Chapter 5
Getting Started with Python
(2 mcq+1 fib+2*1+3*1+5*2=18 Marks)

INTRODUCTION:

- Introduction to Python
- Python Keywords
- Identifiers
- Comments
- Data Types
- Operators
- Expressions
- Statement
- Input and Output
- Type Conversion
- Debugging

Python was created by Guido van Rossum, and first released on February 20, 1991. While you may know the python as a large snake, the name of the Python programming language comes from an old BBC television comedy sketch series called Monty Python's Flying Circus. Guido van Rossum is a Dutch programmer. Python is a programming as well as a scripting language(supports web development). Also, this language has various purposes such as developing, scripting, generation, and software testing, database connectivity. Google, Nokia, Disney, Yahoo, IBM use Python.
Python works in Windows, Linux, MacOS Operating systems.
Python can be used as a front end as well as back end.

Program: An ordered set of instructions to be executed by a computer to carry out a specific task is called a program.

Programming language: The language used to specify the set of instructions to the computer is called a programming language.

Machine language: Computers understand the language of 0s and 1s which is called machine language or low level language.

High-level programming languages: in HLL the programs are written using simple English. For example Python, C++, Visual Basic, PHP, and Java that is easier to manage by humans but is not directly understood by the computer.

Source code: A program written in a high-level language is called source code.

Language translators: Language translators perform the translation of the source code into machine language. There are two types of translators: *Compilers* and *interpreters.*

Compiler: Compiler translates the entire source code, as a whole, into the object code (Machine language). After scanning the whole program, it generates error messages, if any.
Interpreter: An interpreter processes the program statements one by one, first translating and then executing. This process is continued until an error is encountered or the whole program is executed successfully.

Python uses an interpreter to convert its instructions into machine language, so that it can be understood by the computer.
Python is a General Purpose high level Programming language used for developing application software.

Features of Python

- Python is a high level language. It is a free and open source language.
- It is an interpreted language, as Python programs are executed by an interpreter.
- Python programs are easy to understand as they have a clearly defined syntax and relatively simple structure.
- Python is case-sensitive. For example, NUMBER and number are not same in Python.
- Python is portable and platform independent, means it can run on various operating systems and hardware platforms.
- Python has a rich library of predefined functions.
- Python is also helpful in web development. Many popular web services and applications are built using Python.
- Python uses indentation for blocks and nested blocks.

Working with Python:

- To write and run (execute) a Python program, we need to have a Python interpreter installed on our computer or we can use any online Python interpreter.
- The interpreter is also called Python shell.
- The symbol >>> is the Python prompt, which indicates that the interpreter is ready to take instructions.
- We can type commands or statements on this prompt to execute them using a Python interpreter.

Execution Mode: There are two ways to use the Python interpreter:
a) Interactive mode b) Script mode

Interactive mode:

- Interactive mode allows execution of individual statement instantaneously.
- We can simply type a Python statement on the >>> prompt directly.
- As soon as we press enter, the interpreter executes the statement and displays the result.
- Working in the interactive mode is convenient for testing a single line code for instant execution.
- But in the interactive mode, we cannot save the statements for future use and we have to retype the statements to run them again.

Script mode:

- In the script mode, we can write a Python program in a file, save it and then use the interpreter to execute it.
- Python scripts are saved as files where file name has extension ".py".
- By default, the Python scripts are saved in the Python installation folder.
- To execute a script, we can either:
  a) Type the file name along with the path at the prompt. For example, if the name of the file is prog5-1.py, we type prog5-1.py. We can otherwise open the program directly from IDLE.
  b) While working in the script mode, after saving the file, click [Run]->[Run Module] from the menu
  c) The output appears on shell.

Write a program to show print statement in script mode.

print(“Save Earth”)

Python Keywords:

- Keywords are reserved words.
- Each keyword has a specific meaning to the Python interpreter, and we can use a keyword in our program only for the purpose for which it has been defined.
- As Python is case sensitive, keywords must be written exactly.

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

Identifiers:

In programming languages, identifiers are names used to identify a variable, function, or other entities in a program.

The rules for naming an identifier in Python are as follows:

- The name should begin with an uppercase or a lowercase alphabet or an underscore sign (_).Thus, an identifier cannot start with a digit.
- This may be followed by any combination of characters a–z, A–Z, 0–9 or underscore (_).
- It can be of any length. (However, it is preferred to keep it short and meaningful).
- It should not be a keyword or reserved word
- We cannot use special symbols like!, @, #, $, %, etc., in identifiers.

Variables:

- A variable in a program is uniquely identified by a name (identifier).
- Variable in Python refers to an object — an item or element that is stored in the memory.
- Value of a variable can be a string (e.g., ‘b’, ‘Global Citizen’), numeric (e.g., 345) or any combination of alphanumeric characters (CD67).
- In Python we can use an assignment statement to create new variables and assign specific values to them.

```
gender = 'M'
message = "Keep Smiling"
price = 987.9
```

Write a program to display values of variables in Python.

```
message = "Keep Smiling"
print(message)
userNo = 101
print('User Number is', userNo)
```

Output: Keep Smiling
User Number is 101

Write a Python program to find the area of a rectangle given that its length is 10 units and breadth is 20 units.

```
#To find the area of a rectangle
length = 10
breadth = 20
area = length * breadth
print(area)
```

Comments:

- Comments are used to add a remark or a note in the source code.
- Comments are not executed by interpreter.
- Comment starts with # (hash sign).
- Everything following the # till the end of that line is treated as a comment.

