Assignment 1

1. In the below elements which of them are values or an expression? eg.:- values can be integer or string and expressions will be mathematical operators.

(* 'hello' -87.8 - / + 6)

Answer:

*: Multiplication operator. Multiplies value on either side of the operator.

a*b if a=5 and b=10 then it's result a*b = 50.

'hello': String Literals, A group of characters is called a string literal. These string literals are enclosed in single quotes (′) or double quotes (″) or triple quotes (‴‴ or """). In Python, there is no difference between a single-quoted string and double-quoted string.

-87.8: Float Datatype, the float datatype represents a floating-point number. A floating-point number is a number that contains a decimal point. That is num = -87.8 here num is a variable that stores a float number.

-: Subtraction operator. Subtracts one value from another.

a-b if a=70 and b=20 then its result a-b = 50.

/: Division operator. Divides left operand by the right operand.

a/b if a=13 and b=5 than its result a/b = 2.6.

+: Addition operator. Adds two values.

a+b if a=30 and b=20 than its result a+b = 50.

6: int datatype represents an integer number. An integer number is a number without a decimal point or fraction part. For ex. 6, 200, -50, 1254897, 0, etc. Here a=6, 'a' is called int type variable since it is storing 6 which is an integer value. In Python, there is no limit to the size of an int datatype. It can store very large integer numbers conveniently.

2. What is the difference between string and variable?

Answer:

String: A string represents a group of characters. See the example: "Hello". When a string is passed to the print() function, the string is displayed as it is: Print("Hello") the get output: Hello.

Remember that in the case of strings, double quotes and single quotes have the same meaning, and hence can be used interchangeably.

Variable: In all languages, a variable is imagined as a storage box that can store some value. Suppose, we write a statement as; a=5 some memory is allocated with the name 'a' and there the value '5' is stored.

3. Describe three different data types?

Answer:

Datatypes in Python: A datatype represents the type of data stored in a variable or memory. The datatypes which are already available in Python language are called Built-in datatypes. The datatype which can be created by the programmers are called User-defined datatypes.

Built-in Datatypes:

The built-in datatypes are of five types:

I. NoneType
II. Numeric types
III. Sequences
IV. Sets
V. Mappings.

1.NoneType:

In Python, the 'NoneType' datatype represents an object that does not contain any value. To represent such an object, we use 'null' in Java, but in Python, we use 'None' to represent an empty object and its datatype is considered as 'Nonetype'. In a Python program, a maximum of only one 'None' object is provided. One of the uses of 'None' is that it is used inside a function as a default value of the arguments. When called the function, if no value is passed, then the default value will be taken as 'None'. If some value is passed to the function, then that value is used by the function.

2.Numeric Types:

i. int
ii. float
iii. complex

1.int datatype:

The int datatype represents an integer number. An integer number is a number without any decimal point or fraction part. For example, 500, -45, 0, 8458294457, etc. Are treated as integer numbers. Ex. a=-45 here 'a' is called int type variable since it is storing -45 which is an integer value.

2. float Datatype:

The float datatype represents floating point numbers. A floating-point number is a number that contains a decimal point. For example, 0.5, -3.4485, 390.48, 0.0001, etc. are called floating point numbers. Example as: num = 45.9554 here num is called float type variable and it is storing floating point value. The floating-point number also written in the notation ‘e’ or ‘E’ represents the power of 10. Here ‘e’ or ‘E’ represents ‘exponentiation’. Ex. 2.5 *10^4 is written as 2.5E4. or x = 2.5e4.

3. Complex Datatype:

A complex number is a number that is written in the form of a+bj or a+bJ. Here, ‘a’ represent the real part of the number, and ‘b’ represents the imaginary part of the number. The suffix ‘j’ or ‘J’ after ‘b’ indicates the square root value of -1. The parts ‘a’ and ‘b’ may contain integers or floats. Ex. 3+5j, -1-5.5J, and 0.2+10.5J are complex numbers. Statement: c1 = -1-5.5J.

