

U18CSI2201

Python Programming



KCED

Learning Augmented

Basics of Python Programming

Ms. Aswini D

Department of Computer Science and Engineering

Contents

- Course Outcomes
- Introduction
- History of Python
- Features of Python
- Python Installation steps
- Execution modes of Python
- Basics – Variables, Data Types
- Input/Output Functions
- Comments

Course Modules

- Basics of Python Programming
- Control Statements and Functions in Python
- Data Structures – List , Tuple, Strings,
- Data Structures – Set, Dictionaries
- Files, Modules, Packages

Course Outcomes

CO1

Classify and make use of python programming elements to solve and debug simple logical problems.(K4,S3)

CO2

Experiment with the various control statements in Python.(K3,S2)

CO3

Develop Python programs using functions and strings.(K3,S2)

CO4

Analyze a problem and use appropriate data structures to solve it.(K4,S3)

CO5

Develop python programs to implement various file operations and exception handling.(K3,S2)

Quick Introduction to Computer and Programming



Computers: is a digital electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program.



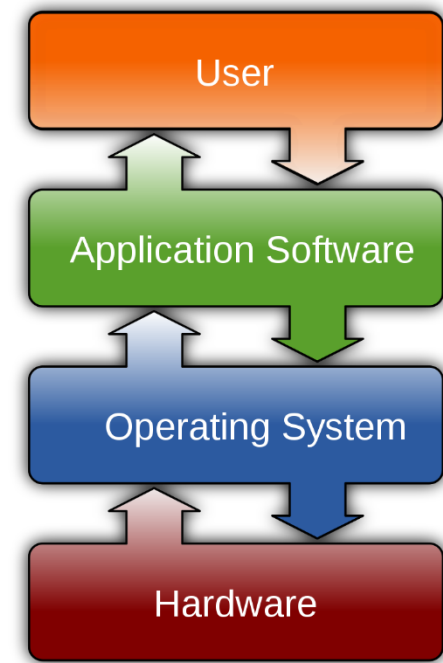
Through programming instructions, we interact with the CPU, memory, input /output devices to solve any simple or high-level problems.



A program is set of instructions written in a programming language to perform a task.



Collection of programs written for a specific application are commonly termed as software (Example: MS office, Media player, Adobe Photoshop, paint etc).



Types of Software



Windows® System Software

- System software is a collection of programs that supports computer operations
- Example: Operating System, Language Processor

Application Software



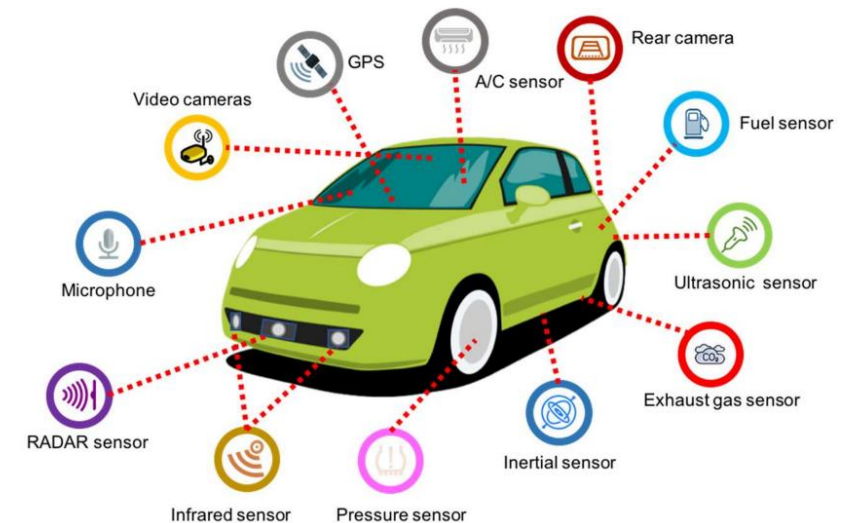
- Application software are the computer programs for performing user tasks
- Example: MS Office , Video player

How to develop
software???

What is Programming?

- Act of writing instructions for a computer to make the computer perform a task.
- Also called as coding
- Every app, game, etc., we use requires a coded program to work

Examples



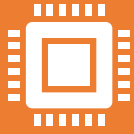
Programming language

- For a program to work it must be written in a way that a computer understands.
- This is done with a **programming language**.



Programming Language is a means of communication between a Human and Computer

Programming language



Programmers write the programs through the programming language C, C++, JAVA, C#, Python etc to instruct the computer.



Machine can understand any instruction in the form of 0s and 1s (Binary /Machine Language).



The compiler/interpreter converts the program into machine code.

Why to study Python?

- Python Programming language has ranked as one of the top 10 programming language .
- Python is used for developing applications ranging from simple gaming development to sophisticated software development.
- Tech giants like **Microsoft, YouTube, Instagram, Google, Spotify** has important part in developing their products using Python language.

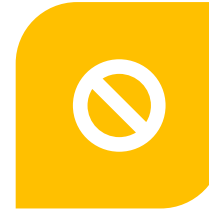
Features of Python



**EASY TO LEARN -
PERFECT
BEGINNER'S
LANGUAGE**



**INTERPRETED, HIGH-
LEVEL, GENERAL
PURPOSE LANGUAGE**



OPEN SOURCE



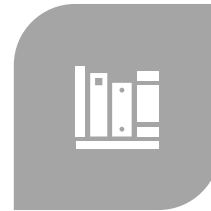
**PLATFORM
INDEPENDENT**



OBJECT ORIENTED



**POWERFUL
LANGUAGE FOR
BUSINESS
ANALYTICS**



**VAST SUPPORT OF
LIBRARIES**



**GLUES EASILY
WITH OTHER
LANGUAGES**



**STRONG
COMMUNITY
SUPPORT**

History of Python



Guido van Rossum was the creator of Python Programming language . Named after the Monty Python circus show



First version of Python (0.9.0) released in 1991-includes classes, lists, strings and powerful built-in functions.



Python 2.0 released in 2000-includes Unicode representation of characters, List comprehension and garbage collector.



Python 3.0 was released in Dec 2008



Latest version: Python 3.11.3



Official website: www.python.org

About IDE programming environment

- Integrated **D**evelopment **E**nvironments (IDEs) are available for writing programs, modifying and executing Python program.
- **Features to create and execute a Python program:**
 - Text editor to create Python program or script.
 - Bracket matching- matches the bracket's for decision making statement and loops
 - Highlighting the syntax-Spot and highlight the variables, keywords and symbols .
 - Auto completing your code- IDLE would recognise the end of loops/if statement and automatically indent the next line.
 - Execute the Python program – Run the code in the environment.
 - Check for errors- Use of debugging support to step through the code

Top Python IDEs





Step 1:

- Python Installer can be obtained from the **Python Software Foundation** website at www.python.org.
- Navigate to Downloads page of Python.org
- Click on latest version of Python

Step 2

Do you want to run this file?



Name: C:\Users\ASWINIII\Downloads\python-3.7.4.exe

Publisher: [Python Software Foundation](#)