Example:

```
#Variable amount is the total spending on
#grocery amount = 3400
```

Everything is an Object:

- Python treats every value or data item whether numeric, string, or other type as an object.
- Every object in Python is assigned a unique identity (ID) which remains the same for the lifetime of that object.
- This ID is akin (similar) to the memory address of the object. The function id ( ) returns the identity of an object.

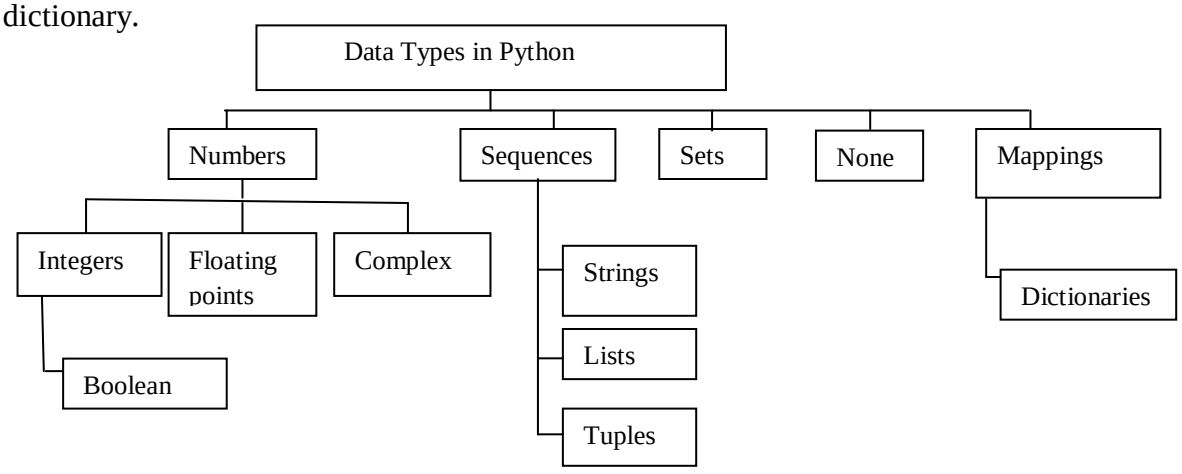
Example

```
num1 = 20
id(num1)
1433920576   #identity of num1

num2 = 30 – 10
id(num2)
1433920576        #identity of num2 and num1 are same as both refers to object 20
```

Data Types: Data type identifies the type of data values a variable can hold and the operations that can be performed on that data.

There are several data types in Python — integer, boolean, float, complex, string, list, tuple, sets, None and dictionary.



1. Number:

- Number data type stores numerical values only.
- It is further classified into three different types: int, float and complex.

Type/Class	Description	Examples
int	integer numbers	–12, –3, 0, 125, 2
float	real or floating point numbers	–2.04, 4.0, 14.23
complex	complex numbers	3 + 4j, 2 – 2j

Boolean data type:

- Boolean (bool) is a subtype of integer.
- It is a unique data type, consisting of two constants, True and False.
- Boolean True value is non-zero, non-null and non-empty. Boolean False is the value zero.

Example:

```
 num1 = 10
 type(num1)
<class 'int'>

 num2 = -1210
 type(num2)
<class 'int'>

 var1 = True
 type(var1)
<class 'bool'>

 float1 = -1921.9
 type(float1)
<class 'float'>

 float2 = -9.8*10**2
 print(float2, type(float2))
-980.0000000000001 <class 'float'>

 var2 = -3+7.2j
 print(var2, type(var2))
(-3+7.2j) <class 'complex'>
```

2. Sequence:

- A Python sequence is an ordered collection of items, where each item is indexed by an integer.
- The three types of sequence data types available in Python are Strings, Lists and Tuples.

a. String:

String is a group of characters.
These characters may be alphabets, digits or special characters including spaces.
String values are enclosed either in single quotation marks (‘Hello’) or in double quotation marks (“Hello”).

The quotes are not a part of the string, they are used to mark the beginning and end of the string for the interpreter.

For example:

```
str1 = 'Hello Friend'
str2 = "452"
```

We cannot perform numerical operations on strings, even when the string contains a numeric value, as in str2.

b. List: List is a sequence of items separated by commas and the items are enclosed in square brackets []

Example:

```
#To create a list
list1 = [5, 3.4, “New Delhi”, ‘20C’, 45]
#print the elements of the list list1

print(list1)
[5, 3.4, “New Delhi”", '20C', 45]

#to change the list item value
List1[2]=’Mumbai’
```

c. Tuple:

- Tuple is a sequence of items separated by commas and items are enclosed in parenthesis ( ).
- This is unlike list, where values are enclosed in brackets [ ].
- Once created, we cannot change the tuple.

