

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.



Figure Python interpreter or 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 Modes

There are two ways to use the Python interpreter:

- a) Interactive mode
- b) Script mode

Interactive mode allows execution of individual statement instantaneously. Whereas, **Script mode** allows us to write more than one instruction in a file called Python source code file that can be executed.

(A) Interactive Mode

- To work in the interactive mode, 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(s).
- 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.

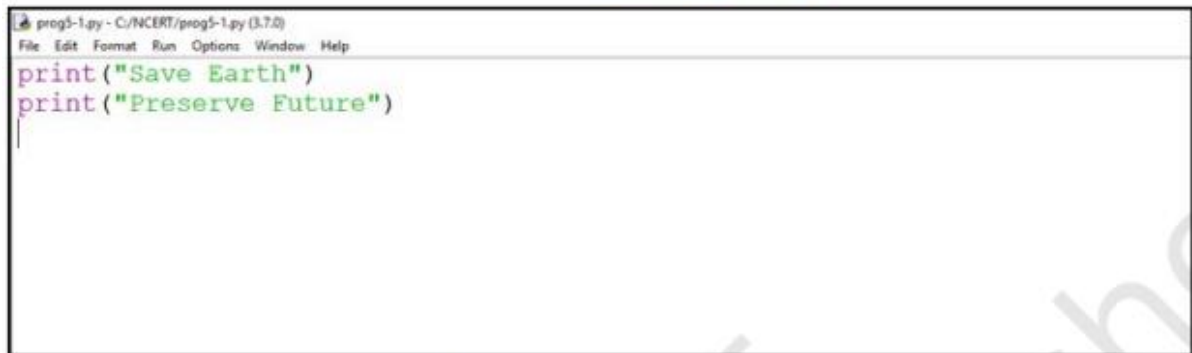
```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 1 + 2
3
>>> 4 - 2
2
>>> "Hello" + "World"
'HelloWorld'
>>>
```

Figure Python interpreter in interactive mode

(B) 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 as shown in Figure 5.3.
 - b) While working in the script mode, after saving the file, click [Run]->[Run Module] from the menu.
 - c) The output appears on shell

Program Write a program to show print statement in script mode.



```

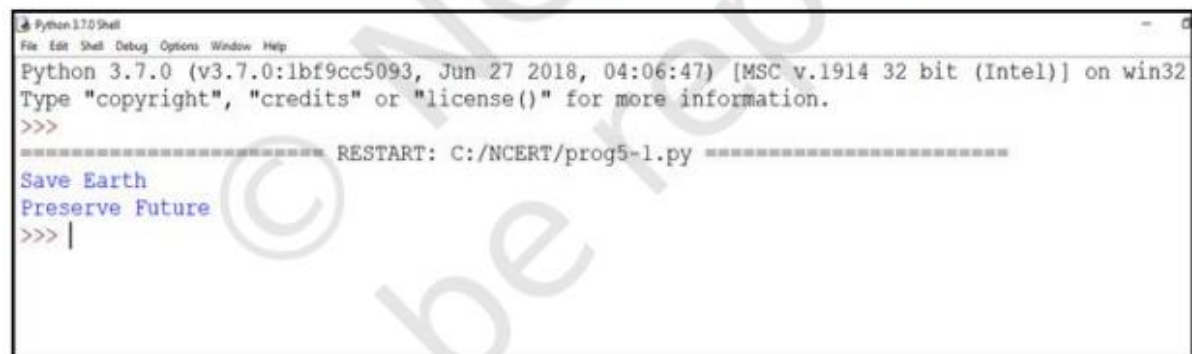
prog5-1.py - C:/NCERT/prog5-1.py (3.7.0)
File Edit Format Run Options Window Help
print("Save Earth")
print("Preserve Future")

```

Figure Python source code file (prog5-1.py)



Figure Execution of Python in Script mode using IDLE



```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/NCERT/prog5-1.py =====
Save Earth
Preserve Future
>>> |

```

Figure Output of a program executed in script mode

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.

Table Python keywords

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	
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 (_). This may be followed by any combination of characters a–z, A–Z, 0–9 or underscore (_). Thus, an identifier cannot start with a digit.
- 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.
- For example, to find the average of marks obtained by a student in three subjects, we can choose the identifiers as marks1, marks2, marks3 and avg rather than a, b, c, or A, B, C.

$$\text{avg} = (\text{marks1} + \text{marks2} + \text{marks3})/3$$

Similarly, to calculate the area of a rectangle, we can use identifier names, such as area, length, breadth instead of single alphabets as identifiers for clarity and more readability.

$$\text{area} = \text{length} * \text{breadth}$$

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.
- Variable declaration is implicit in Python, means variables are automatically declared and defined when they are assigned a value the first time.
- Variables must always be assigned values before they are used in expressions as otherwise it will lead to an error in the program.
- Wherever a variable name occurs in an expression, the interpreter replaces it with the value of that particular variable.