Type: Application

From: C:\Users\ASWINIII\Downloads\python-3.7.4.exe

Run

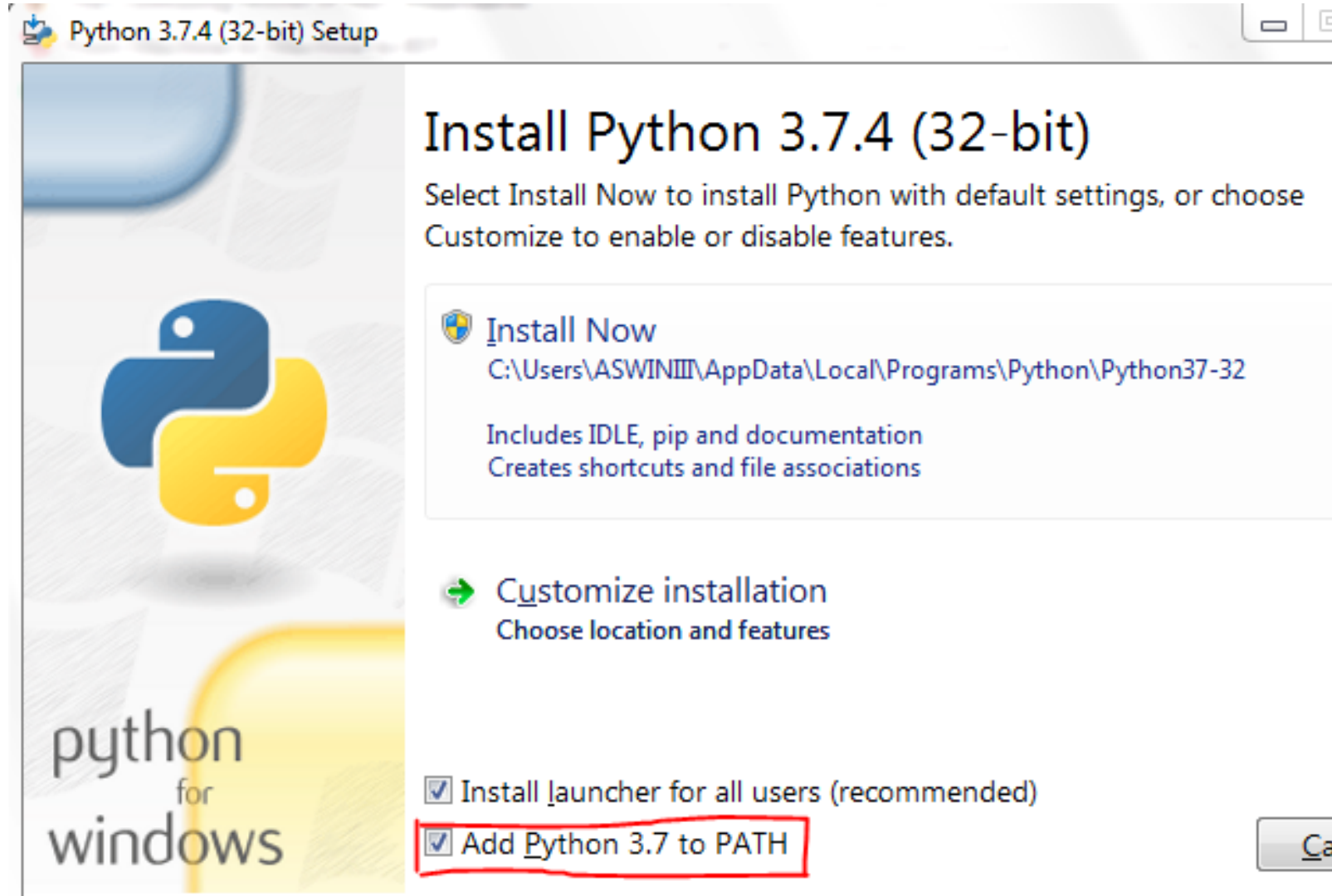
Cancel

☒ **Always ask before opening this file**

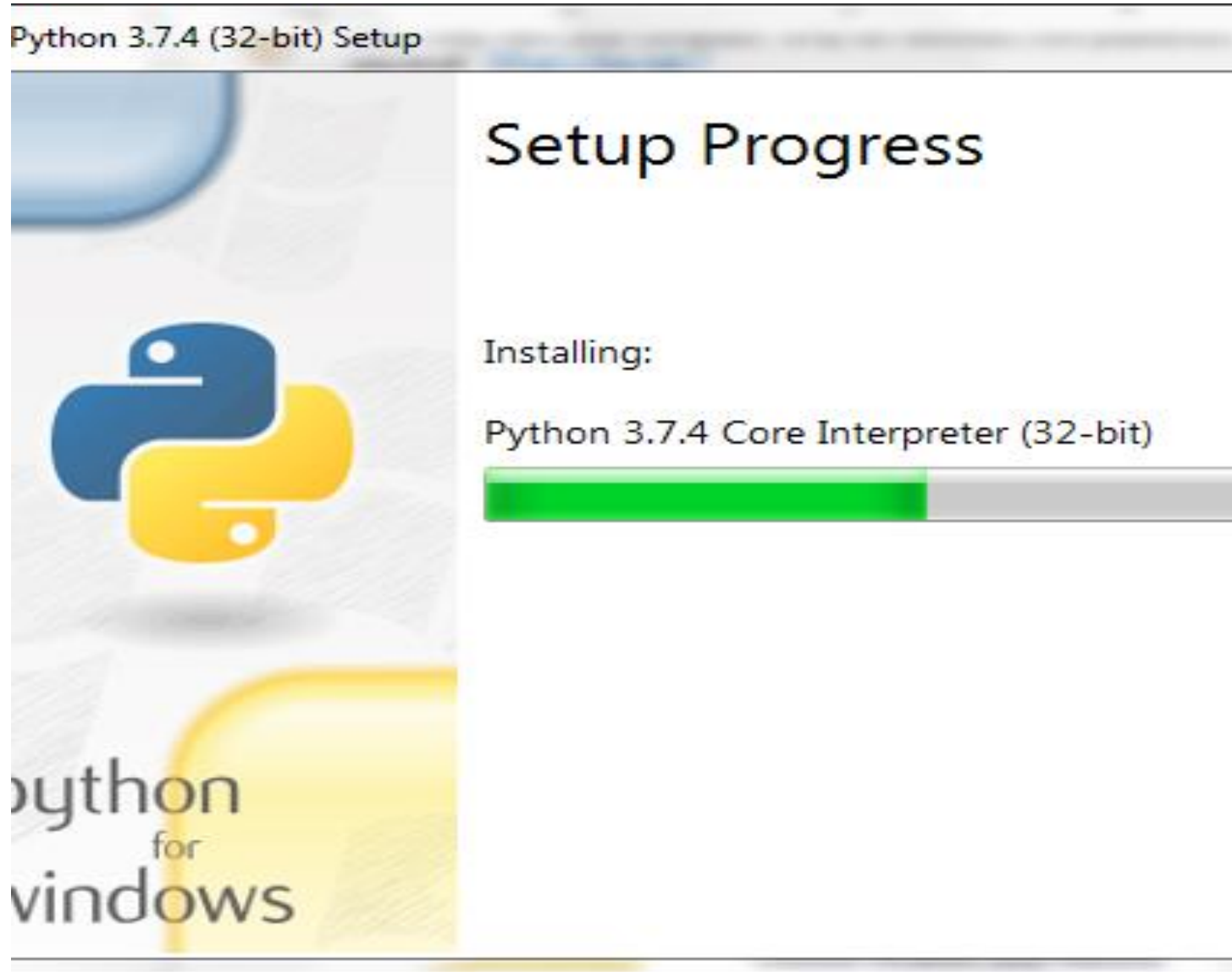


While files from the Internet can be useful, this file type can potentially ham your computer. Only run software from publishers you trust. [What's the risk?](#)