Example:

```
#create a tuple tuple1
tuple1 = (10, 20, "Apple", 3.4, 'a')
#print the elements of the tuple tuple1

print(tuple1)
(10, 20, "Apple", 3.4, 'a')
```

3. Set:

- Set is an unordered collection of items separated by commas and the items are enclosed in curly brackets { }.
- A set is similar to list, except that it cannot have duplicate entries.
- Once created, elements of a set cannot be changed.

Example:

```
#create a set
set1 = {10, 20, 3.14,"New Delhi"}
print (type (set1))
print (set1)
{10, 20, 3.14, "New Delhi"}

#duplicate elements are not included in set
set2 = {1, 2, 1, 3}
```

```
print(set2)
{1, 2, 3}
```

4. None:

None is a special data type with a single value.
It is used to signify the absence of value in a situation.
None supports no special operations, and it is neither False nor 0 (zero).

Example:

```
myVar = None
print(type(myVar))
print(myVar)
None
```

5. Mapping:

- Mapping is an unordered data type in Python.
- Currently, there is only one standard mapping data type in Python called dictionary.

Dictionary:

- Dictionary in Python holds data items in key-value pairs.
- Items in a dictionary are enclosed in curly brackets { }.
- Dictionaries permit faster access to data.
- Every key is separated from its value using a colon (:) sign.
- The key: value pairs of a dictionary can be accessed using the key.
- The keys are usually strings and their values can be any data type.
- In order to access any value in the dictionary, we have to specify its key in square brackets [ ].

Example:

```
#create a dictionary
 dict1 = {'Fruit':'Apple', 'Climate':'Cold', 'Price':'120'}
 print(dict1)  # Displays as {'Fruit': 'Apple', 'Climate': 'Cold', 'Price': '120'}

 print(dict1['Price'])  #prints 120

dict1['Price']=200 #change the price value to 200
print(dict1)  # Displays as {'Fruit': 'Apple', 'Climate': 'Cold', 'Price': '200'}
```

Mutable and Immutable Data Types:

Python data types can be classified into mutable and immutable.

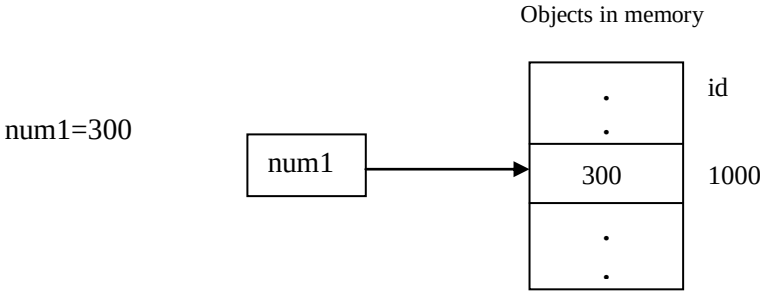
- Variables whose values can be changed after they are created and assigned are called mutable.
- Mutable data types: Lists, Dictionary

- Variables whose values cannot be changed after they are created and assigned are called immutable.
- When an attempt is made to update the value of an immutable variable, the old variable is destroyed and a new variable is created by the same name in memory.
- Immutable data types: int, float, bool, complex, string, tuple, set

When an attempt is made to update the value of a variable:

num1 = 300

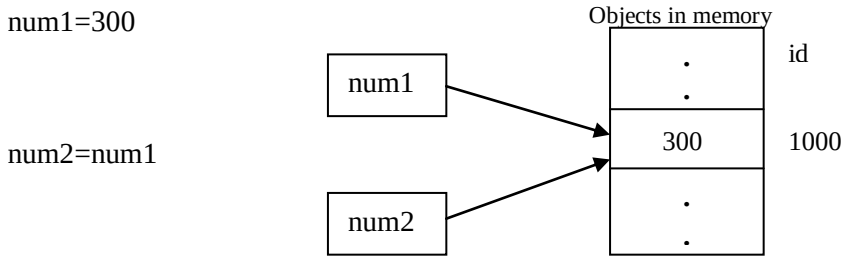
This statement will create an object with value 300 and the object is referenced by the identifier num1.



num2 = num1

The statement will make num2 refer to the value 300, also being referred by num1, and stored at memory location number, say 1000.