*bool datatype:

The bool datatype in Python represents boolean values. There are only two Boolean values True or False that can be represented by this datatype. Python internally represented True as 1 and False as 0. A blank string like "" is also represented as False. The condition will be evaluated internally to be either True or False. Ex. a = 10 and b = 20, if (a < b): print("Hello") # display Hello.

3. Sequences in Python:

A Sequence represents a group of elements or items. For example, a group of integer numbers will form a sequence. There are six types of sequences in Python:

i. str
ii. bytes
iii. bytearray
iv. list
v. tuple
vi. range

1. str Datatype:

In Python, str represents string datatype. A string is represented by a group of characters. Strings are enclosed in single quotes or double quotes. Both are valid: str = 'Welcome' or str = "welcome".

2. bytes Datatype:

The bytes datatype represents a group of byte numbers just like an array does. A byte number is any positive integer from 0 to 255 (inclusive). Bytes array can store numbers in the range from 0 to 255 and it cannot even store negative numbers.

Ex. element = [10, 20, 0, 40, 15]

x = bytes(element)

print(x[0]) # display 0^{th} element, i.e. 10.

We cannot modify or edit any element in the bytes-type array. For ex. x[0]=45 gives an error.

3.bytearray Datatype:

The bytearray datatype is similar to the bytes datatype. The difference is that the bytes type array cannot be modified but the bytearray type array can be modified. It means the array element or all the element of the bytearray type can be modified. To create a bytearray type array, we use the function bytearray as:

Element = [10, 20, 0, 40, 15] # This is list of byte numbers

x = bytearray(Element) # convert the list into bytearray type array

print(x[0]) # display 0^{th} element, i.e. 10

we can modify or edit the element of bytearray.

x[0] = 45 # replace 0^{th} element by 45

x[1] = 59 # replace 1^{st} element by 59.

4. list Datatype:

A list represents a group of elements. The main difference between a list and an array is that a list can store different types of elements but an array can store only one type of elements. Also, lists can grow dynamically in memory. But the size of arrays is fixed and they cannot grow at runtime. Lists are represented using square brackets [ ] and the elements are written in [ ], separated by commas. Ex. list = [10, -20, 15.5, 'Shivsagar', "Priyanka"] will create a list with a different types of elements. The slicing operation like [0:3] represents elements from 0^{th} to 2^{nd} positions, i.e., 10, -20, 15.5.

5. tuple Datatype:

A tuple is similar to a list. A tuple contains a group of elements which can be different types. The elements in the tuple are separated by commas and enclosed in parentheses ( ). Whereas the list

elements can be modified, it is not possible to modify the tuple elements. That means a tuple can be treated as a read-only list. Let's create a tuple as:

tpl = (10, -20, 15.5, 'Shivsagar', "Priyanka")

The individual elements of the tuple can be referenced using square braces as tpl[0], tpl[1], tpl[2]

6. range Datatype:

The range datatype represents a sequence of numbers. The numbers in the range are not modifiable. Generally, the range is used for repeating a for loop for a specific number of times. To create a range of numbers, we can simply write: r = range(10) here, the range object is created with the numbers starting from 0 to 9. We can display these numbers using a for loop as: for i in r: print(i). The above statement will display numbers from 0 to 9. We can use a starting number, an ending number and a step value in the range object as: r = range(30, 40, 2). Starting number 30 and ending number 39, the step size is 2. It means the numbers in the range will increase by 2 every time. for loop as: for I in r: print(i) # display 30,32,34,36,38.

4. Sets:

A set is an unordered collection of elements much like a set in mathematics. The order of elements is not maintained in the sets. It means the elements may not appear in the same order as they are into the set. Moreover, a set does not accept duplicate elements. There are two sub-types in sets:

1. set datatype
2. frozenset datatype.

1. set Datatype:

To create a set, we should enter the elements separated by commas inside curly braces { }.

