2
Python Basics

"A Language that doesn't affect the way you think about programming, is not worth knowning"
~Alan J. Perlis

Topics covered in this Chapter

2.1 Variables

Variables are some identification name which is given to the memory location(memory location can be found by using id(x) method), let say we want to store value 5 in variable x, here x is the name of memory location where value 5 has been stored and '=' is assignment operator which is used to cope reference from one variable to another variable. A variable is a name that refers to a value, whenever it is required we can retrieve it by just calling its name x and it will provide value using given syntax.

```
<variable name> = <value>
```

Example-

```
>>>x=5
```

```
>>>id(x)
4518782192
```

In the below example it will store value 5 at some memory location and this memory location is assigned with three variable name, it is just like one person have three different name. A value is one of the basic things a program works with like a letter or a number, = symbol is used to assign value to any variable, assigning a value to a variable itself declares and initializes the variable with that value. No keyword can be declared as variable because those are having some special meaning to the python language. You can't declare variable without assigning it a value. This assignment process goes from left to right means you can't declare variable like (5=x), which is absolutely invalid.

```
>>> x=y=z=5                          #assigning a value to a variable
>>> id(x)                            #print memory location of listed variable
1891092464
>>> id(y)
1891092464
>>> id(z)
1891092464
>>>5=x                              #Wrong variable declaration
>>>tarkesh_123_barua= "Author Name"  #valid variable
>>>name= "Mr. TB"
>>>Name= "Mr. TB"                   #name and Name variables are different
>>>x=y=z=5                          #id(x) id(y) id(z)   4518782192
>>>a, b, c, d=10, 20, 30, 40        #different values can assigned
>>>print(abc)                       #variable must be defined before use
Traceback (most recent call last):
    File "<stdin>", line 1 in <module>
NameError: name 'abc' is not defined
```

If we change its value then it will create new memory location other wise all three variable will share the same memory location. Have a look on snapshot

```
Python 3.7.1 (default, Dec 14 2018, 13:28:58)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> x=5
>>> id(x)
4518782192
>>> x=y=z=5
>>> id(x)
4518782192
>>> id(y)
4518782192
>>> id(z)
4518782192
>>> x is y
True
>>> x=y
>>> x==y
True
```

Figure 2.1 Variable declaration

Table 2.1 Keywords

There are certain rules to declare variable.

1. Variable name must start from the character and underscore
2. Variable can contain letters, numbers, and underscore
3. No period(.), hyphen(-) and special character (`,~,!,@,#,$,%,^,&,*) not allowed
4. Keywords can't be variable name
5. variable names are case sensitive(hello, Hello, HELLO, hELLO are different)
6. No need to specify data type, because automatically python can convert according to assigned value

keywords can be listed using the given line of code, exec is no longer a keyword

```
>>>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']
>>>x=10
>>>id(x)
4518782192d
```

2.2 Data types

There are two type of data in python mutable datatype and immutable datatype, mutable datatype could be make any changes during the execution of the program while immutable data types could not have any changes in stead of they always allocate new memory location.

Table 2.2 Data Types

Property	**Behavior**	**Type**	**Example**
Immutable	Numeric	Integer	x = 25
		Float	pi = 3.14
		Complex	complex_num=3.14j
		Long	long_num=343445435453L
	Sequence	String	fname = "Tarkeshwar" lname = "Barua" nameInHindi=u 'tarkeshwar'
		Byte	byteString = b'Tarkeshwar'
Mutable		Byte Array	byteArrayString = bytearray(b"Tarkeshwar")
		List	testList = [1,2,3,4,5,5,2,5,5,2,3,4]
Immutable		Tuple	testTuple = ("Francis", "Patrick", "Nixon")
Mutable	Sets	Set	fullname = set("Tarkeshwar Barua")
Immutable		Frozen Set	testFrozenSet = (["James", "Sam", "Dennis", "Toney"])
Mutable	Mapping	Dictionary	student = {"name":"Tarkeshwar Barua", "address": "123 Monrovia", "phone":123456789}