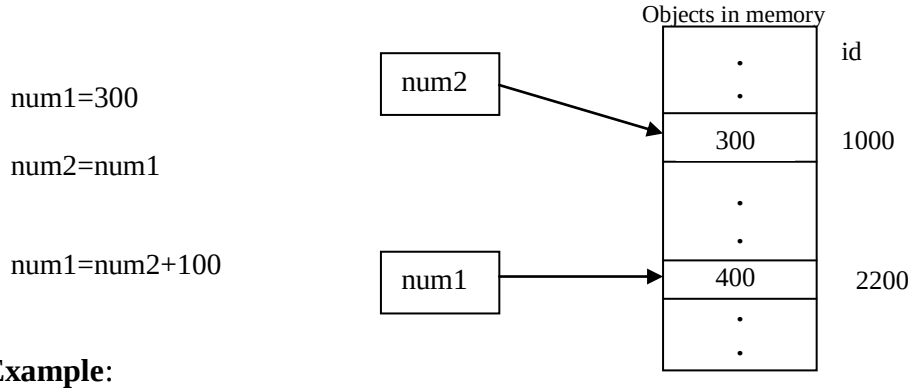
So, num1 shares the referenced location with num2. In this manner Python makes the assignment effective by copying only the reference, and not the data.



num1 = num2 + 100

This statement links the variable num1 to a new object stored at memory location number say 2200 having a value 400.

As num1 is an integer, which is an immutable type, it is rebuilt,



Example:

num1=300
print(id(num1)) #Returns 2832845547312

num2=num1
print(id(num2)) #Returns 2832845547312

num1=num2+100

print(id(num1)) #Returns 2832845547152

print(id(num2)) #Returns 2832845547312

Operators:
An operator is used to perform specific mathematical or logical operation on values.
The values that the operators work on are called operands.
For example, in the expression 10 + num, the value 10, and the variable num are operands and the + (plus) sign is an operator.

Types of operators: Mainly **t**here are two types of operators.
Unary operators and binary operators.
The unary operators need only one operand, and they have a higher precedence than the binary operators.
+ Unary plus, - unary minus, **not** logical negation are the different unary operators.

Binary operators: Binary operators are operators with two operands.
The different binary operators are:

1. Arithmetic Operators:

Operator	Operation	Description	Example
+	Addition	Adds the two numeric values on either side of the operator. This operator can also be used to concatenate two strings on either side of the operator	num1 = 5 num2 = 6 num1 + num2 11 str1 = "Hello" str2 = "India" str1 + str2 'HelloIndia'
-	Subtraction	Subtracts the operand on the right from the operand on the left	num1 = 5 num2 = 6 num1 - num2 -1
*	Multiplication	Multiplies the two values on both side of the operator. Repeats the item on left of the operator if first operand is a string and second operand is an integer value	num1 = 5 num2 = 6 num1 * num2 30 str1 = 'India' str1 * 2 'IndiaIndia'
/	Division	Divides the operand on the left by the operand on the right and returns the quotient	num1 = 8 num2 = 4 num2 / num1 0.5
%	Modulus	Divides the operand on the left by the operand on the right and returns the remainder	num1 = 13 num2 = 5 num1 % num2

			3
//	Floor Division	Divides the operand on the left by the operand on the right and returns the quotient by removing the decimal part. It is sometimes also called integer division.	num1 = 13 num2 = 4 num1 // num2 3 num2 // num1 0
**	Exponent	Performs exponential (power) calculation on operands. That is, raise the operand on the left to the power of the operand on the right	num1 = 3 num2 = 4 num1 ** num2 81

2. Relational Operators

Relational operator compares the values of the operands on its either side and determines the relationship among them. Assume the Python variables num1 = 10, num2 = 0, num3 = 10, str1 = "Good", str2 = "Afternoon"

Operator	Operation	Description	Example
= =	Equals to	If the values of two operands are equal, then the condition is True, otherwise it is False	num1 == num2 False
			>> str1 == str2 False
!=	Not equal to	If values of two operands are not equal, then condition is True, otherwise it is False	num1 != num2 True
			str1 != str2 True
			num1 != num3 False
>	Greater than	If the value of the left-side operand is greater than the value of the right side operand, then condition is True, otherwise it is False	num1 > num2 True
			str1 > str2 True
<	Less than	If the value of the left-side operand is less than the value of the right side operand, then condition is True, otherwise it is False	num1 < num3 False
			str1 < str2 False
>=	Greater than or equal to	If the value of the left-side operand is greater than or equal to the value of the right-side operand, then condition is True, otherwise it is False	num1 >= num2 True
			num2 >= num3 False
			str1 >= str2 True
<=	Less than or equal to	If the value of the left operand is less than or equal to	` num1 <= num2 False

		the value of the right operand, then is True otherwise it is False	num2 <= num`3 True
			str1 <= str2 False

3. Assignment Operators

Assignment operator assigns or changes the value of the variable on its left.

