही साथ लिखकर पीछे से रन नहीं कर सकते।
- यह mode सिर्फ testing या फिर learning के लिए उपयोग में लिया जाता है।
- Interpreter mode पायथन **Shell** और **IDLE** दोनों में available है।
- हम **IDLE** की मदद से Interpreter mode को समझेंगे।

#### STEP - 1

Search **IDLE** in search bar and click on **Open** to open Python IDLE.



Python 3.8.2 Shell

```

File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |

```

Ln: 3 Col: 4

**STEP - 2**

In IDLE window, **>>>** indicates interactive **interpreter mode**.

**STEP - 3**

To print **Hello World**, write below code:

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

Press **Enter** for see output.

Notice here, Python doesn't have **semicolon**.

Python 3.8.2 Shell

```

File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020,
22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()
()" for more information.
>>> print("Hello World") ← Code
Hello World ← Output
>>> |

```

Ln: 5 Col: 4

Python 3.8.2 Shell

```

File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020,
22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()
()" for more information.
>>> print("Hello World")
Hello World
>>> print(4+5) ←
9
>>> 4+5 ←
9
>>>

```

Ln: 9 Col: 4

**STEP - 4**

As we discussed before, that **interpreter mode** is for **testing and learning**.

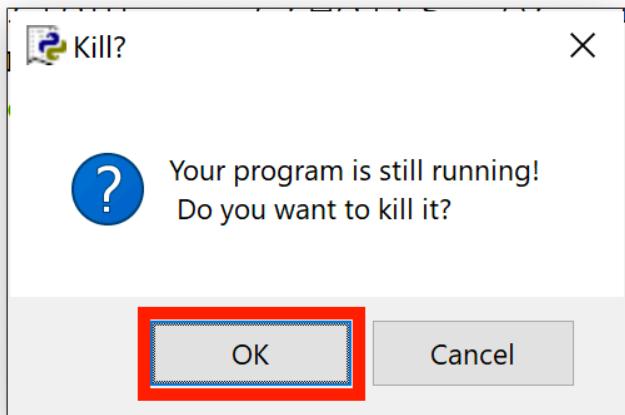
That's why if we **skip print()** function, even after that we get our output.

**STEP - 5**

To get exit from IDLE window, run `exit()` function.