As we know python is auto type caste language but in the background it manages the various data type to store different types of values. Data type of any variable can be visualize by method type() as given in the snapshot

```
>>>message = “Good Morning!”
>>>print(“message data type is ”, type(message))
message data type is <type ‘str’>
>>>message= “Hi, There”
>>>message
Hi, There
>>>age = 20
>>>print( “age data type is ”, type(age))
age data type is <type ‘int’>
>>>height = 5.6
>>>print( “height data type is ”, type(height))
height data type is <type ‘float’>
>>>paid = True
>>>print( “paid data type is ”, type(paid))
paid data type is <type ‘bool’>
>>> x="Greeting"
>>> type(x)
<class 'str'>
>>> x=True
>>> type(x)
<class 'bool'>
>>> x=3.1343434
>>> type(x)
<class 'float'>
```

2.2.1 Booleans- Boolean data type stores value either True or False with logical operations like **and, or, not,** can be performed on Boolean. Boolean could be integer values too which stands for 1 means True and 0 means False.

```
>>> x=True              # boolean assignment to variable
>>> y=False
>>> type(x)
<class 'bool'>
>>>a and b             #if x is True and y is True then result will be TRUE
>>>a or b              #if x is False and y is True then result will be TRUE
>>>a not b             #if x is True and y is False then result will be TRUE
```

2.2.2 Integer- integer data type is used to store a fractional value of a number. In another way it stores only value without decimal point.

```
>>>x=10
>>>type(x)
<class ‘int’>
```

2.2.3 Float- Float data type is used to store a decimal value with period**.** In another way it stores only value without decimal point.

```
>>>x=10.4434343
>>>type(x)
<class ‘float’>
```

2.2.4 String- String data type is used to store a string (a sequence of characters) or a paragraph.

```
>>>name = “Tarkeshwar Barua”          #variable declaration
>>>type(name)                        # type of variable
<class ‘str’>
>>>name[0]                           #slicing from string
‘T’
>>>name[0:10]                        #slicing from String
‘Tarkeshwar’
```

2.2.5 None- None stands for void or null. This type of data type is used to clear value from the given variable or memory location

```
>>>name = None
>>>type(name)
<class ‘NoneType’>
```

2.3 Block Indentation

Like other programming language such as java, C and C++, it doesn’t have block creation using curly bracket {}. In python tab are being used to create block so that you can control and loop constructs. Here programmer have use white space carefully, wrong calibration may causes error. Python uses colon (:) and indentation for showing where blocks of code begin and end. If the line following a colon is not indented, Wrong indentation will raise **IndentationError** Example is given below

```
>>> if 3>4:
print("Hello")
SyntaxError: expected an indented block
```

`if 5 < 2:`
`  print("Hello")`
`else:`
 	
`  print("World")`
	`if 5 < 2: print("Hello")`
	`else: print("World")`

Table 2.3 Block Indentation

Here both examples are correct but second one is not consider as good practice therefore we should avoid this practice. Always use 4 spaces for indentation, python3 is not allows mixing the use of tabs and spaces for indentation.

```
>>>def myFunction():
…        a=10
…        return a
>>>print(myFunction())
```

2.4 Math Calculation

Maths operations are very important in any programming language, that is why python also have a one individual module of maths to perform complex mathematical operations. Only this needs to import from standard library to your program. Python has a bunch of built-in arithmetic operators, Math module contains implementations of common mathematical operations such as square root etc.

```
>>>3*6+3+4*(6+3)
 57
 >>>3*6+2-3*(6+4)-5
 -15
 >>>2**3+8 - 2**2+16
 28
 >>>2**2*4 /16%15 + 1 -3
 -1.0
 >>>import math
 >>>math.sin(math.radians(30))
 4.99999999999994
 >>>math.cos(math.radians(30))
 0.8660254037844387
 >>>math.tan(math.radians(30))
 0.5773502691896256
 >>>math.pi
 3.141592653589793
 >>>math.e
 2.718281828459045
 >>>math.sqrt(64)
 8.0
```