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

Program Write a program to display values of variables in Python.

```
#Program 5-2
#To display values of variables
message = "Keep Smiling"
print(message)
userNo = 101
print('User Number is', userNo)
```

Output:

```
Keep Smiling
User Number is 101
```

In the above program, the variable message holds string type value and so its content is assigned within double quotes " " (can also be within single quotes ' '), whereas the value of variable userNo is not enclosed in quotes as it is a numeric value.

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

```
#Program 5-3
#To find the area of a rectangle
length = 10
breadth = 20
area = length * breadth
print(area)
```

Output:

```
200
```

Comments

- Comments are used to add a remark or a note in the source code.
- Comments are not executed by interpreter.
- They are added with the purpose of making the source code easier for humans to understand.
- In Python, a comment starts with # (hash sign). Everything following the # till the end of that line is treated as a comment and the interpreter simply ignores it while executing the statement.

Example

```
#Variable amount is the total spending on
#grocery
amount = 3400
#totalMarks is sum of marks in all the tests
#of Mathematics
totalMarks = test1 + test2 + finalTest
```

Program **Write a Python program to find the sum of two numbers.**

```
#Program 5-4
#To find the sum of two numbers
num1 = 10
num2 = 20
result = num1 + num2
print(result)
```

Output:

30

Everything is an Object

- Python treats every value or data item whether numeric, string, or other type as an object in the sense that it can be assigned to some variable or can be passed to a function as an argument.
- Every object in Python is assigned a unique identity (ID) which remains the same for the lifetime of that object. This ID is akin 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

- Every value belongs to a specific data type in Python.
- Data type identifies the type of data values a variable can hold and the operations that can be performed on that data.

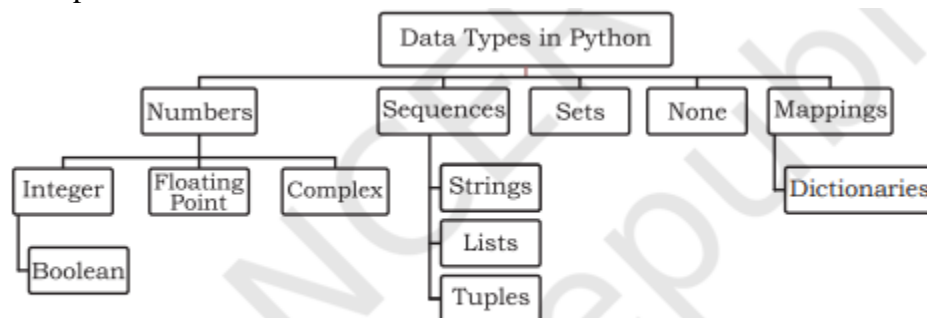


Figure Different data types in Python

Number

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

Table Numeric data types

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

In interactive mode to determine the data type of the variable using built-in function `type()`.

- Variables of simple data types like integers, float, boolean, etc., hold single values. But such variables are not useful to hold a long list of information,
- for example, names of the months in a year, names of students in a class, names and numbers in a phone book or the list of artifacts in a museum.
- For this, Python provides data types like tuples, lists, dictionaries and sets.

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'>
```

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 (e.g., 'Hello') or in double quotation marks (e.g., "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 [].

```
#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]
```

(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.

```
#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')
```

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.

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

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).

```
>>> myVar = None
>>> print(type(myVar))
<class 'NoneType'>
>>> print(myVar)
None
```

Mapping

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

(A) 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 [].