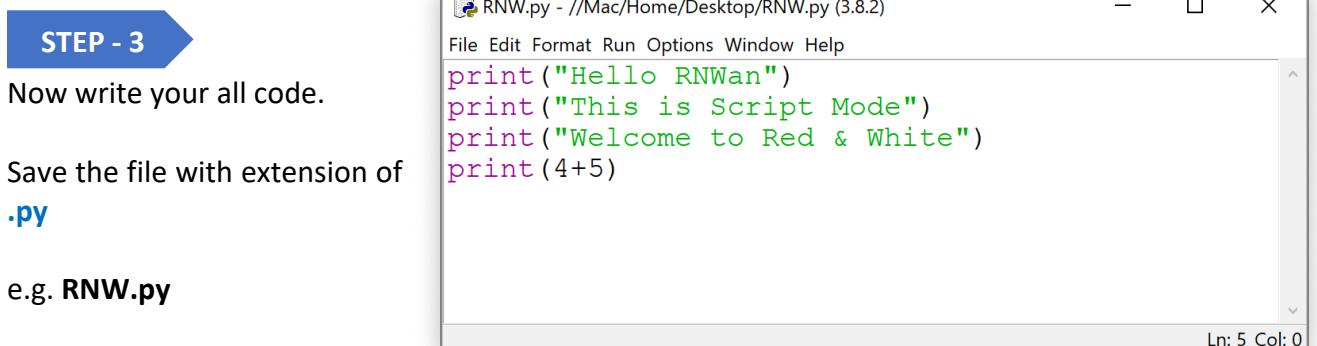
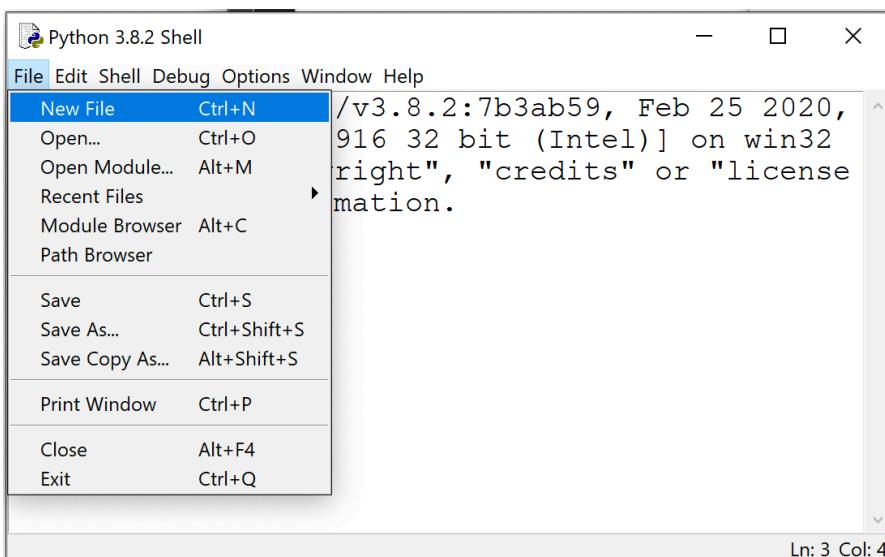
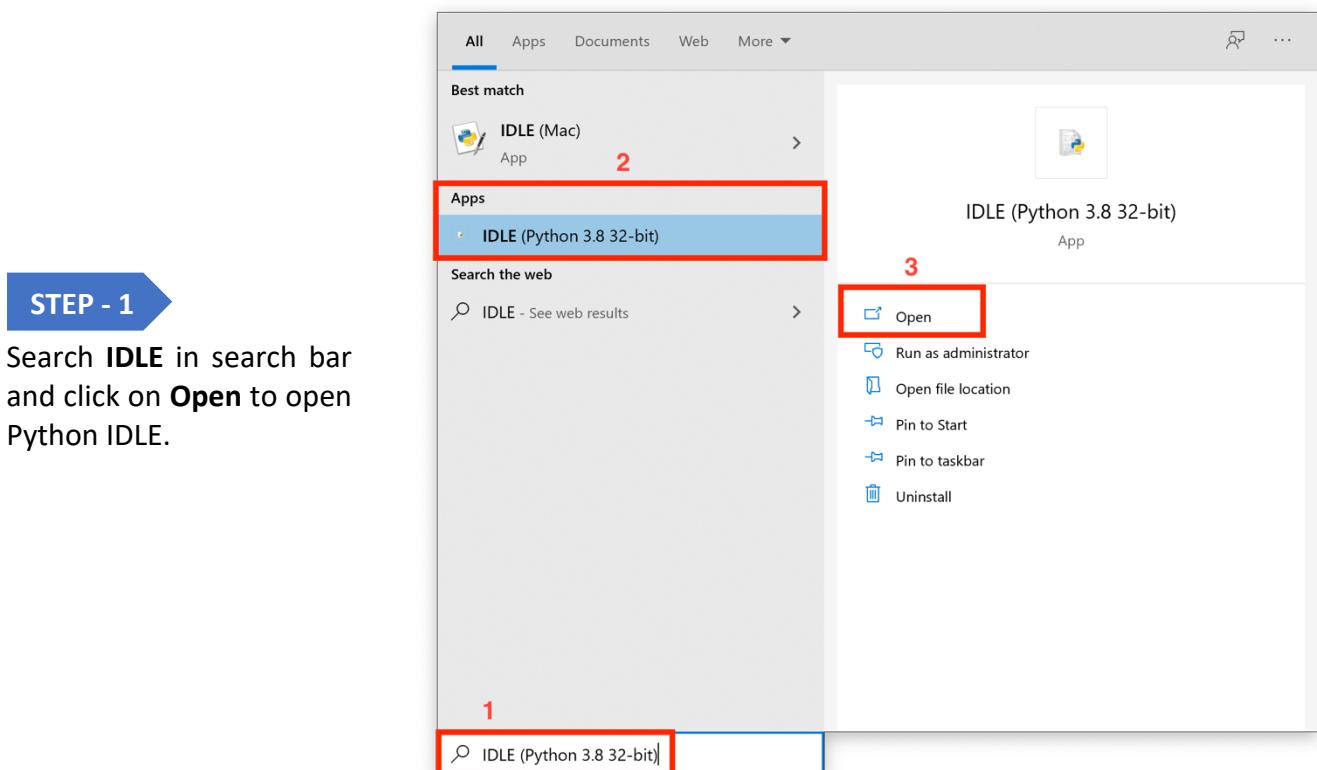
```
*Python 3.8.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020,
22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license"
()" for more information.
>>> print("Hello World")
Hello World
>>> print(4+5)
9
>>> 4+5
9
>>> exit()
Ln: 9 Col: 10
```