Operator	Description	Example
=	Assigns value from right-side operand to left side operand	num1 = 2 num2 = num1 num2 2
		country = 'India' country 'India'
+=	It adds the value of right-side operand to the left-side operand and assigns the result to the left-side operand. Note: x += y is same as x = x + y	num1 = 10 num2 = 2 num1 += num2 num1 12 num2 2
		str1 = 'Hello' str2 = 'India' str1 += str2 str1 'HelloIndia'
-=	It subtracts the value of right-side operand from the left-side operand and assigns the result to left-side operand Note: x -= y is same as x = x - y	num1 = 10 num2 = 2 num1 -= num2 num1 8
*=	It multiplies the value of right-side operand with the value of left-side operand and assigns the result to left-side operand Note: x *= y is same as x = x * y	num1 = 2 num2 = 3 num1 *= num2 num1 6
		a = 'India' a *= 3 a 'IndiaIndiaIndia'
/=	It divides the value of left-side operand by the value of right-side operand and assigns the result to left-side operand. Note: x /= y is same as x = x / y	num1 = 6 num2 = 3 num1 /= num2

		num1 2.0
%=	It performs modulus operation using two operands and assigns the result to left-side operand. Note: x %= y is same as x = x % y	num1 = 7 num2 = 3 num1 %= num2 num1 1
//=	It performs floor division using two operands and assigns the result to left-side operand Note: x //= y is same as x = x // y	num1 = 7 num2 = 3 num1 //= num2 num1 2
**=	It performs exponential (power) calculation on operands and assigns value to the left-side operand. Note: x **= y is same as x = x ** y	num1 = 2 num2 = 3 num1 **= num2 num1 8

4. Logical Operators:

- There are three logical operators supported by Python.
- These operators (and, or, not) are to be written in lower case only.
- The logical operator evaluates to either True or False based on the logical operands on either side.
- Every value is logically either True or False.
- By default, all values are True except None, False, 0 (zero), empty collections "", (), [], {}, and few other special values.
- So if we say num1 = 10, num2 = -20, then both num1 and num2 are logically True.

Operator	Operation	Description	Example
and	Logical AND	If both the operands are True, then condition becomes True	True and True True
			num1 = 10 num2 = -20 bool(num1 and num2) True
			True and False False
			num3 = 0 bool(num1 and num3) False
			False and False False
or	Logical OR	If any of the two operands are True, then condition becomes True	True or True True
			True or False True

			bool(num1 or num3) True
			False or False False
not	not	Used to reverse the logical state of its operand	num1 = 10 bool(num1) True
			not num1 bool(num1) False

5. Identity Operators:

Identity operators are used to determine whether the value of a variable is of a certain type or not. Identity operators can also be used to determine whether two variables are referring to the same object or not. There are two identity operators.

Operator	Description	Example
is	Evaluates True if the variables on either side of the operator point towards the same memory location and False otherwise. var1 is var2 results to True if id(var1) is equal to id(var2)	num1 = 5 type(num1) is int True
		num2 = num1 id(num1) 1433920576
		id(num2) 1433920576 num1 is num2 True
is not	Evaluates to False if the variables on either side of the operator point to the same memory location and True otherwise. var1 is not var2 results to True if id(var1) is not equal to id(var2)	num1=6 num2=7 num1 is not num2 True

6. Membership Operators

Membership operators are used to check if a value is a member of the given sequence or not.

Operator	Description	Example
in	Returns True if the variable/value is found in the specified sequence and False otherwise	a = [1,2,3] 2 in a True
		'1' in a False
not in	Returns True if the variable/value is not found in the specified sequence and False otherwise	a = [1,2,3] 10 not in a True
		1 not in a False

Expressions:

An expression is defined as a combination of constants, variables, and operators.

An expression always evaluates to a value.

A value or a standalone variable is also considered as an expression but a standalone operator is not an expression.

Some examples of valid expressions are given below:

(i) 100 (ii) num (iii) num – 20.4 (iv) 3.0 + 3.14 (v) 23/3 -5 * 7(14 -2) (vi) "Global" + "Citizen"

Precedence of Operators:

The order in which the different types of operators are evaluated is called as operator precedence. It is also known as hierarchy of operators.

When an expression contains different kinds of operators, precedence determines which operator should be applied first.

Higher precedence operator is evaluated before the lower precedence operator.

Order of Precedence	Operators	Description
1	**	Exponentiation (raised to the power)
2	~ ,+, -	Complement, unary plus and unary minus
3	* ,/, %, //	Multiply, divide, modulo and floor division
4	+, -	Addition and subtraction
5	<= ,< ,> ,>=	Relational operators
6	== ,!=	Equality operators
7	=, %=, /=, //=, -=, +=, *=, **=	Assignment operators
8	is, is not	Identity operators
9	in, not in	Membership operators
10	not, or, and	Logical operators

Note:

a) Parenthesis can be used to override the precedence of operators. The expression within ( ) is evaluated first.

b) For operators with equal precedence, the expression is evaluated from left to right.

Example:

1. 20 + 30 * 40 Sol: =20 + (30 * 40) 20 + 1200 = 1220	2. 15.0 / 4 + (8 + 3.0) Sol: = 15.0 / 4 + (8.0 + 3.0) = (15.0 / 4.0)+ 11.0 = 3.75 + 11.0 =14.75

Statement: A statement is a unit of code that the Python interpreter can execute.

Example:

```
x = 4 #assignment statement
cube = x ** 3 #assignment statement
print (x, cube) #print statement
```

Input and Output functions:

input():

- In Python, we have the input() function for taking the user input.
- The input() function prompts the user to enter data.
- It accepts all user input as string.
- The user may enter a number or a string but the input() function treats them as string and assigns it to the variable on left-hand side of the assignment operator (=).