```
#create a dictionary
>>> dict1 = {'Fruit':'Apple',
             'Climate':'Cold', 'Price(kg)':120}
>>> print(dict1)
{'Fruit': 'Apple', 'Climate': 'Cold',
 'Price(kg)': 120}
>>> print(dict1['Price(kg)'])
120
```

Mutable and Immutable Data Types

- Sometimes we may require to change or update the values of certain variables used in a program.
- However, for certain data types, Python does not allow us to change the values once a variable of that type has been created and assigned values.
- Variables whose values can be changed after they are created and assigned are called **mutable**.
- 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.

Python data types can be classified into mutable and immutable as shown in below:



Figure Classification of data types

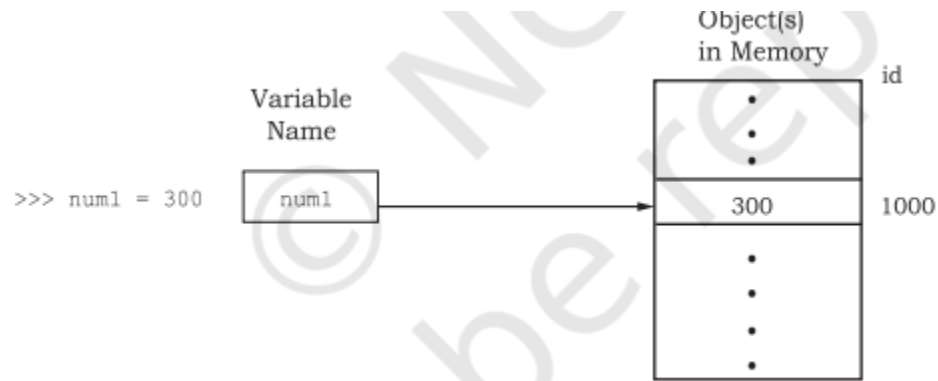


Figure Object and its identifier

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.

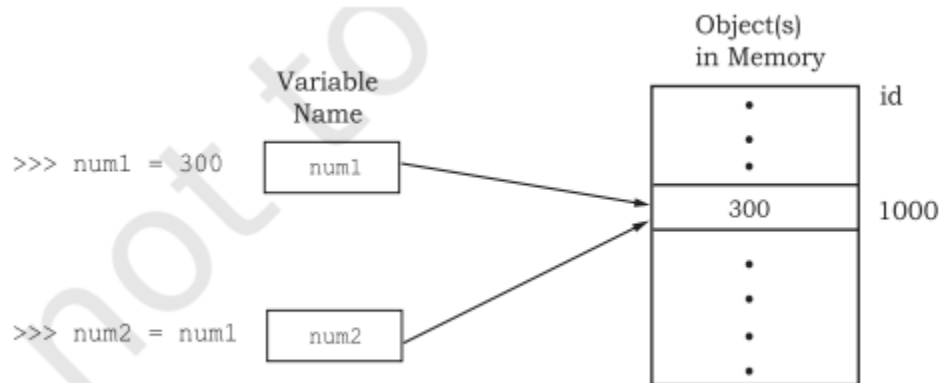


Figure Variables with same value have same identifier

The statement `num2 = num1` 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.

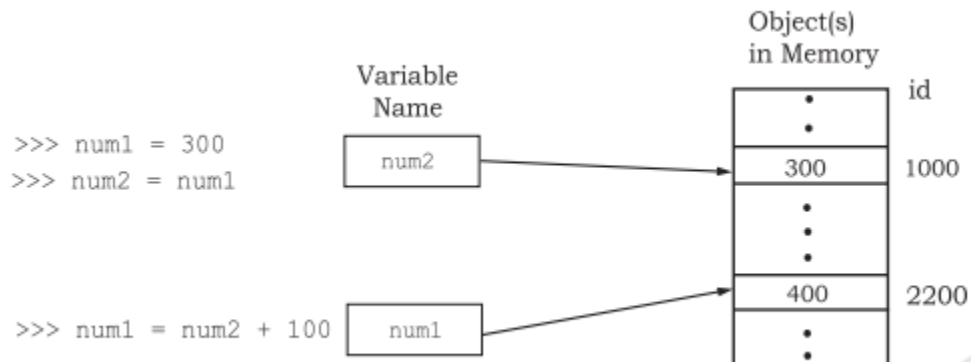


Figure Variables with different values have different identifiers

- In this manner P y t h o n makes the a s s i g n m e n t effective by copying only the reference, and not the data:
`>>> num1 = num2 + 100.`
- This statement `num1 = num2 + 100` 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

Deciding Usage of Python Data Types

- It is preferred to use lists when we need a simple iterable collection of data that may go for frequent modifications.
 For example, if we store the names of students of a class in a list, then it is easy to update the list when some new students join or some leave the course.
- Tuples are used when we do not need any change in the data.
 For example, names of months in a year.
- When we need uniqueness of elements and to avoid duplicacy it is preferable to use sets, for example, list of artifacts in a museum.
- If our data is being constantly modified or we need a fast lookup based on a custom key or we need a logical association between the key : value pair, it is advised to use dictionaries. A mobile phone book is a good application of dictionary