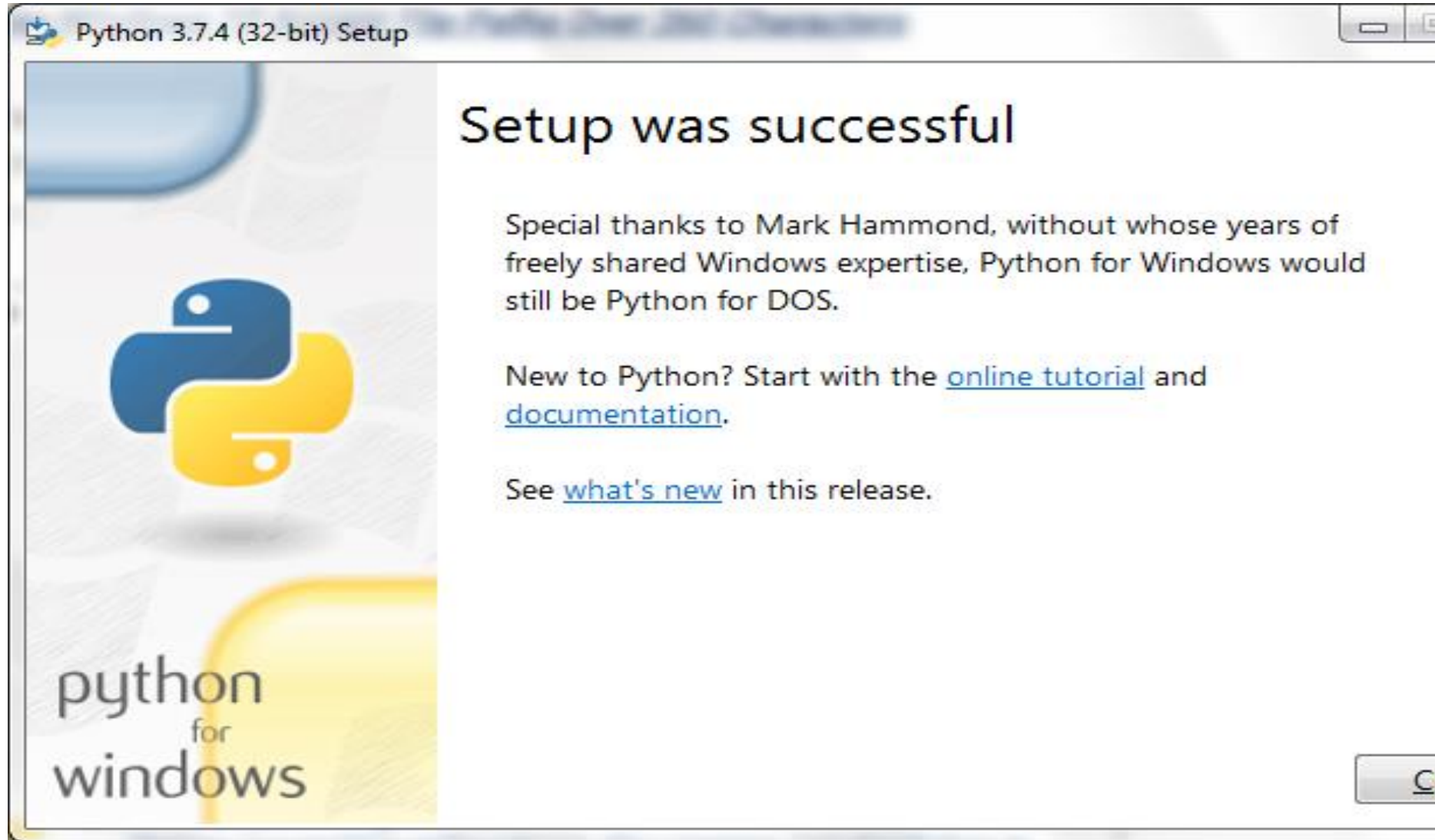
Step 3: Installation set up



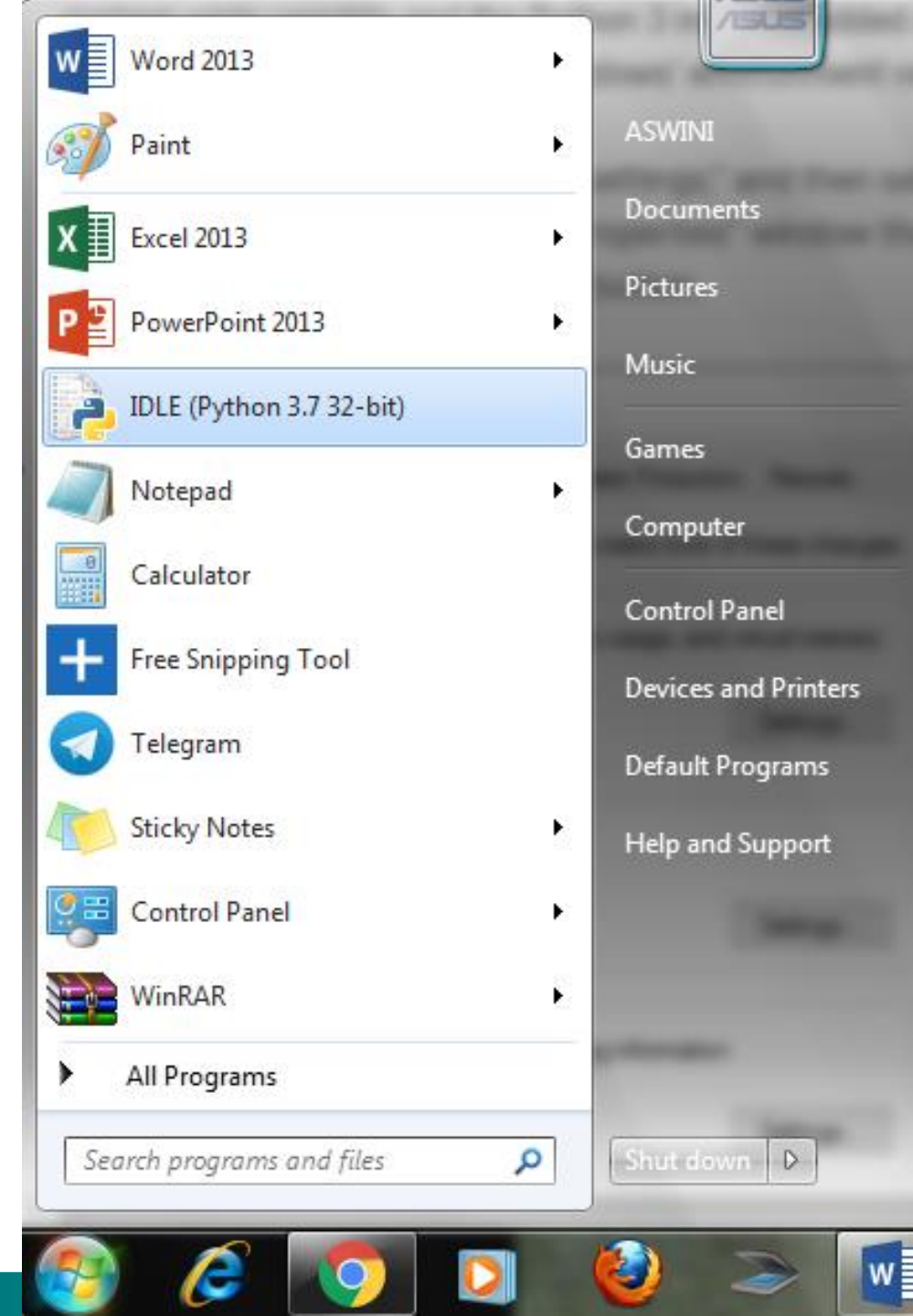
Step 4: Installation begins.



Step 5: Installation completed



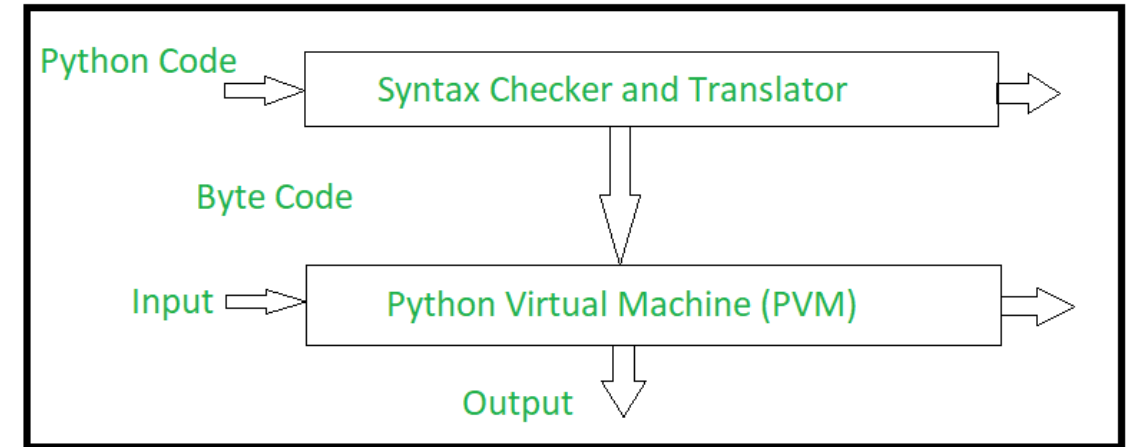
Step 6: Navigating to start menu-> IDLE to invoke Python IDLE



Python Interpreter

The Python interpreter performs following tasks to execute a Python program :

- **Step 1** : The interpreter reads a python code or instruction. Then it verifies that the instruction is well formatted, i.e. it checks the syntax of each line. If it encounters any error, it immediately halts the translation and shows an error message.
- **Step 2** : If there is no error, then the interpreter translates it into its equivalent form in intermediate language called “Byte code”..



Step 3 : Byte code is sent to the Python Virtual Machine(PVM). Here again the byte code is executed on PVM. If an error occurs during this execution then the execution is halted with an error message

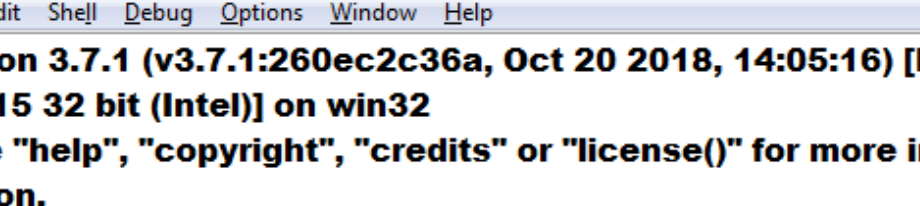
Execution Modes in Python

Python program execution can be done using two modes using IDLE:

- I. Interactive mode (Command Line mode)
- II. Script mode

Interactive mode

- **Step 1:** Press the start button and type “Python IDLE” in the search bar.
- **Step 2:** Select IDLE (Python 3.7 32-bit). This is Python shell which comes up with the Python installation.
- The >>> prompt indicates the readiness to accept the Python statements for execution.