The **syntax** for input() is: input ([Prompt])
Prompt is the string we may like to display on the screen prior to taking the input and it is optional. When a prompt is specified, first it is displayed on the screen after which the user can enter data.

Entering data for the input function is terminated by pressing the enter key.

Example:

```
fname = input("Enter your first name: ")
Enter your first name: Arnab

age = input("Enter your age: ")
Enter your age: 19

print(type(age))
<class 'str'>
```

 # function int() to convert string to integer.

```
age = int( input("Enter your age:"))
Enter your age: 19
print(type(age))
<class 'int'>
```

output():

- Python uses the print() function to output data to standard output device — the screen.
- The function print() evaluates the expression before displaying it on the screen.
- The print() outputs a complete line and then moves to the next line for subsequent output.

The **syntax** for print() is:

```
print(value [, ..., sep = ' ', end = '\n'])
```

- sep: The optional parameter sep is a separator between the output values. We can use a character, integer or a string as a separator. The default separator is space.
  Ex: print("Program","Python",sep=":"). Output: Program: Python
- end: This is also optional and it allows us to specify any string to be appended after the last value. The default is a new line.

Examle:
print("Combination", end=":")
print(“PCMC’')

Output:
Combination= PCMC

Example:

Statement	**Output**
print(“Hello”)	Hello
print(10*2.5)	25.0
print("I" + "love" + "my" + "country")	Ilovemycountry
print("I'm", 16, "years old")	I'm 16 years old

We use + (plus) between two strings to concatenate them. The fourth print function also appears to be concatenating strings but uses commas (,) between strings. Actually, here we are passing multiple arguments, separated by commas to the print function. As arguments can be of different types, hence the print function accepts integer (16) along with strings here. But in case the print statement has values of different types and ‘+’ is used instead of comma, it will generate an error.

Example:
a=10
print(“hello” + a) # displays error

print(“hello”, a) #shows output as hello 10

Type Conversion:
Changing the data type of a variable in Python from one type to another is called as type conversion. There are two types.

a. Explicit conversion (type casting)
b. Implicit conversion

Explicit Conversion (type casting): In explicit conversion, programmer specifies/forces the interpreter to convert from one data type to another data type.

Syntax:

(new_data_type) (expression)

With explicit type conversion, there is a risk of loss of information since we are forcing an expression to be of a specific type.

Example:

x = 20.67
int(x) #will discard the fractional part .67 and displays 20

Implicit Conversion: Implicit conversion, also known as coercion, happens when data type conversion is done automatically by Python and is not instructed by the programmer.

#Program to show implicit conversion from int to float.

num1 = 10 #num1 is an integer
num2 = 20.0 #num2 is a float

```
sum1 = num1 + num2  #sum1 is sum of a float and an integer
print(sum1)
print(type(sum1))
```

Output:

```
30.0
<class 'float'>
```

In the above example, an integer value stored in variable num1 is added to a float value stored in variable num2, and the result was automatically converted to a float value stored in variable sum1 without explicitly telling the interpreter.

Following are some of the functions in Python that are used for explicitly converting an expression or a variable to a different type.

Function	Description
int(x)	Converts x to an integer
float(x)	Converts x to a floating-point number
str(x)	Converts x to a string representation
chr(x)	Converts x to a character
unichr(x)	Converts x to a Unicode character

#Program of explicit type conversion from int to float.

```
num1 = 10
num2 = 20
num3 = num1 + num2
print(num3)
print(type(num3))
num4 = float(num1 + num2)
print(num4)
print(type(num4))
```

Output:

```
30
<class 'int'>
30.0
<class 'float'>
```

#Program of explicit type conversion from float to int.

```
num1 = 10.2
num2 = 20.6
num3 = (num1 + num2)
print(num3)
print(type(num3))
num4 = int(num1 + num2)
print(num4)
print(type(num4))
```

Output:

```
30.8
<class 'float'>
30
<class 'int'>
```

#Example of type conversion between numbers and strings.

```
price1 = 25
price2 = 45
totalPrice = price1 + price2
print("The total is Rs." + totalPrice )
```

On execution, the above program gives an error, informing that the interpreter cannot convert an integer value to string implicitly.
However, the interpreter may not decide on its own when to convert as there is a risk of loss of information.
Python provides the mechanism of the explicit type conversion so that one can clearly state the desired outcome.

#Program works perfectly using explicit type casting:

```
priceIcecream = 25
priceBrownie = 45
totalPrice = priceIcecream + priceBrownie
print("The total in Rs." + str(totalPrice))
```

Output: The total in Rs.70

Similarly, type casting is needed to convert float to string. In Python, one can convert string to integer or float values whenever required.