2.5 Operators

Operators are special symbols that represents computation like arithmetics, logical, assignment, increment and decrement operator. The value which is applied to operator is called operands. These calculation are goes by BODMAS which stands for bracket, of, division, multiplication, addition, subtraction. Where (**) means power 10^3

```
 >>>print(10+30*50-40/20)
 1508.0
 >>>print(10+30*(50-40)/20)
 25.0
 >>>print(10+30*(50-40)**3/20)
 1510.0
```

Parentheses have highest precedence and can be used to force and expression to evaluate in the order you want and subtraction have lowest precedence.

2.6 String Operations

String can be create by enclosing any word character or number with single and double quotes, string is a group of numbers and character and special symbols. Process of adding two or more then two string called concatenation. Sting can be created as given below-

```
 >>>firstName= “Tarkeshwar”
>>>type(firstName)
<type ‘str’>
>>>lastName= “Barua”
>>>type(lastName)
<type ‘str’>
>>>fullName=firstName+ “ ” +lastName
Tarkeshwar Barua
>>>type(fullName)
<type ‘str’>
```

----------above given program using PyCharm IDE----------

```
string="tarkeshwar barua"
print("Camel Case - "+string.capitalize()) # camel case
string1="TARKESHWAR"
print("Lower Case - "+string1.lower()) #lower case
print("Upper Case - "+string.upper()) #upper case
print("Replace A with B - "+string.replace("a","b")) # replace String
splitedword=string.split(" ") #split a string with space
for s in splitedword: #iterating splited words
        print("Splited Word is  - "+s)
print("Case fold - "+string1.casefold()) #Return a version of the string suitable for caseless comparisons
print("Swap case - "+string1.swapcase()) #Convert uppercase characters to lowercase and lowercase characters to uppercase
string2="data data data"
print("Find A - ",string2.find("d")) #Return the lowest index in S where substring sub is found,such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation
```

2.7 Indexing

String are the sequence of characters. You can access any character in a Python string using its index, either counting from the beginning or the end of the string. This indexing begins from 0(zero), as given in the example

T	**A**	**R**	**K**	**E**	**S**	**H**	**W**	**A**	**R**		**B**	**A**	**R**	**U**	**A**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
-16	*-15*	*-14*	*-13*	*-12*	*-11*	*-10*	*-9*	*-8*	*-7*	*-6*	*-5*	*-4*	*-3*	*-2*	*-1*

Table 2.3 String Indexing

```
>>>name = “TARKESHWAR BARUA”
>>>name[0]
‘A’
>>>name[11]
‘B’
>>>name[-3]
‘R’
>>>name[-13]
‘K’
```

2.8 Slicing

Slicing is a sequence of character can be sliced from the given string by specifying a range of indexing number this process known as slicing. This will be a shorter string from the long string. Have a look on the below example

```
>>>name(name)
 16
>>>name[0:10]
 ‘TARKESHWAR’
>>>name[11:16]
 ‘BARUA’
>>>name[-16:-6]
 ‘TARKESHWAR’
>>>name[-5:-1]
 ‘BARU’
```

2.9 Concatenation

it is a process to join two or more then two string together is called concatenation. This can be performed by ‘+’ plus sign between two strings. Refer example

```
>>>greeting = “Good Morning”
>>>fname= “Tarkeshwar”
>>>lname= “Barua”
>>>print(greeting+fname+lname)
‘Good MorningTarkeshwarBarua’
>>>fname*3
‘TarkeshwarTarkeshwarTarkeshwar’
```