The screenshot shows a 'Python 3.7.1 Shell' window. The title bar includes standard window controls (minimize, maximize, close) and the text 'Python 3.7.1 Shell'. The menu bar contains 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays the following content:

```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC  
v.1915 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license()" for more infor  
mation.  
>>> print("welcome to python programming")  
welcome to python programming  
>>> print("my first python code")  
my first python code  
>>> 10*3  
30
```

The status bar at the bottom right indicates 'Ln: 7 Col: 0'.

Python has command line interpreter that helps in executing python commands by directly entering without writing a script

Interactive mode

Advantages:

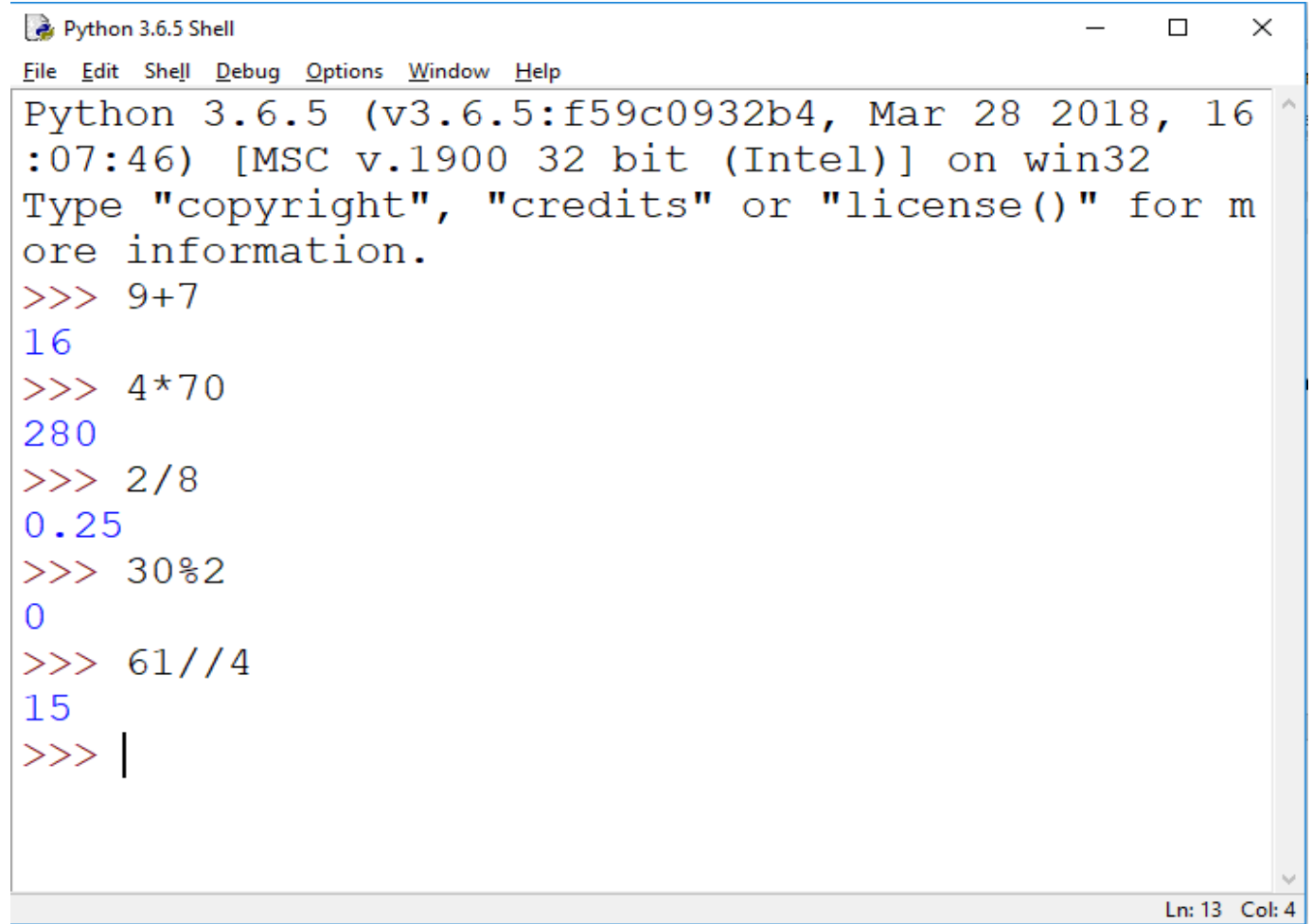
- Command mode is useful when you want to check the simple Python statements execution quickly.
- Useful for beginners to understand the basics of Python.
- Faster in executing the line of code.

Disadvantages:

- Difficult to run a longer piece of code
- Python statement executed previously will not be stored.
- Python statements that you have typed in shell will be available in the shell till it is open.
- Incorrect Python statement –Difficult to edit the statement.
- Need to retype it.

Try out the following and check for the response in Python shell:

- a) $9+7$
- b) $4*70$
- c) $2/8$
- d) $30\%2$
- e) $61//4$



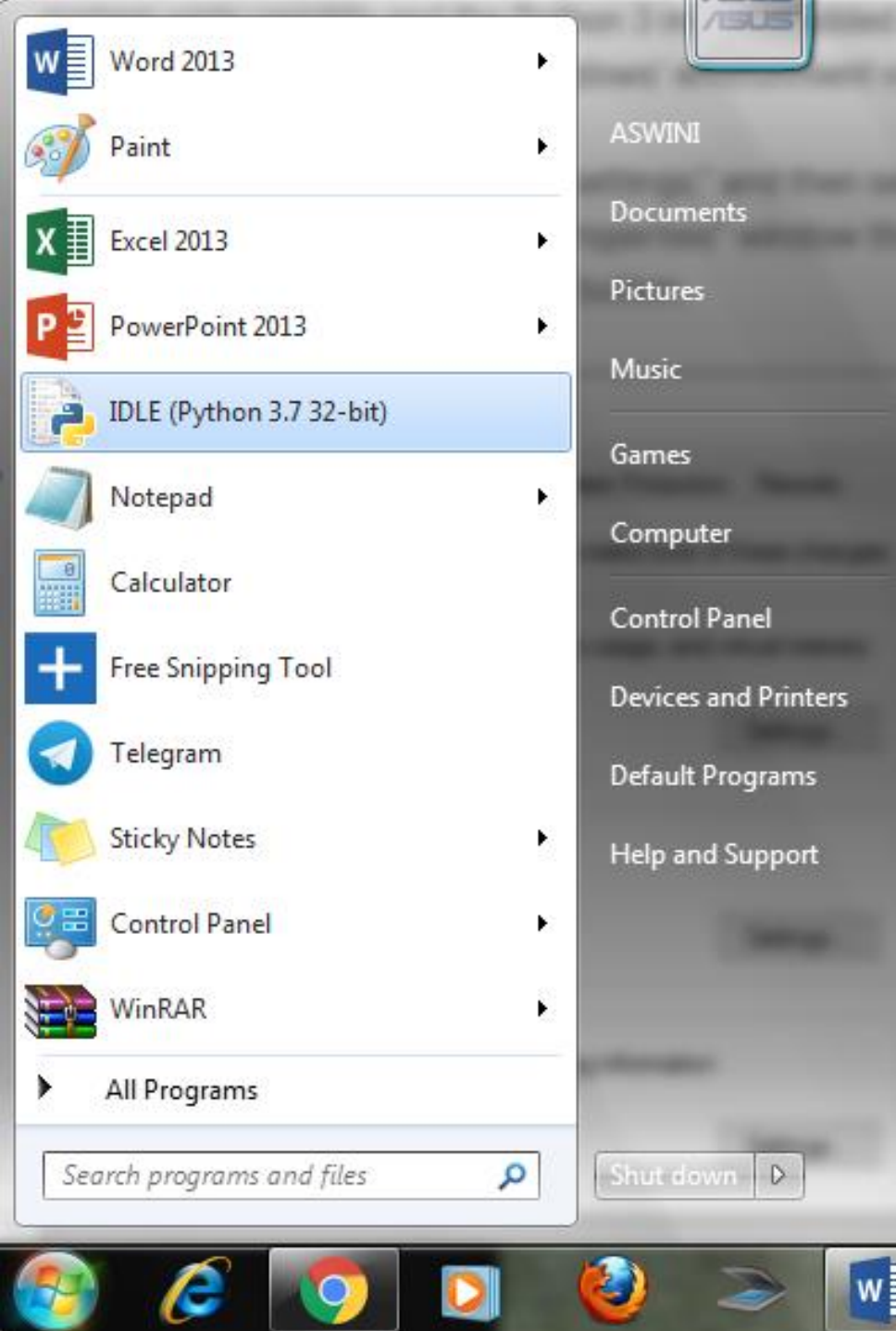
```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 9+7
16
>>> 4*70
280
>>> 2/8
0.25
>>> 30%2
0
>>> 61//4
15
>>> |
```

Ln: 13 Col: 4

Script mode

What is script?