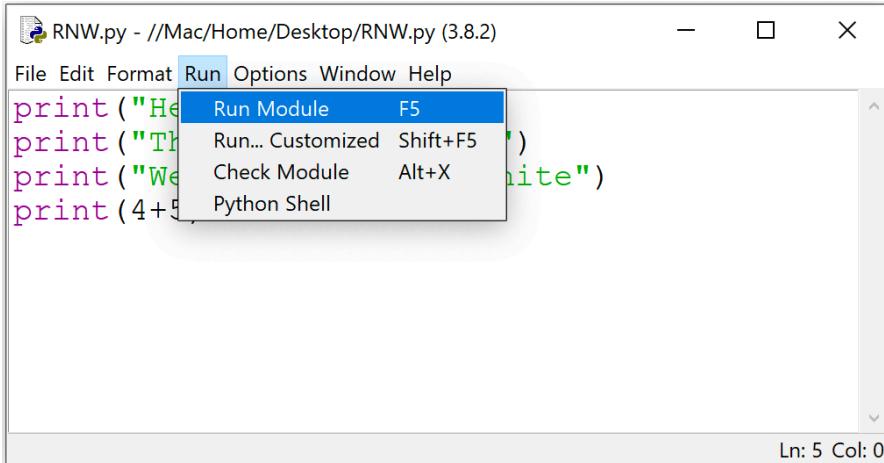
**STEP - 6**

Click on **OK** button to completely close IDLE window.

## B. Using Script Mode

- **Script mode** का मतलब प्रोग्राम को पूरी तरह से लिखे जाने के बाद चलाना है।
- Script mode में आपको किसी अन्य प्रोग्रामिंग लैंग्वेज की तरह एक फाइल बनानी होगी और उसमें कोडिंग करने के बाद आपको उस फाइल को चलाना होगा। ताकि हम पूरे प्रोग्राम के आउटपुट को एक साथ लास्ट में देखें।
- इस mode को developing के main mode के रूप में उपयोग में लिया जाता है।
- पायथन प्रोग्राम की एक **file** बनाकर उस file को रन करने से हमें output मिलता है | तो इस पायथन की file को **script** के रूप में भी जाना जाता है |
- हम **IDLE** का उपयोग करके स्क्रिप्ट मोड को समझेंगे।





```
RNW.py - //Mac/Home/Desktop/RNW.py (3.8.2)
File Edit Format Run Options Window Help
print ("Hello RNWan")
print ("This is Script Mode")
print ("Welcome to Red & White")
print (4+5)

Ln: 5 Col: 0
```

**STEP - 4**

(Run Program)

From **Run** menu, select **Run Module** to run a script.

or

Press **F5** to run a script.

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: //Mac/Home/Desktop/RNW.py ====
filepath
Hello RNWan
This is Script Mode
Welcome to Red & White
9
>>>

Ln: 9 Col: 4
```

**STEP - 5**

As you can see, output always displays in a separate **Shell** window.

File path also mentioned with output.

## #2

## FUNDAMENTALS OF PYTHON

## I/O (INPUT/OUTPUT) FUNCTIONS

- पायथन आउटपुट के लिए `print()` फ़ंक्शन और इनपुट के लिए `input()` फ़ंक्शन का उपयोग करता है।
- इसका मतलब यह है कि पायथन भाषा में 2 कार्यों का उपयोग I/O functionality के लिए किया जाता है:
  - `print()`
  - `input()`

**A. For OUTPUT**

- पायथन आउटपुट के लिए `print()` फ़ंक्शन का उपयोग करता है।
- `print()` फ़ंक्शन का उपयोग 2 तरीकों से किया जा सकता है:
  - For simple printing
  - For modified printing

Simple Printing:

## Syntax