2.10 Single Line and Multiline comments

Nobody is perfect that is why we may required to deactivate some code for troubleshooting purpose, and this commenting can make your code more human understandable. As your code size is getting bigger and it becomes more complicate to read, code understanding and bug into your code. Then comments plays very important role to explain your code functionality in natural language. These comments could be single line and multi-line. Single line comment by using # while multi-line comment need triple quote. Comments are very useful when document non-obvious features of the code. Have a look on the given below example.

```
>>>def myFunction():                                    #this is single line comment
…       a=10
…       return a
 '''this is the multiline comment example
 which can be achieved by using triple
 quote open and close and this is not part of code only for information to programmer'''

>>>print(myFunction())

 """this is the multiline comment example
 which can be achieved by using triple
 quote open and close and close and this is not part of code only for information to programmer
"""
```

2.11 Functions

Function is a named sequence of statements that is designed to perform a certain task or function is method that stored as a class attribute. Function returns a value based on its given arguments. You can create it by using **"def"** keyword and function name then you can write number of lines to execute in the function. Later this function can be called by its name to execute its written statement. There are some rules to define a function given below

1) A function can't be start with any special symbol(`, ~, !, @, #, $, %, ^, &, *, -, +, =, ", ", <, >, /, ?,).
2) A function should not have a special symbol (., @)
3) A function could not start with number
4) A function could start with _ (underscore) symbol
5) A keyword could not be a function name

A function definition specifies the name of a new function and the sequence of statements that execute when the function is called. There two types of functions

2.11.1 Built-in functions

built-in functions are already created in python and no need to create them. Built-in functions can be list using **dir(__builtins__)** function. Functionality of any function can be list by **help(max)** function. Follow the examples where id is a built-in function of python

```
 >>>x=10
 >>>id(x)
 4504909200
 >>>pow(2,3)
>>>dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError',
'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError',
'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis',
'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError',
'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError',
'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError',
'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError',
'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError',
```

'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '_', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']

2.12.2 User defined functions - User defined functions are created by programmer to perform some task. Follow the examples

```
>>>def myFunction():                    #defining a customized function
…       a=10
…       return a
>>>print(myFunction())                  #calling defined function
>>>def greeting():
…       print(“Good Morning Friends”)   #printing output inside function
>>>print(greeting())
```

2.12.3 Function with argument

Function could be created with argument too that is very useful to value inside function to achieve dependency injection. After this injection function works independently without its outside interaction and it may return some output or may not, it depends upon the system requirement. In the example we have passed two values x and y as function parameters, after passing these values these values can perform task independently as a local variable for this function and they can produce output. In the second line we have printed output and the second line method is returning value out of function. Have a look on the given example-

```
 >>>def addition(x, y):
…       print(“passed value result is ”, (x+y))      #printing output inside function
…       return x + y                                 #returning value out of function
>>>print(addition(10, 20))
```

2.11.4 Flow of execution

Function should be defined before its execution because of python is interpreted programming language so that it will interpret function definition then it can its execution could be possible. This sequence called flow of execution. Execution begins with first statement of the program. Statement inside the function are not executed until the function is called.

2.11.5 Default Values

some times happens that a function need to pass a value as a parameter unfortunately we forgot to pass those value it may cause the error in the program resulting program stops executing. To avoid this situation python by provides a default value feature in case programmer doesn't provide any value it will take default value. Passing default value also have certain rules. Second parameter could not be dynamic while first is default. First parameter should be dynamic but other could be default if function have more then one parameter. Have look on example

```
>>>def greeting(name= "Guest"):
        print("Hello Mr. ",name)
>>>greeting()
Hello Mr. Guest
>>>greeting("Tarkeshwar Barua")
Hello Mr. Tarkeshwar Barua
>>>def greeting1(fname= "Tarkeshwar", lname= "Barua"):
        print("Hello Mr. ",fname, lname)
>>>greeting1()
Hello Mr.  TarkeshwarBarua
>>>def greeting2(fname= "Tarkeshwar", lname):
        print("Hello Mr. ",fname, lname)
   File "<stdin>", line 1
SysntaxError: non-default argument follows default argument
```