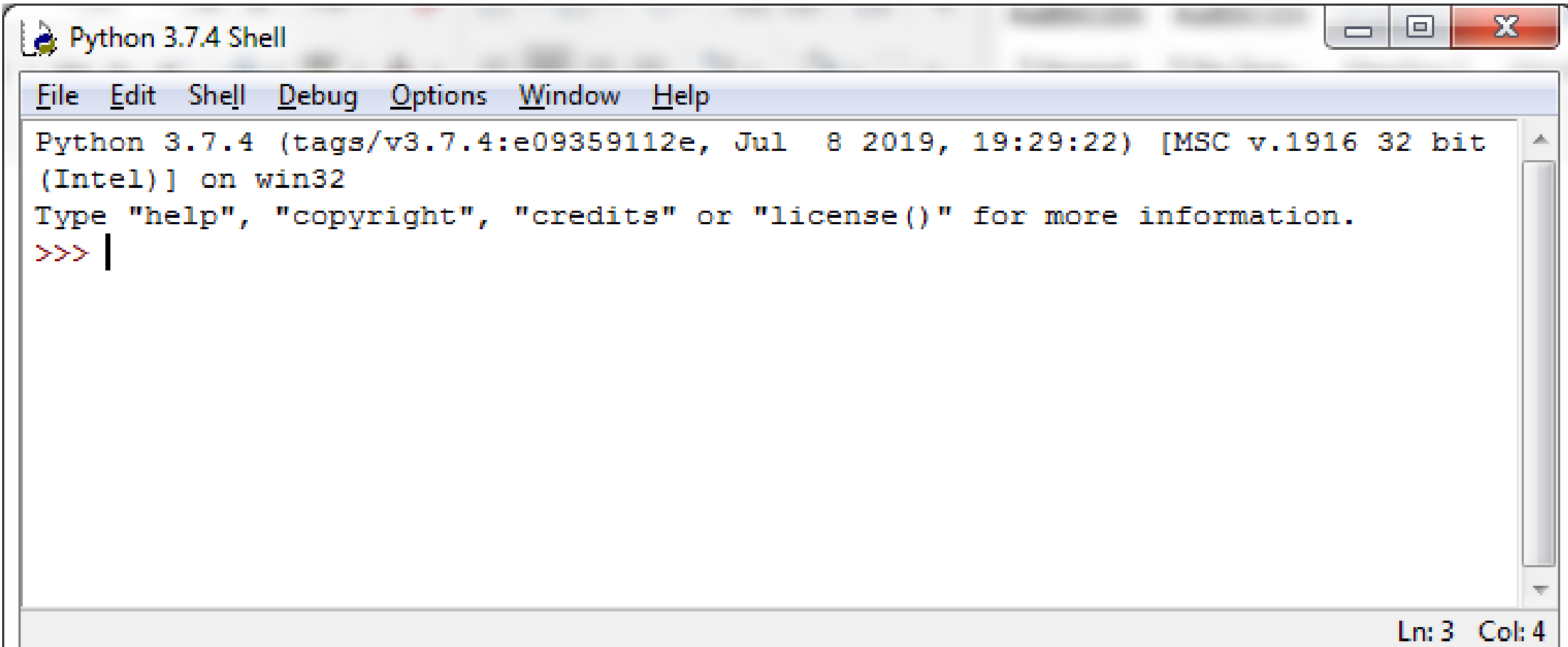
- Store the Python statements in a file, it is known as script and file is stored with .py extension.
- Python statements (also known as Source code) is stored in the file named “hello.py”.
- Source code can be edited anytime because, it is stored in a file and can be accessed anytime for execution as well.



Steps for running the program in script mode

- **Step 1:** Choose the IDLE (Python 3.7.4)

Step 2: Python IDLE opens the Shell.

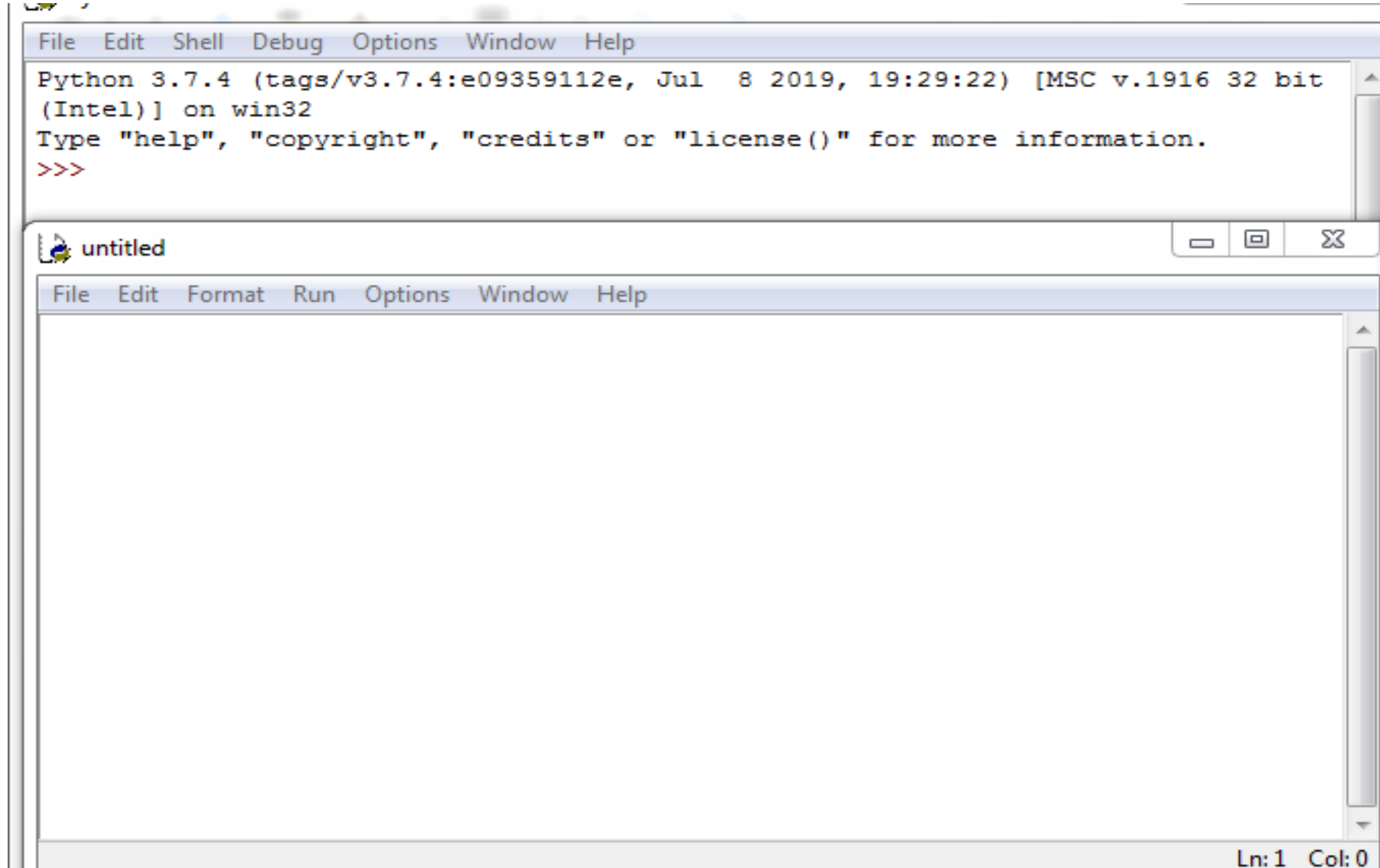


A screenshot of the Python 3.7.4 Shell window. The window has a title bar that says "Python 3.7.4 Shell" and standard Windows window controls (minimize, maximize, close). Below the title bar is a menu bar with the following options: File, Edit, Shell, Debug, Options, Window, and Help. The main text area contains the following text: "Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32", "Type 'help', 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>> |". The status bar at the bottom right shows "Ln: 3 Col: 4".