Lst = [1,2,5,3,4,8,9,7,6]

s = set(Lst)

print(s) # may display {1,2,3,4,5,6,7,8,9} if, since sets are unordered, we cannot retrieve the elements using indexing or slicing operations. But using the update() method we used to add element to set as:

s.update([10,11,12])

print(s) # may display {1,2,3,4,5,6,7,8,9,10,11,12} on other hand using remove() method we are remove particular from a set as:

s.remove(12)

print(s) # may display {1,2,3,4,5,6,7,8,9,10,11}

2.frozenset Datatype:

The frozenset datatype is the same as the set datatype. The main difference is that the elements in the set datatype can be modified; the elements of frozenset cannot be modified. For creating a frozenset by passing a set to frozenset() function as:

s = {50,60,70,80,90}

print(s) # may display {80,90,50,60,70}

fs = frozenset(s) 3 create frozenset fs

print(fs) # may display frozenset({80,90,50,60,70}) another way of creating a frozenset is by passing a string (a group of characters) to the frozenset() function as:

fs = frozrnset("abcdefg")

print(fs) # may display frozenset ({'e', 'g', 'f', 'd', 'a', 'c', 'b'}) However, update() and remove() methods will not work on frozensets since they cannot be modified or updated.

5. Mapping Types:

A map represents a group of elements in the form of key-value pairs so that when the key is given, we can retrieve the value associated with it. The dict datatype is an example for a map. The 'dict' represents a 'dictionary' that contains pairs of elements such as the first element reparents the key and the next one becomes its value. The key and its value should be separated by a comma. Elements enclosed curly brackets{ }.

d = {10: 'Shivsagar', 11: 'Priyanka', 12: 'Dipeeka', 13: 'Sayali', 14: 'Namrata', 15: 'Ram'}

Here, roll numbers are key and names will become values. We can create an empty dictionary without any elements as:

d = { }

we can store the key and value into d as:

d[8] = 'Kamal'

d[9] = 'Pranav'

print(d) # may display {8: 'Kamal', 9: 'Pranav'} to retrieve value upon giving the key, by mentioning d[key].

Note: **User-defined datatypes:** the datatypes which are created by the programmers are called 'user-defined' datatypes. For ex., an array, a class, or a module is user-defined datatypes.

4. What is an expression made up of? What do all expressions do?

Answer:

The expression in Python can be considered as a logical line of code that is evaluated to obtain some result. A combination of operands and operators is called an expression. The expression in Python produces some value or result after being interpreted by the Python interpreter.

An example of an expression can be: x = x + 20. In this expression, the first 10 is added to the variable x. After the addition is performed, the result is assigned to the variable x.

```
x = 25  # a statement

x = x + 20  # an expression

print(x)  # may display 45.
```

An expression in Python is very different from a statement in Python. A statement is not evaluated for some results. A statement is used for creating variables or for displaying value. Example : a = 45 # a statement

```
print(a)  # a statement

# output = 45
```

An expression in Python can contain identifiers, operators, and operands. An identifier is a name that is used to define and identify a class, variable, or function in Python. An operand is an object that is operated on. On the other hand, an operator is a special symbol that performs the arithmetic or logical computation on the operands.

 # What do all expressions do?

In Python, there are various types of expressions that perform or produce some result or value. Let us understand:

1. Constant expression: An expression in Python that contains only a constant value is known as a constant expression. Ex: x = 10 + 15 # Here both 10 and 15 are constants but x is a variable.

2. Arithmetic Expressions: An expression in Python that contain a combination of operators, operands, and sometimes parenthesis is known as an Arithmetic expression. The result of an arithmetic expression is also a numeric value just like the constant expression. Ex. x = 45 and y = 05

Addition x+y # may display 50; Subtraction x-y #may display 40; Product = x*y # may display 225; Division x/y # may display 9.

3. Integral Expression: An integral expression in Python is used for computations and type conversion (integer to float, a string to integer, etc.) An integral expression always produces an integer value as a resultant.

Ex. x = 10 # an integer number and y = 5.0 # a floating-point number
# We need to convert the floating-point number into an integer
result = x + int(y) # may result 15

4. Floating Expression: A floating expression in Python is used for computations and type conversion (integer to float, a string to integer, etc.) A floating expression always produces a floating-point number as a result.