2.12 Return Value

Python is a dynamically type casting language because of that there is no return type need to specify during function definition. if you want to encapsulate this logic in a function such a manner go through with given example

```
>>>def addition(a, b):
        return a+b
>>>x=addition(3,6)
>>>type(x)
<class 'int'>
>>>x
9
>>>y=addition("Tarkeshwar", "Barua")
>>>type(y)
<class 'str'>
>>>y
'TarkeshwarBarua'
```

as you have seen same function can work on string and numbers as well because its is auto type cased

2.13 Type casting

Python provides built-in functions that convert values from one type to another type. Lets assume we want to convert from **string** to **int,** follow the given example.

```
>>>int('15')
15
>>>int('Hello World')
ValueError: invalid literal for int() : Hello World
>>>int(3.1456453)
3
>>>str(3.1456453)
'3.1456453'
```

2.14 Dis-assembler

Dis-assembler module we can write in short **dis.** Byte code is an implementation of Cython interpreter. It is used for analysis of Cython byte code by dissembling. The byte code analysis API allows piece of Python code to be wrapped into byte code object. Cython byte code is an input of for this module in the include opcode.h. lets see the example

```
import dis
def greeting(name="Guest"):
        return "Hello, Mr "+name+"!"
print(greeting())
print(greeting("Tarkeshwar"))
dis.dis(greeting)
```

----------output----------

```
Hello, Mr Guest!
Hello, Mr Tarkeshwar!
  5           0 LOAD_CONST               1 ('Hello, Mr ')
              2 LOAD_FAST                0 (name)
              4 BINARY_ADD
              6 LOAD_CONST               2 ('!')
              8 BINARY_ADD
             10 RETURN_VALUE
```

2.15 Lambda Keyword

Lambda keyword provides a shortcut way for declaring small and anonymous function. Lambda functions are single-expression function that are not necessarily bound to the name means anonymous, lambdas can't use regular python statements and always include an implicit ***"return"*** statement, these functions can have any number of arguments but only one expression which is evaluated and return. It is syntactically restricted to a single expression, see the example

```
def add(x,y):              #an ordinary function
        return x+y
sub=lambda x,y:x-y         #A Lambda Function
```

```
print(add(2,2))          #lambda function calling
print(sub(4,2))          #Ordinary function calling
print((lambda x,y:x*y)(3,4)) #lambda with *function expression
```

filter function take in a function and a list of argument and can return group of item after performing certain task over it, lets see in the code

```
multiple=list(filter(lambda x:(x>10),items))  #being copied in to list, those items are greater then 10
print(multiple)
print(type(multiple))
```

and the above given code can be used with map function too as it is given in the next code

```
multiple=list(map(lambda x:(x>10),items))  #being copied in to list, those items are greater then 10
print(multiple)
print(type(multiple))
```

lambda function can return any type of function lets see in the code

```
from functools import reduce
items =[34,3,42,4,23,4,2,3,43]
multiple = reduce((lambda x, y: x * y), items)
print (sum)
```

Summary

- Python is very easy programming language to understand and develop application in sort period of time. Python comes with basics all data types, variables, etc. python is widely used among programmers.
- Python is free to use with big community support. It is very good in terms of memory management by garbage collector.
- Python basics includes Data types variables and its type conversions and various operators. Some special words those have some special meaning for python called keyword.
- Python doesn't have curly bracket { } to create Blocks. It need to manage by indentation. You can perform mathematical calculations easily with math module. Various types of operators like arithmetical, logical, comparison, assignment, etc.
- String is immutable object in all the programming language the same this applied in python too, meaning can't be modified at run time of program still you want to perform some task, its is called concatenation and slicing to get sub-string from big string.
- Comments can be done by two way, single line comment by applying **#** before the line and multi-line can be done by enclosing with three single/double quotes.
- Functions are combination of statements those executes while the been called. it make programming very easy to maintain. one function can be called many times, in python functions supports default value too.
- Conversion from one datatype to another, process known as data type casting, python supports auto type casting but explicitly also possible too.