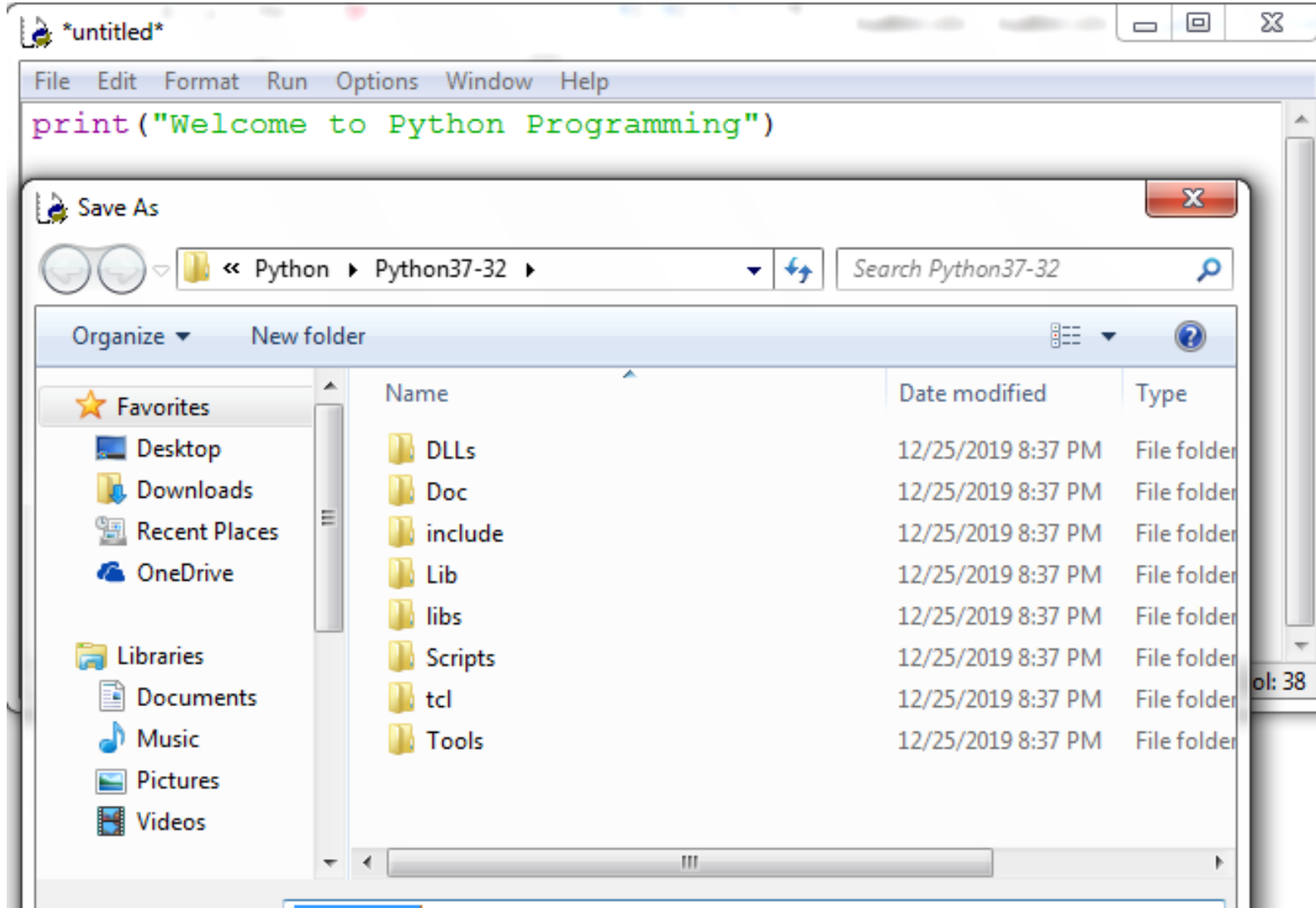
```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 3 Col: 4

Step 3: Click on File and open a new file.



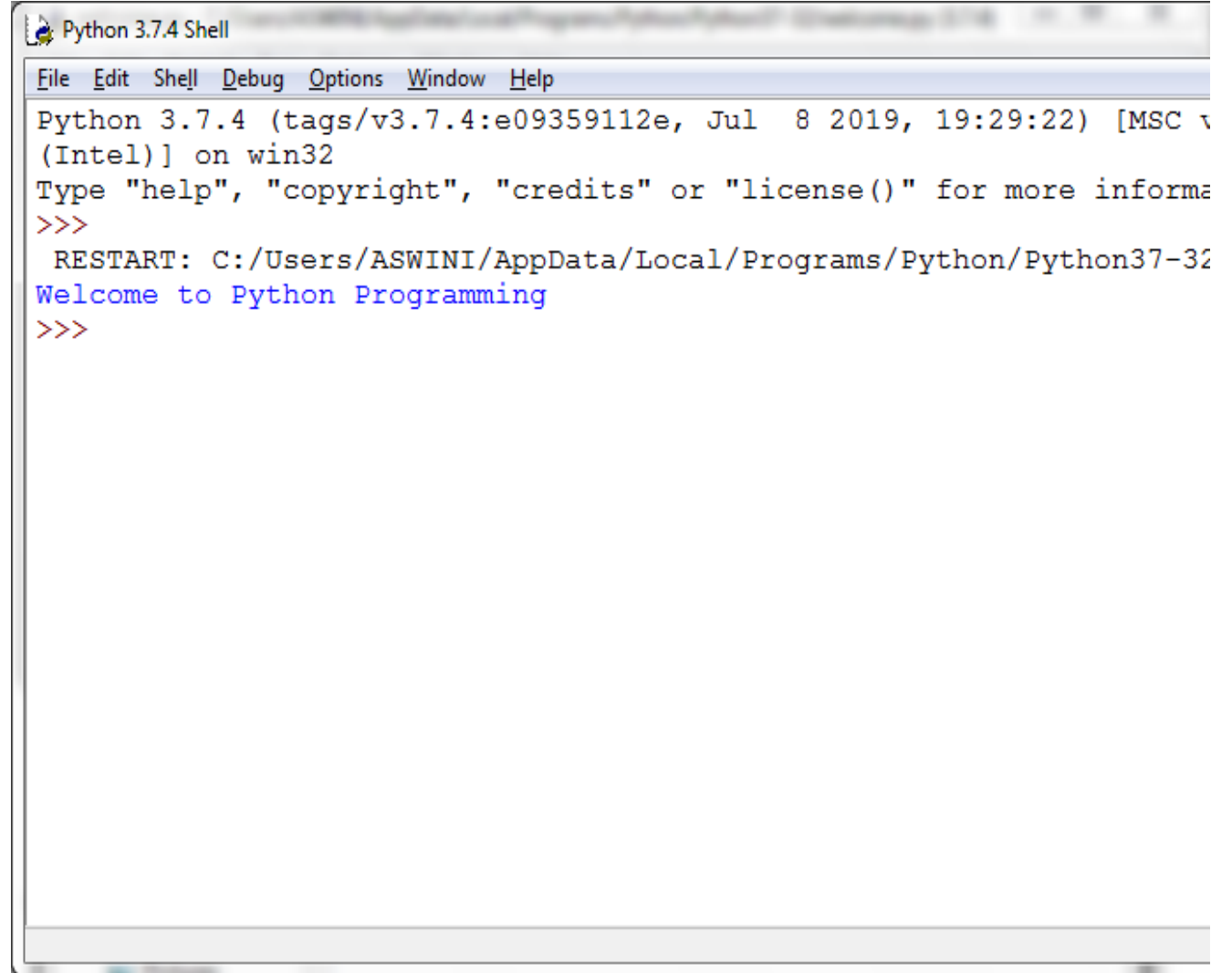
Step 4: Type the Python statements in the file and save the file



Store the file with the name "welcome.py", in the current working directory.

Step 5: Execute the Python program.

- Click on Run button in the menu bar and choose Run Module or Press F5 to execute the Python program “hello.py”.
- Print statement echoes the result in the Python shell after successful execution.



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more informa
>>>
  RESTART: C:/Users/ASWINI/AppData/Local/Programs/Python/Python37-32
Welcome to Python Programming
>>>
```


Comments in Python

- Comments are very important while writing a program.
- It describes what's going on inside a program so that a person looking at the source code does not have a hard time figuring it out.
- You might forget the key details of the program you just wrote in a month's time.
- So, taking time to explain these concepts in form of comments is always fruitful.
 - Single line comment
 - Multi line comment

Single Line Comment

- A hash sign (#) that is not inside a string literal begins a comment.
- All characters after the # and up to the physical line end are part of the comment and the Python interpreter ignores them.

Example:

Here you should write comments

print ("Hello, Python")

#This line is skipped by interpreter because it is after hash sign

Multiline comments

“” or ''' are used to write multi line comments

Example 1:

```
''' This is a multi line  
comment '''
```

Example 2:

```
“” This is a multi line  
comment “”
```

Quotations in Python

- Single (') quotes
- Double (") quotes
- Triple (''' or ''') quotes
- to denote string literals, as long as the same type of quote starts and ends the string.
- The triple quotes can be used to span the string across multiple lines.

```
word = 'word'  
sentence = "This is a sentence."  
paragraph = """This is a paragraph. It is  
made up of multiple lines and  
sentences."""
```

Statements

Python Instructions that a Python interpreter can execute are called statements. For example, `a = 1` is an assignment statement. `if` statement, `for` statement, `while` statement etc.