Ex. x = 10 # an integer number and y = 5.0 # a floating-point number
# we need to convert the integer number into a floating-point number
result = float(x) + y # may display 15.0

5. Relational Expression:

A relational expression in Python can be considered as a combination of two or more arithmetic expressions joined using relational operators. The overall expression results in either True or False (Boolean result). We have four types of relational operator in Python (i.e., >, <, >=, <=). A relational operator produces a Boolean result so they are also known as Boolean Expression. For ex. 10 + 15>20 # may display the result True.

6. Logical Expression: As the name suggests, a logical expression performs the logical computation, and the overall expression results in either True or False. We have three types of logical expression in Python that is:

Operator	Syntax	Working
and	x and y	The expression returns True if both x and y are true, else it returns False.
or	x or y	The expression returns True if at least one of x or y is True.
not	not x	The expression returns True if the condition of x is False.

7. Bitwise Expressions: The expression in which the operation or computation is performed at the bit level is know as bitwise expression in python. The bitwise expression contains the bitwise operators. Ex. x = 25 left_shift = x << 1 right_shift = x >> 1 print("One right shift of x results: ", right_shift) print("One left shift of x results: ", left_shift)

Output: One right shift of x results: 12 One left shift of x results: 50

8. Combinational Expressions: As the name suggests, a combination expression can contain a single or multiple expressions which result in an integer or Boolean value depending upon the expressions involved. Ex. x = 25 y = 35 result = x + (y <<1) print("Result obtained :",result) output: Result obtained: 95.

Conclusion: All expression is reduced down to a single value.

5. This assignment statements, like spam = 10. What is the difference between an expression and a statement?

Answer:

In given question spam = 10 is a statement and spam are variable name assign value 10. It done not give any operational result or value. The main difference between expression and statement is :

Statement in Python	Expression in Python
A statement in Python is used for creating variables or for displaying values.	The expression in Python produces some value or result after being interpreted by the Python interpreter.
A statement in Python is not evaluated for some results.	An expression in Python is evaluated for some results.
The execution of a statement changes the state of the variable.	The expression evaluation does not result in any state change.
A statement can be an expression.	An expression is not a statement.
Example: x=3. **Output :** 3	**Example**: x=3+6. **Output :** 9

6. After running the following code, what does the variable bacon contain?

bacon = 22

bacon + 1

Answer: variable bacon contain value 22, because it is a statement which is assigned value 22 to variable bacon. The bacon + 1 expression does not reassign the value in bacon.

7. What should the values of the following two terms be?

'spam' + 'spamspam'

'spam' * 3

Answer: 'spamspamspam′ 'spamspamspam′

8. Why is eggs a valid variable name while 100 is invalid?

Answer: As we know variable name done not start with any number or integer, and here 100 is integer where as eggs can be used as variable name because it contains group of characters.

9. What three functions can be used to get the integer, floating-point number, or string version of a value?

Answer: integer: int( ) The int datatype represents an integer number. An integer number is a number without any decimal point or fraction part. For example, 500, -45, 0, 8458294457, etc. Are treated as integer numbers. Ex. a=-45 here 'a' is called int type variable since it is storing -45 which is an integer value.

Floating-point ( ) : float( ) The float datatype represents floating point numbers. A floating-point number is a number that contains a decimal point. For example, 0.5, -3.4485, 390.48, 0.0001, etc. are called floating point numbers. Example as: num = 45.9554 here num is called float type variable and it is storing floating point value. The floating-point number also written in the notation 'e' or 'E' represents the power of 10. Here 'e' or 'E' represents 'exponentiation'. Ex. 2.5 $*10^4$ is written as 2.5E4. or x = 2.5e4.

String: str( ) str represents string datatype. A string is represented by a group of characters. Strings are enclosed in single quotes or double quotes. Both are valid: str = 'Welcome' or str = "welcome".

10. Why does this expression cause an error? How can you fix it?

'I have eaten ' + 99 + ' burritos.'

Answer: In above expression, + 99 + which is integer value and remaining portion is string so the operations are incompatible until "99" is converted into a string. so to fix this problem we used str( ) function and convert '+99+'integer into string.

****