Key terms

Python is auto type casting language, that is why not need to specify variable type for variable, according to the stored value its type changes. Keyword can't be used as variable name. Blocks are maintained by indentation. Mathematical calculation are very easily can be performed with built-in library math. Operators are used to perform operations over operands. Indexing always begins with zero and it goes up to length -1. on behalf of this string operations like slicing and concatenation are performed. Comments are used to deactivate and provide some information related to programming code.

Review Question

1. Execute the given code and find out the output

```
width = 30
 height = 20
 length = 50
 area = width*height*length
 perimeter = width+height+length
```

2. find out area and perimeter of a circle while it radius is 3.5 Mtr

```
 radius = 3.5
 pi=3.14
 area = pi*r*r
 perimeter = 2*pi*r
```

3. let's assume base price of one kilogram apple is 100 INR, but fruit seller got 10% discount while he purchased, 3% freight charge to carry in shop. What is the sale price if he want to make 10% profit on it.

```
 >>>baseprice = 100
 >>>puchase_discount = 100*(10/100)
 >>>purchase_price = baseprice - purchase_discount
 >>>freight= purchase_price*(3/100)
 >>>total_purchase_price = purchase_price + freight
 >>>sale_price = total_purchase_price + total_purchase_price * (10/100)
```

4. Explain different data types in python?
5. What type of functions are available in python? Explain.

Exercise

Tick the correct option

Q.1 Is Python case sensitive when dealing with identifier?

a) Yes

b) No

c) Machine Dependent
d) None of the mentioned

Answer: b) Yes

Q.2. What is the maximum possible length of an identifier?

a) 31 Characters
b) 63 Characters
c) 79 Characters
d) None of the mentioned

Answer: d) None of the mentioned

Q.3. Which of the following is an invalid variable?

a) my_string_1
b) 1st_string
c) foo
d) __

Answer: b) 1st_string

Q.4. Which of the following is not a keyword?

a) eval
b) assert
c) nonlocal
d) pass

Answer: a) eval

Q.5. All keywords in Python are in

a) lower case
b) UPPER CASE
c) Capitalized
d) None of the mentioned

Answer: d) None of the mentioned

Q.6. Which of the following is true for variable names in Python?

a) Unlimited length
b) All private members must have leading and trailing under scores

c) Underscore and ampersand are the only two special characters allowed

d) None of the mentioned

Answer: a) Unlimited length

Q.7. Which of the following is an invalid statement?

a) abc=1,000,000

b) a b c =1000 2000 3000

c) a, b, c = 1000, 2000, 3000

d) a_b_c = 1,000,000

Answer: b) a b c = 1000 2000 3000

Q.8. Which of the following can not be a variable?

a) __init__

b) in

c) it

d) on

Answer: b) in

Q.9. print("Hello {0!r} and {0!s}".format('foo', 'bin'))?

a) Hello foo and foo

b) Hello ‘foo’ and foo

c) Hello foo and ‘bin’

d) Error

Answer: b) Hello ‘foo’ and foo

Q.10. Which of the following data types is not supported in python?

a) Numbers

b) String

c) List

d) Slice

Answer: d) Slice

Fill in the Blanks

1. are the some special characters that are having some special meaning for programming language.
2. _____datatype could be make any changes during the execution of the program.
3. _____is the process to join two or more then two string together
4. _____is a sequence of statements that is designed to perform a certain task.
5. Sequence of character from the given String known as _____

Answers

1. Keywords
2. Mutable
3. Concatenation
4. Function
5. Slicing