Multiline statements in Python

In Python, end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the line continuation character (`\`).

Example 1: explicit line continuation.(\)

```
Solution=4+6*\
```

```
2-\
```

```
9/2
```

Example2 - line continuation is implied inside parentheses (), brackets [] and braces { }

```
Student=[111,"Abu",
```

```
90]
```

Input and Output Statements

- In Python, Input and Output statements makes the program interactive with the user.
- The Input statements are used to take the user input from keyboard and Output statement is used to display the data to the screen/file.
- The built-in functions input (), eval() are the input statement .
- print() is an output statement.

print()

print() function displays the contents on the screen.

Syntax

```
print(argument)
```

Example:

```
print('This sentence is output to the screen')
```

Output:

This sentence is output to the screen

```
a = 5
```

```
print('The value of a is', a)
```

Output:

The value of a is 5

Syntax of Print()

`print(*objects, sep=' ', end='\n', file=sys.stdout)`

- Here, objects is the value(s) to be printed.
- The “sep” is a separator, used between the values. Its default value is space.
- After all values are printed, end is printed. Its default value is new line.
- The file is the object where the values are printed and its default value is sys.stdout (screen).

Example

```
print(1,2,3,4)
```

Output: 1 2 3 4

```
print(1,2,3,4,sep='*')
```

Output: 1*2*3*4

```
print(1,2,3,4,sep='#',end='&')
```

Output: 1#2#3#4&

Print-Formatting

```
>>> x = 12.3456789  
>>> print("The value of x is % .2f" %x)  
The value of x is 12.35
```

```
>>> print("%d is Integer "%10)  
10 is Integer  
  
>>> print("%.5s" %"Hello World")  
Hello
```

input()

input() function is used to accept an input from the user

- **Syntax**

input([prompt])

where prompt is the string we wish to display on the screen. It is optional.

Example

```
Name=input("Enter your name")
```

```
print("Hello", Name, "welcome to python programming")
```

```
>>> num = input('Enter a number: ')
Enter a number: 10
>>> num
'10'
```

```
>>> num = int(input('Enter a number: '))
Enter a number: 10
>>> num
10
```

entered value 10 is a string, not a number. To convert this into a number we can use int() or float() functions.

eval()

eval() is a built-in function that takes an expression(string) as an input and returns the result of the expression on evaluation

Example

```
>>> x=eval('12+25')
```

```
>>> x
```

```
37
```

```
>>> type(x)
```

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

eval() can be applied to input() function

```
>>> a=eval(input("Enter a number"))
```

```
Enter a number 5.65
```

```
>>> type(a)
```

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

Type Conversion:

- The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion. Python has two types of type conversion.
 - Implicit Type Conversion
 - Explicit Type Conversion

Implicit

In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

```
num_int = 123
num_flo = 1.23
num_new = num_int + num_flo
print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))
print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

Output:

```
datatype of num_int: <class 'int'>
datatype of num_flo: <class 'float'>
Value of num_new: 124.23
datatype of num_new: <class 'float'>
```

Type Conversion-Explicit

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like `int()`, `float()`, `str()`, etc to perform explicit type conversion.

This type conversion is also called typecasting because the user casts (changes) the data type of the objects

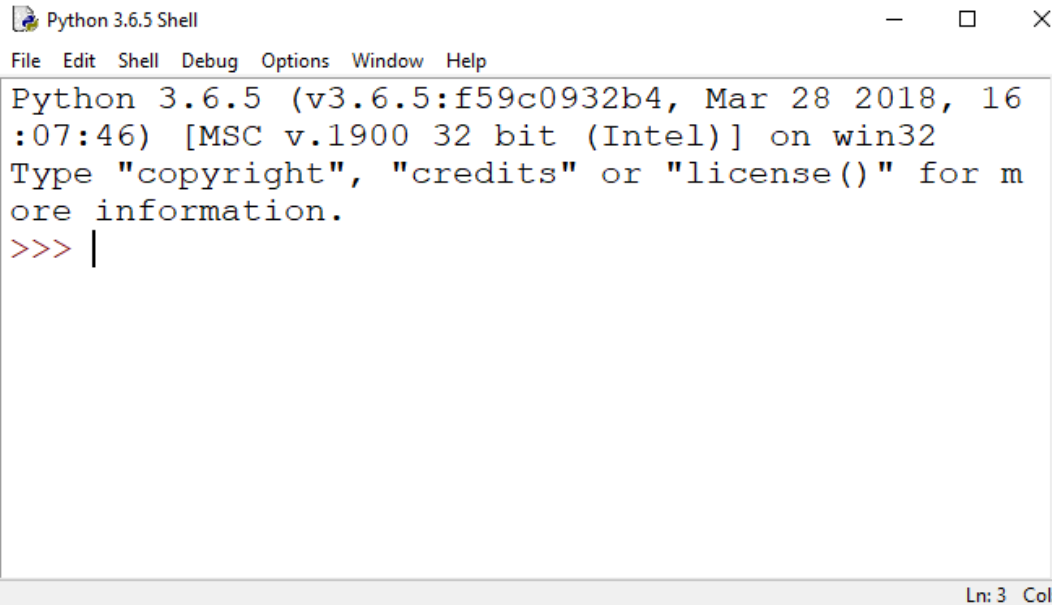
```
>>> test="456"
>>> type(test)
<class 'str'>
>>> test=int(test)
>>> test
456
>>> type(test)
<class 'int'>
```

Function	Description
<u>int</u> (a)	Converts 'a' to an integer
<u>long</u> (a)	Converts 'a' to a long integer
<u>float</u> (a)	Converts 'a' to a floating - point number
<u>complex</u> (real [, <u>imag</u>])	Creates a complex number
<u>str</u> (a)	Converts object 'a' to a string representation
<u>eval</u> (a)	Evaluates a string and returns an object
<u>tuple</u> (a)	Converts 'a' to a <u>tuple</u>
<u>list</u> (a)	Converts 'a' to a list.
<u>set</u> (a)	Converts 'a' to a set.
<u>dict</u> (a)	Creates a dictionary. 'a' must be a sequence of (key, value) <u>tuples</u> .

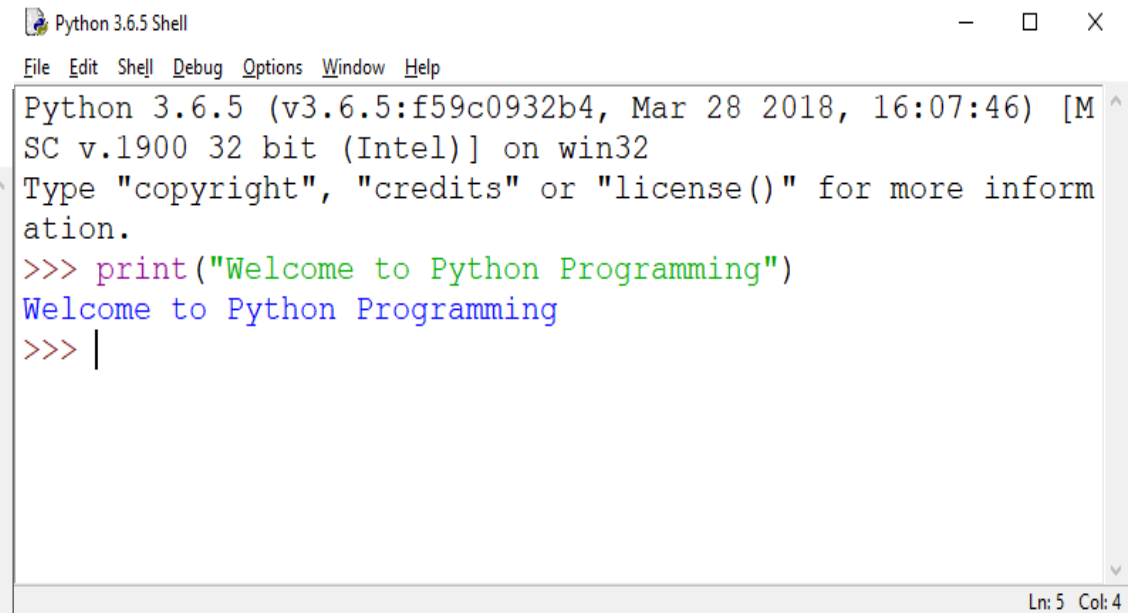
First program in Python

Firstly, we will start using the Python interpreter in IDLE

Simple print statement



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Welcome to Python Programming")
Welcome to Python Programming
>>> |
```

Comparison of python with java

Python v/s Java

A simple Program to print "Hello World"

Java Code	Python Code
<pre>public class HelloWorld { public static void main(String args []) { System.out.println ("Hello World!") } }</pre>	<pre>print ("Hello World!")</pre>

Write a Program to get the details of a student (name ,roll num, mark) given by the user at run time and display the details along with the datatype.

