Q1: What are the differences between operators and values in the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Description** | **Definition** | **Examples** (if required) |
|  | 'hello’ | A string is a sequence/combination of characters. ‘hello’ is a string of 5 characters, implying it will be stored in 5 indexes in Python.  Strings help us save a combination of English alphabets in python for various purposes like input from the user, data from the coder, etc.  To declare a string, we use ‘ ’ , “ ” or “““ ”””. We use triple quotes for declaring a multi-line string. | 1. ‘Hello World.’ 2. “Udyan’s Assignemnt” 3. “““   This is a multi-line string. Data Science has various streams sub-dividing it into a plethora of branches.  ”””   1. c = ‘Saving a string in a variable.’ |
|  | \* | An asterisk (\*) is the multiplication operator in python that helps us multiply two (or more) integer or float type operands.  We can also use the \* operator to output a string type operand. It won’t multiply the said string but it will repeat the said string as many number of times as we instruct it to. | 1. a = 8\*9 2. b = 82.14\*5.55\*36.4558 3. c = 5\*str |
|  | -87.8 | Is a negative float type value. We can save it in a variable for further operations like multiplication using the \* operator with another int/float value. | 1. a = -87.8 2. b = -87.8\*4 3. c = -87.8/2 |
|  | - | This operator will Subtract right operand from the left or unary minus (where the (unary minus) operator negates the value of the operand. The operand can have any arithmetic type). | 1. a = x - y- 2 2. a = 9 - 5 3. a = 5 – 9.55 |
|  | / | The / operator divides left operand by the right operand and will always results into a float type value. | 1. a = 8/9 2. a = x/y |
|  | + | The + operator adds two operands or a unary plus. | 1. a = 58+6 2. c = x+y+z |
|  | 6 | This is an integer type value (without any decimal places) that usually is either manipulated with another int integer value or a float type (values with decimals) value.  We can either ask the user for an input or we can declare it in a variable in the code itself. | 1. a = 6\*6 2. a = int(input(“Enter a value: ”) |

Q2: What is the difference between string and variable?

A2:

Strings: A string is a sequence/combination of characters. One character will simply be a symbol. The English language has 26 characters. Computers do not deal with the said characters. They deal with binary numbers – 0 and 1. Even though you may see characters on your screen, internally it is being stored and manipulated as a combination of 0s and 1s. Strings in python are surrounded by either single quotation marks, or double quotation marks. A triple quotation marks can be used in case of multi-line string.

‘hello world’ is the same as “hello world” or “““hello world”””.

You can display string literals with the print() function as follows:

print(‘hello’)

print(“It’s a world full of possibilities for Artificial Intelligence!”)

print(“““

The most fascinating fact I’ve heard about universe is that we have existed for only a few seconds compared to the big bang, the evolution and several other world exctintion threats. But be-ware: there is more to come!

”””)

Variables: Variables are created when you create a variable you reserve some space in memory. Python has no specific command for declaring a variable. A variable is created the moment you first assign a value to it. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Thus, by assigning different data types to variables, you can store integers, decimals or characters (or a combination of the three) in these variables.

Example:

x = 500  
y = "John bought a great silk robe for his graduation"  
print(x)  
print(y)

Q3: Describe three different data forms:

A3:

They are:

1. *Float:* they are used when we need to work with and manipulate numbers with decimal values. They are used to store data like one’s bank balance, student’s fees, etc.

Ex:

x = 24.99  
print("John’s robe after discount is exactly",x,"$")

1. *Int:* such data types are used when we have to work with a whole number. They are used to store numbers without any decimals places like a student’s roll number, his admission number, etc.

Ex:

y = 300136428  
print("John’s student number is:",y)

1. *Boolean*: Booleans represent either of the two values: True or False. In (any) programming, you often need to know if an expression is True or False. You can evaluate any expression in Python, and get one of two answers, True or False. When you compare two values, the expression is evaluated and Python returns the Boolean answer.

Ex:

print(10 > 9)  
print(10 == 9)  
print(10 < 9)

We can also use boolean logic in loops like if-else, nested if-elif-else, etc.

Ex:

a = 200

b = 33

if b > a:

print("b is greater than a")

elif a==b:

print("a is equal to b")

else:

print("b is not greater than a")

Q4: What makes up an expression? What are the functions of all expressions?

A4:

[Expressions](http://docs.python.org/reference/expressions.html) contain [identifiers](http://docs.python.org/release/2.5.2/ref/identifiers.html), [literals](http://docs.python.org/release/2.5.2/ref/literals.html) and [operators](http://docs.python.org/release/2.5.2/ref/operators.html) where operators include arithmetic and/or Boolean operators, the function [call operator](https://docs.python.org/3/reference/expressions.html?highlight=subscriptions#calls) () the [subscription operator](https://docs.python.org/3/reference/expressions.html?highlight=subscriptions#grammar-token-subscription) [] and similarly, can be reduced to some kind of value, which can be an object in Python. Metacharacters, Special Sequencing, Sets and other such features can be used to form an expression.