```
print( your_statement )
```

## Syntax

```
print( your_statement )
```

## Example

```
print( "Hello RNWan" )
print( "Welcome" )
```

## Example

```
print( 5+3 )
```

## Output

Hello RNWan  
Welcome

## Output

8

### Modified Printing:

- By default, print() फ़ंक्शन का उपयोग करके, एक नई लाइन (\n) अंत में किसी भी लाइन के आउटपुट से जुड़ी होती है।
- print() फ़ंक्शन में **end** नाम की एक **optional property** होती है। जिसकी मदद से हम नई लाइन (\n) के व्यवहार को बदल सकते हैं।
- **end** property को एक placeholder प्रदान करना होगा जो नई लाइन (\n) को बदल देता है।

### Syntax

```
print( statement, end="placeholder" )
```

### Example

```
print( "Hello RNWan", end="&" )
print( "Welcome" )
```

### Output

Hello RNWan&Welcome

## B. For INPUT

- पायथन input के लिए **input()** फ़ंक्शन का उपयोग करता है।
- input() फ़ंक्शन का उपयोग 2 तरीकों से किया जा सकता है:
  - A. Input without message
  - B. Input with message

### Input without message:

### Syntax

```
variable_name = input( )
```

### Example

```
RW = input()
print( Red )
```

**Output**

```
Red & White      # input  
Red & White
```

**Input with message:****Syntax**

```
variable_name = input( "your_message" )
```

**Example**

```
RW = input( "Enter any value: " )  
print( Red )
```

**Output**

```
Enter any value: Red & White  
Red & White
```

**DATA-TYPES IN PYTHON**

- निम्नलिखित DataTypes पायथन भाषा में उपलब्ध हैं:
  - int
  - float
  - complex
  - str
  - bool

## VARIABLES IN PYTHON

- किसी भी अन्य प्रोग्रामिंग language के विपरीत, पायथन में एक variable declare नहीं किया जा सकता है।
- पहली बार जब हम एक value को assign करते हैं, तो यह एक variable बन जाता है।
- **Assignment operator (=)** का उपयोग किसी variable को value assign करने के लिए किया जाता है।

### Example

```
R = 10
W = 10
```

- किसी variable पर value assign करते समय datatype को नहीं लिखा जा सकता है।
- पायथन में variable का **datatype** value के मुताबिक **automatic decide** हो जाता है।
- और फिर भी हम value assign किए जाने के बाद भी datatype को बदल सकते हैं।

### Example

```
R = 10          // R is of type int
R = "GIM"      // R is of type str
```

- **String variables** ने **single quotes**, **double quotes** अने **triple quotes** वडे declare કરી શકીએ છીએ.

### Example

```
course = 'GIM'
course = "GIM"
course = """GIM"""
course = """"""GIM""""
```

## OPERATORS IN DART

- Below are operators that are available in Python:

- Arithmetic Operators**
- Comparison Operators**
- Assignment Operators**
- Logical Operators**
- Identity Operators**
- Membership Operators**

### A. ARITHMETIC OPERATORS

Operator	Meaning
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo
**	Exponentiation
//	Floor Division

### B. COMPARISON OPERATORS

Operator	Meaning
==	Equal
!=	Not equal
>	Greater Than
<	Less Than
>=	Greater Than or Equal To
<=	Less Than or Equal To

## C. ASSIGNMENT OPERATORS

---

Operator	Meaning
=	Assign value
+=	Increment and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulo and assign
//=	Floor division and assign
**=	Exponentiation and assign

## D. LOGICAL OPERATORS

---

Operator	Meaning
and	True if both statements are true
or	True if one of the statements is true
not	Reverse the result, returns False if the result is true

## E. IDENTITY OPERATORS

---

Operator	Meaning
is	Returns True if both variables are the same object
is not	Returns True if both variables are not the same object

## F. MEMBERSHIP OPERATORS

Operator	Meaning
in	Returns True if a sequence with the specified value is present in the object Operator
not in	Returns True if a sequence with the specified value is not present in the object

## TYPE CASTING CONSTRUCTORS

- कई बार ऐसी situation होती हैं जब हमें एक निश्चित datatype को एक variable को assign करना होता है। तो हम **casting constructor** की मदद से ऐसा कर सकते हैं।

### NOTE:

- **EVERYTHING IS OBJECT IN PYTHON.**

- पायथन एक object-oriented language है, और इसलिए हम class द्वारा datatype को define कर सकते हैं।
- इसलिए हम constructor functions का उपयोग करके casting कर सकते हैं:
  - int()
  - float()
  - str()
  - complex()
  - bool()

### A. int()

#### Example

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

## B. **float()**

### Example