Program:

```
Name=input("Enter your name:")  
Roll_num=eval(input("Enter your roll number:"))  
mark=eval(input("Enter your mark:"))  
print(Name, type(Name))  
print(Roll_num,type(Roll_num))  
print(mark,type(mark))
```

Output:

```
Enter your name: Krishna  
Enter your roll number:10  
Enter your mark:99.9  
Krishna <class 'str'>  
10 <class 'int'>  
99.9 <class 'float'>
```


Python Tokens

- A program in Python contains a sequence of instructions.
- Python breaks each statement into a sequence of lexical components known as tokens.
- Types of Tokens are
 - Literals
 - Identifiers/Variables
 - Keywords
 - Operators
 - Delimiters

Literals

- Literals are numbers or strings or characters that appear directly in a program.
- Example:
 - 85 #Integer Literal
 - 20.50 #Floating point Literal
 - "Hello" #String Literal
 - True #Boolean Literal

Python Data Type

- All features in Python are associated with an object. It is one of the primitive elements of Python.
- All kinds of objects are classified into types.

Data Types:

- Numbers
 - Integer
 - Float
 - Complex
- Boolean
- String

Data Type-Integer

- Integer is a combination of positive and negative numbers including 0.
- integer literals are written without commas
- Leading minus sign to indicate a negative value
- Can be decimal(first digit is non zero), octal(0o) and hexadecimal(0x)

Example

```
>>> print("Decimal :", 10)
```

Decimal : 10

```
>>> print("Octal :",0o15)
```

Octal : 13

```
>>>
```

```
print("Hexadecimal:",0x1a)
```

Hexadecimal: 26

Data Type - Floating Point Number

- It consists of whole number, decimal point and fractional part.
- Can be written using a decimal notation or scientific notation

- **Example**

```
>>> 3.7
```

```
3.7
```

```
>>> 9.7e1
```

```
97.0
```

```
>>> 9.7e-1
```

```
0.97
```

Data Type – Complex number

- A complex number is a number that can be expressed in the form $a+bj$, where a, b are real numbers and j is an imaginary unit
 - Example

```
>>> 2+3j
(2+3j)
```

Data Type – Boolean & String

- **Boolean**

- Primitive data type having one of two values: True or False

- True-1, False-0

- **Example**

```
>>>True
```

```
True
```

```
>>> type(True)
```

```
<class 'bool'>
```

- **String**

- Can be created using single, double and triple quotes.

- **Example**

```
word = 'word'
```

```
sentence = "This is a sentence."
```

```
paragraph = """This is a paragraph. It is made up  
of multiple lines and sentences."""
```

Identifiers/Variable

- Identifier is the name used to find a variable, function, class or other objects.
- It starts with **underscore or alphabets**
- It **cannot be a keyword**
- **Case sensitive**
- No declaration of variables
- Data type of variable can change during program execution compared to other strongly typed languages such as c,c++,java

Best Practices for naming conventions:

- Variable name with lower case letter.
- Multiple words for a variable can be combined with underscore (_).
- For example: vowel_count is a variable to keep the count the number of vowels in a string.
- Give a meaningful name for a variable. For example, to store a factorial of a number, instead of giving **f** as variable name, you can give **factorial** as variable name.

LEARNING
IS
AUGMENTED

Declaring Variables in Python

- variables **do not need declaration** to reserve memory space.
- The "variable declaration" or "variable initialization" happens automatically when we assign a value to a variable.

Assigning values to a variable

- Variable=expression
- Example
 - Radius=5
 - Amount=1000
 - P=q=r=10 (value can be assigned to multiple variable)

Assigning multiple values to multiple variables

```
a, b, c = 5, 3.2, "Hello"  
print (a)  
print (b)  
print (c)
```

Output:

```
5  
3.2  
Hello
```

Memory Allocation for variables in Python



Python uses a different approach to store variables.



In Python, everything is an instance of a class (object).



Objects may be basic data types (int, float, strings etc.) or other standard datatype (list, string, tuple, set, dictionary etc.).



Instead of storing the values in the memory space reserved by the variable, Python reserves memory for the values.



Python automatically takes care of removing the value (Garbage collector) when no variable is tagged or using the value.

KCED Illustration of memory allocation in Python

Learning Augmented

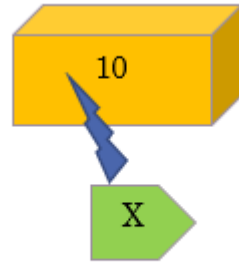
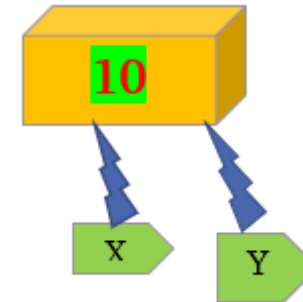


Illustration of X=10

Assigning the value 10 to another variable Y, creates a new tag which is bound to the same value 10.

- Consider the following statement,
`>>> x=10`
`>>> Y=X # Assign the value of`
- This statement creates another tag Y which refers to the same value 10.



Memory allocation- Demonstration

```
>>> X=10 # Value 10 is assigned to X
>>> id(X) # Address of the variable X
2084037536
>>> X=X+10 # Add the value 10 to X
>>> X # Value of X
20
>>> id(X) #Now X has the new value,it will have a different address locat
2084037696
>>> Y=X # Assign the value 20 to X
>>> id(Y) #Let's check the address of Y(Which will be same as X)
2084037696
>>> id(X) #Address of X
2084037696
```

Deleting a variable

- Syntax for the del statement is given below:

```
del variable_name1, variable_name2
```

```
>>> del iNum
>>> iNum
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    iNum
NameError: name 'iNum' is not defined
```

Points to remember:



Python treats everything is as an object. Objects may be basic data types (int, float, strings etc.) or other standard datatype (list, set, dictionary etc.).



Variables in Python are dynamically typed.



Only one copy of value exists in memory .

Values and Types

- Programming languages contain data in terms of input and output and any kind of data can be presented in terms of value. Value can be any form like literals containing numbers, characters and strings.

type() – to know the type of any value

```
>>> type('welcome')
```

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

```
>>> type(453)
```

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

```
>>> type(8.25)
```

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

```
>>> type(1+2j)
```

```
<class 'complex'>
```

```
>>> type(True)
```

```
<class 'bool'>
```

```
>>> a=56
```

```
>>> type(a)
```

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

id() function

- id() → returns address of object in memory
- The id() function returns identity of the object. This is an integer which is unique for the given object and remains constant during its lifetime.

Example

```
print('id of 5 =', id(5))
```

```
a = 5
```

```
print('id of a =', id(a))
```

```
b = a
```

```
print('id of b =', id(b))
```

```
c = 5.0
```

```
print('id of c =', id(c))
```

the output will be something like:

```
id of 5 = 140472391630016
```

```
id of a = 140472391630016
```

```
id of b = 140472391630016
```

```
id of c = 140472372786520
```

- Keywords are the reserved words in Python.
- Every programming language has set of keywords and these reserved words has special meaning to its interpreter.
- Keywords cannot be used as variable name, function name or as any identifier.

```
>>> 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']
```

Keywords in Python

S.NO	Keyword	Description	S.NO	Keyword	Description
1	and	A logical operator	18	import	To import a module
2	as	To create an alias object	19	in	To check the value present in any sequence (list, tuple etc)
3	assert	Used for debugging	20	is	To check if two variables are equal
4	break	To break out of loop	21	lambda	To create an anonymous function
5	class	To define user defined class	22	None	Represents a null value
6	continue	To continue next iteration of loop	23	nonlocal	To declare a non-local variable
7	def	To define a function	24	not	A logical operator
8	del	To delete objects	25	or	A logical operator
9	elif	Used in decision making.\nSimilar to elseif	26	pass	Used in control statements; a placeholder
10	else	Conditional statement	27	raise	To raise an exception
11	except	Used in exception	28	return	To return a value or exit from a function
12	False	Boolean value	29	True	Boolean value
13	finally	Used in exception	30	try	Exception statement
14	for	To start the loop	31	while	To create loop
15	from	To import a specific part from module	32	with	Used in exception handling
16	global	To create a global variable	33	yield	To end a function
17	if	Conditional statement			