Ex:

a = 200

b = 33

print(a+b)

Q5: In this chapter, assignment statements such as spam = 10 were added. What's the difference between a declaration and an expression?

A5:

Declaration: Declaring a variable means permanently assigning it to a data type. Declaration of variables is not required in Python. If there is need of a variable, you may think of a name and start using it as a variable. Another remarkable aspect of Python is that not only the value of a variable can change during program execution but the type of the said variable can change as well. You can assign an integer value to a variable, use it as an integer for a while and then assign a string to the variable.

Ex:

i = 30

i += 1

print(i)

Expression: You can create simple or complex expressions by combining (one or multiple) objects and operators. In Python, operators are special symbols which designate that some sort of computation should be performed. The values that an operator acts on are called operands. A sequence of operands and operators, like a + b - 5, is called an expression. Python supports many operators for combining data objects into expressions using arithmetic operators, comparison operators, identity operators, logical operators, etc.

Ex:

a = 200

b = 33

print(a+b)

Q6. After running the following code, what does the variable bacon contain?

bacon = 22

bacon + 1

A6:

The variable bacon now contains 23 as we assigned 1 using the arithmetic operator to bacon, which had 22 assigned to it already.

Q7. What should the values of the following two terms be?

'spam' + 'spamspam'

'spam' \* 3

A7:

Both the expressions will output: 'spamspamspam'

However:

1. 'spam' + 'spamspam' is string concatenation.
2. 'spam' \* 3 is the multiplication of a string using the mathematical operator.

Q8. Why is it that eggs is a true variable name but 100 is not?

A8:

eggs is a true variable whilst 100 is not name because the former follows the rules of variable naming nomenclature which clearly states that:

1. A variable name cannot start with a number.
2. A variable name must start with a letter or the underscore character. (A-z or \_)
3. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_)
4. Variable names are case-sensitive. (myvariable, Myvariable, MyVariable and MYVARIABLE are four different variables.)

Q9: Which of the following three functions may be used to convert a value to an integer, a floating-point number, or a string?

A9:

The three functions are used to cast (*explicit conversion type\**) are used to cast one data type into another. They are:

1. int() type function is used to specify integer literal. This function can be used to cast into an integer number from a string literal or Float.
2. str() type function is used to specify the string literal. This function can be used to cast into a string from an integer literal or float literal.
3. float() type function is used to specify the float data types. This function can be used to cast into a float literal from integer or string literal.

Two other typecasting functions are boolean() and complex().

You can’t convert complex datatype to int.

*explicit conversion type\**: When Python cannot typecast the data on its own and users convert the data type of an object to required data type. This type of conversion is also called typecasting because the user casts (changes) the data type of the objects. Sometimes, it is also called downcasting

1. *Note 1:* In some cases, Python can automatically convert one data type to another data type. That is *implicit conversion type* This process doesn't need any user involvement.

*explicit conversion type using functions can be done as follows:*

# 1 --> Converting to integer type using int()  
print("1: converting to integer type:")  
print(int(9.6)) # float to int  
print(int(True)) # bool to int  
print(int(False)) # bool to int  
print("123") # str to int  
print('\n')  
  
# 2 --> converting to float type using float()  
print("2: converting to floating type:")  
print(float(12)) # int to float  
print(float(True)) # bool to float  
print(float(False)) # bool to float  
print(float("123")) # string to float  
print('\n')  
  
# 3 --> converting to bool type using bool()  
print("3: converting to boolean type:")  
print(bool(12)) # integer to bool  
print(bool(0.0)) # float to bool  
print(bool([1,2,3])) # list to bool  
print(bool()) # tuple to bool  
print('\n')  
  
# 4 --> converting to string type using str()  
print("4: converting to string type:")  
print(str(123)) # integer to string  
print(str(12.12)) # float to string  
print(str([10, 20, 30])) # list to stringS

1. *Note 2:* # (hashtags) are used to denote the start of a comment in Python.

Q10. What is the error caused by this expression? What would you do about it?

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

A10:

The error caused by the expression above is:

TypeError: can only concatenate str (not "int") to str which means that the int type value 99 cannot be added to the string unless we are typecasting it to a string using the str() function.

We can correct the error by performing the following correction:

'I have eaten ' + str(99) + ' burritos.'

The output will be:

Out[1]: 'I have eaten 99 burritos.'