#Explicit type conversion string to integer

```
icecream = '25'
fruit = '45'
price = icecream + fruit                    #String concatenation
print("Total Price Rs." + price)            # Total Price Rs.2545
price = int(icecream)+int(fruit)
print("Total Price Rs." + str(price))
```

Output: Total Price Rs.2545
Total Price Rs.70

Debugging: The process of identifying and removing mistakes, also known as bugs or errors, from a program is called debugging.
Errors occurring in programs can be categorized as: i) Syntax errors ii) Logical errors iii) Runtime errors

Syntax Errors: Errors that occur due to the wrong use of grammatical rules (syntax) is called as syntax error. For example, parentheses must be in pairs, so the expression (10 + 12) is syntactically correct, whereas (7 + 11 is not due to absence of right parenthesis. Such errors need to be removed before the execution of the program.

Logical Errors (**semantic errors**)**:** A logical error is a bug in the program that causes incorrect output. A logical error produces an undesired output but without termination of the execution of the program.

For example, if we wish to find the average of two numbers 10 and 12 and we write the code as 10 + 12/2, it would run successfully and produce the result 16. Surely, 16 is not the average of 10 and 12. The correct code to find the average should have been (10 + 12)/2 to give the correct output as 11.
Logical errors are also called semantic errors as they occur when the meaning of the program (its semantics) is not correct.

Runtime Error: A runtime error causes abnormal termination of program while it is executing. Runtime error is when the statement is correct syntactically, but the interpreter cannot execute it.
For example, we have a statement having division operation in the program. By mistake, if the denominator entered is zero then it will give a runtime error like "division by zero".

Example:
print(12/0) # will give a runtime error like division by zero

Exercise:
1. Which of the following identifier names are invalid and why?
Serial_no.
1st_Room
Hundred$
Total_Marks
total-Marks
Total Marks
True _Percentag

i) Serial_no. :- Invalid
Reason- (.) is not allowed in identifier
ii) 1st_Room :- Invalid
Reason- identifier can't start with number
iii) Hundred$:- Invalid
Reason- We can't use special symbol $ in identifier
iv) Total Marks :- Invalid
Reason- We can't use space between in identifier
v) Total_Marks :- Valid
Reason- We can use underscore between in identifier
vi) total-Marks :- Invalid
Reason- We can't use hyphen between in identifier
vii) _Percentage:- Invalid
Reason- Identifier can't begin with underscore
viii) True :- Invalid
Reason- We can't use keyword as a identifier

2. Write the corresponding Python assignment statements:
a) Assign 10 to variable length and 20 to variable breadth.
length = 10
breadth = 20

b) Assign the average of values of variables length and breadth to a variable sum.
Sum = (length + breadth)/2

c) Assign a list containing strings 'Paper', 'Gel Pen', and 'Eraser' to a variable stationery.
Stationary=['Paper' , 'Gel Pen' , 'Eraser']

d) Assign the strings 'Mohandas', 'Karamchand', and 'Gandhi' to variables first, middle and last.
first = 'Mohandas'
middle= 'Karamchand'
last= 'Gandhi'

e) Assign the concatenated value of string variables first, middle and last to variable fullname. Make sure to incorporate blank spaces appropriately between different parts of names.
fullname = first +" "+ middle +" "+ last

3. Write logical expressions corresponding to the following statements in Python and evaluate the expressions (assuming variables num1, num2, num3, first, middle, last are already having meaningful values):

STATEMENT	LOGICAL EXPRESSIONS
The sum of 20 and –10 is less than 12.	(20 + (-10)) < 12
num3 is not more than 24.	num3 <= 24 or not(num3 > 24)
6.75 is between the values of integers num1 and num2.	(6.75 >= num1) and (6.75 <= num2)
The string 'middle' is larger than the string 'first' and smaller than the string 'last'.	(middle > first) and (middle < last)
List Stationery is empty.	len(Stationery) == 0

4. Add a pair of parentheses to each expression so that it evaluates to True.

EXPRESSION	EXPRESSION WITH PARENTHESIS
0 == 1 == 2	(0 == (1 == 2))
2 + 3 == 4 + 5 == 7	(2 + (3 == 4)+ 5) == 7
1 < -1 == 3 > 4	(1 < -1) == (3 > 4)

5. Write the output of the following:

a) num1 = 4
num2 = num1 + 1
num1 = 2
print (num1, num2)
Output: 2, 5

b) num1, num2 = 2, 6
num1, num2 = num2, num1 + 2
print (num1, num2)
Output: 6,4

c) num1, num2 = 2, 3
num3, num2 = num1, num3 + 1
print (num1, num2, num3)
Output: error

6. Which data type will be used to represent the following data values and why?

DATA VALUES	DATA TYPE	REASON
Number of months in a year	Integer	Number of months contains only number values
Resident of Delhi or not	Boolean	It gives the answer in the form of true and false
Mobile number	Integer	Mobile number only contain integer number value
Pocket money	Float	Money can be count as 100rs 50 paisa(100.50)
Volume of a sphere	Float	The volume can be calculated in the decimal point
Perimeter of a square	Float	The perimeter can be calculated in the decimal point
Name of the student	String	Name is a set of character, hence data type of name of student is string
Address of the student	String	Address is a set of character, hence data type of address of student is string