```
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
```

## C. **str()**

### Example

```
x = str("s1")          # x will be 's1'  
y = str(2)              # y will be '2'  
z = str(3.0)            # z will be '3.0'
```

## D. **complex()**

### Example

```
x = complex("6")        # x will be (6+0j)  
y = complex(7j)          # y will be (7j)  
z = complex(5.2)         # z will be (5.2+0j)
```

## E. **bool()**

### Example

```
a = bool(15)             # a will be True  
b = bool("Hello")         # b will be True  
c = bool(3.2)             # c will be True  
d = bool(0)               # d will be False  
e = bool(False)           # e will be False  
f = bool(None)            # f will be False  
g = bool("")              # g will be False
```

## id() & type() FUNCTIONS

### A. type() function

- **type()** फ़ंक्शन का उपयोग किसी भी **object** के **datatype** को प्राप्त करने के लिए किया जाता है।

#### Example

```
RED = 100
print( type(RED) )
```

```
WHITE = 15.2
print( type(WHITE) )
```

```
GIM = 'Best Course'
print( type(GIM) )
```

```
RNW = True
print( type(RNW) )
```

```
WEB = 7j
print( type(WEB) )
```

#### Output

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
<class 'complex'>
```

### B. id() function

- **id()** फ़ंक्शन किसी भी **object** की एक **unique id** दिखाता है।
- पायथन के सभी objects की अपनी unique id है।
- id केवल तब दी जाती है जब कोई object बनाया जाता है।
- id एक object का memory address है, जिसकी value प्रोग्राम चलने पर बदलता रहता है।

**Example**

```
RED = 100  
WHITE = 15.2  
GIM = 'Best Course'  
RNW = True  
WEB = 7j
```

```
print( id(RED) )  
print( id(WHITE) )  
print( id(GIM) )  
print( id(RNW) )  
print( id(WEB) )
```

**Output**

```
4305393408  
4308755984  
4311279472  
4305170344  
4311227568
```

#3

## STRING IN DETAIL

### STRING DATATYPE

- हम single quotes, double quotes और triple quotes के साथ String variables declare कर सकते हैं।
- जिसमें single quotes और double quotes का मतलब एक ही होता है। यानी 'hello' और "hello" दोनों एक ही माने जाते हैं।

#### Example

```
x = 'Hello'  
y = "Hello"
```

- Triple quotes का उपयोग करके multiline string बनाई जा सकती है। Triple quotes को सिंगल या डबल quotes के सिंबल को तीन बार टाइप करके बनाया जा सकता है।

#### Example – Using single quotes

```
RNW = """Welcome to Red & White Group of Institutes.  
This is Multimedia Education department.  
You are learning Python."""
```

#### Example – Using double quotes

```
RNW = """"""Welcome to Red & White Group of Institutes.  
This is Multimedia Education department.  
You are learning Python.""""""
```

## STRING FORMATTING

- पायथन language में string और numbers को जोड़ नहीं सकते, जैसे नीचे दिए गए उदाहरण में:

### Example

```
age = 30
txt = "My name is Python, I am " + age
print(txt)
```

- उपरोक्त उदाहरण हमें एक error देता है।
- लेकिन हम 3 अलग-अलग तरीकों का उपयोग करके string और numbers को जोड़ सकते हैं।
- कोई भी object तो string में inject करना, उसे **string interpolation** कहते हैं:
  - Using format() method**
  - Using format specifier placeholders**
  - Using f-string**

### A. Using **format()** method

- String के अंदर inject किए जाने वाले objects को **format()** method के **arguments** के तौर पर दिया जाता है | और यह arguments, string में दर्शाए गए **placeholder {}** की जगह पर replace हो जाते हैं |

### Example

```
age = 30
txt = "My name is Python, I am {}"
print(txt.format(age))
```

### Output

My name is Python, I am 30

- format() method को कितने भी **arguments** दे सकते हैं और वे क्रम-wise अपने placeholders की जगह पर inject हो जाते हैं।

### Example

```
quantity = 100
branch = 5
RNW = "Red & White Group of Institutes have total of {} branches
and more than {} courses available."
print(RNW.format(branch, quantity))
```

### Output

Red & White Group of Institutes have total of 5 branches and  
more than 100 courses available.