7. Give the output of the following when num1 = 4, num2 = 3, num3 = 2

Answer:

a) num1 += num2 + num3
print (num1)
Output:9

b) num1 = num1 ** (num2 + num3)
print (num1)
Output:1024

c) num1 **= num2 + num3
Output:1024

d) num1 = '5' + '5'
print(num1)
Output:55

e) print(4.00/(2.0+2.0))
Output:1.0

f) num1 = 2+9*((3*12)-8)/10
print (num1)
Output:27.2

g) num1 = 24 // 4 // 2
print(num1)
Output:3

h) num1 = float(10)
print (num1)
Output:10.0

i) num1 = int('3.14')
print (num1)
Output: error

j) print('Bye' == 'BYE')
Output: False

k) print(10 != 9 and 20 >= 20)
Output:True

l) print(10 + 6 * 2 ** 2 != 9//4 -3 and 29
>= 29/9)
Output: True

m) print(5 % 10 + 10 < 50 and 29 <= 29)
Output: True

n) print((0 < 6) or (not(10 == 6) and
(10<0)))
Output: True

8. Categorise the following as syntax error, logical error or runtime error:

a) 25 / 0 :- Runtime Error, because of divisible by zero
b) num1 = 25; num2 = 0; num1/num2 :- Runtime Error, because of divisible by zero

9. Write a Python program to convert temperature in degree Celsius to degree Fahrenheit (Hint: T(°F) = T(°C) × 9/5 + 32)

```
c=float(input("Enter temp in Celsius: ")
f=c*9/5 + 32
print(f)
```

10. Write a Python program to calculate the amount payable if money has been lent on simple interest. Simple Interest (SI) = (P x R x T)/ 100. Amount payable = Principal + SI. P, R and T are given as input to the program.

```
principal=int(input("P="))
rate=float(input("R="))
time=int(input("T="))
simple_intrest=(principal*rate*time)/100
total_amount= principal +simple_intrest
print("Simple interest=", simple_intrest)
print("Total Payble amount", total_amount)
```

11. Write a program to calculate in how many days a work will be completed by three persons A, B and C together. A, B, C take x days, y days and z days respectively to do the job alone. The formula to calculate the number of days if they work together is xyz/(xy + yz + xz) days where x, y, and z are given as input to the program.

```
x=int(input("x="))

y=int(input("Y="))

z=int(input("Y="))

days=x*y*z/(x*y+y*z+z*x)
```

```
print(days)
```

12. Write a program to enter two integers and perform all arithmetic operations on them.
13. Write a program to swap two numbers using a third variable.
14. Write a program to swap two numbers without using a third variable.

```
x=int(input(“Enter First Number: “))
y=int(input(“Enter Second Number: “))
x,y=y,x
print(“First Number=”,x)
print(“Second Number=”,y)
```

Another logic:

```
x = x + y
y = x - y
x = x - y
```

15. Write a program to repeat the string ‘‘GOOD MORNING” n times. Here ‘n’ is an integer entered by the user.

```
n=int(input(“Enter Number: “))
for i in range(n):
print(“GOOD MORINING “)
```

16. Write a program to find average of three numbers.

```
x=int(input(“Enter First Number: “))
y=int(input(“Enter Second Number: “))
z=int(input(“Enter Third Number: “))
avg=(x+y+z)/3
print(“Average : “,avg)
```

17. The volume of a sphere with radius r is 4/3πr3. Write a Python program to find the volume of spheres with radius 7cm, 12cm, 16cm, respectively.

```
r1 = 7
r2 = 12
r3 = 16
#calculating the volume using the formula
volume1 = (4/3*22/7*r1**3)
volume2 = (4/3*22/7*r2**3)
volume3 = (4/3*22/7*r3**3)

print("When the radius is",r1,"cm, the volume of the sphere will be", round(volume1,2),"cc")
print("When the radius is",r2,"cm, the volume of the sphere will be", round(volume2,2),"cc")
print("When the radius is",r3,"cm, the volume of the sphere will be", round(volume3,2),"cc")
```

18. Write a program that asks the user to enter their name and age. Print a message addressed to the user that tells the user the year in which they will turn 100 years old.

```
#Program to tell the user when they will turn 100 Years old
name = input("Enter your name: ")

# age converted to integer for further calculation
age = int(input("Enter your age: "))

#calculating the 100th year for the user considering 2024 as the current year
hundred = 2024 + (100 - age)

#printing the 100th year
print("Hi",name,"! You will turn 100 years old in the year", hundred)
```

19. The formula E = mc2 states that the equivalent energy (E) can be calculated as the mass (m) multiplied by the speed of light (c = about 3×10^8 m/s) squared. Write a program that accepts the mass of an object and determines its energy.

```
mass = float(input("Enter the mass of object(in grams): "))

#Speed of light is known and given in the question
c = 3 * 10 ** 8

#calculating the energy, mass is divided by 1000 to convert it into kilogram
Energy = (mass/1000) * c ** 2

#printing the output
print("The energy of an object with mass",mass," grams is",Energy,"Joule.")
```