- हम **index numbers** का भी उपयोग कर सकते हैं ताकि सभी arguments उनके निर्दिष्ट placeholder स्थान पर रखे जाएं।

### Example

```
quantity = 100
branch = 5
RNW = "Red & White Group of Institutes have total of { 1 }
branches and more than { 0 } courses available."
print(RNW.format(quantity, branch))
```

### Output

Red & White Group of Institutes have total of 5 branches and  
more than 100 courses available.

## B. Using format specifier placeholders

- पायथन language में हम C language format की specifier वाली syntax के अनुसार "%" operator का उपयोग करके string formatting कर सकते हैं।
- जैसे %d का उपयोग integer को दर्शाने के लिए किया जाता है और string को दर्शाने के लिए %s का उपयोग किया जाता है।

### Example

```
name = "RNWan"
print("Hello, %s" % name)
```

### Output

Hello, RNWan

- दो या अधिक arguments के लिए नीचे दिखाए गए example की तरह parentheses का उपयोग करें।

### Example

```
name = "RNW"
age = 12
print("%s is %d years old." % (name, age))
```

### Output

RNW is 12 years old.

- पायथन सभी objects को %s के साथ भी प्रदर्शित कर सकता है, चाहे object का कोई भी datatype हो।

### Example

```
name = "RNW"
courses = [ "GIM", "WEB", "FLUTTER", "PYTHON" ] // list datatype
print("%s has %s courses." % (name, courses))
```

**Output**

RNW has [ “GIM”, “WEB”, “FLUTER”, “PYTHON” ] courses.

**C. Using f-string**

- पायथन language में string formatting को **f-string** की मदद से आसान बनाया जा सकता है।
- f-string** को पायथन 3.6 version में introduce किया गया था।
- f-string में भी format() method की तरह **placeholders { }** का उपयोग किया जाता है। पर यह placeholders { } के अंदर related objects या variables को दर्शाना अनिवार्य है।
- f-string को निचे बताये गए example की तरह **lower f** या तो **upper F** द्वारा indicate किया जा सकता है।

**Example**

```
name = "RNW"
age = 12
print(f"{name} is {age} years old.")
print(F"{name} is {age} years old.")
```

**Output**

RNW is 12 years old.  
RNW is 12 years old.

## STRING MANIPULATION

- पायथन भाषा में **character** datatype available नहीं है। लेकिन पायथन में, एक single character को 1 की लंबाई के साथ एक **string** माना जाता है।
- String को अन्य languages की तरह पायथन में **characters** के एक **array** की तरह माना जाता है।
- तो string के elements को access करने के लिए **square brackets [ ]** का उपयोग किया जाता है।

### Syntax

```
string[ index ]
```

### Example

```
RNW = "Welcome to Red & White"
print( RNW[1] )
print( RNW[-2] )      # negative index starts from(-1) last position
```

### Output

```
e
t
```

- यदि हम string से कोई **specific characters** चाहते हैं, तो हम string **slicing** का उपयोग कर सकते हैं।
- Slicing करते समय, **starting index** और **ending index** को **square brackets [ ]** में specify करना पड़ता है जिसको comma द्वारा separate किया जाता है।

### Syntax

```
string[ starting_index : ending_index ]
```

### Example

```
RNW = "Welcome to Red & White"
print( RNW[11 : 14] )
print( RNW[-5 : -1] )
```

**Output**

```
Red
Whit
```

- Slicing करते समय, हम index के साथ-साथ **stepping value** भी provide कर सकते हैं। जितना अधिक stepping value होता है, उतने ही character, slicing करते समय **skip** होते रहेंगे।

**Example**

```
RNW = "Welcome to Red & White"
print( RNW[0 : 7 : 2] )
```

**Output**

```
Wloe
```

**TYPES OF COMMENTS**

- पायथन में 2 प्रकार की **comments** हैं:
  - Single-line Comment**
  - Multi-line Comment**

**A. Using Single-line Comment**

- Single-line Comment करने के लिए **#** का उपयोग किया जाता है।

**Example**

```
R = 10
# R = R - 3
print( R )
```

**Output**

10

**B. Using Multi-line Comment**

- Multi-line Comment करने के लिए **triple quotes** ( """ ) का उपयोग किया जाता है।

**Example**

```
"""Red & White Group of Institutes,  
Located in Surat,  
Gujarat, India  
"""
```