Assignment No. 1

Submitted By: Sachin Dodake

Q-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.

```
i) *
ii) 'hello'
iii) -87.8
iv) -
v) /
vi) +
vii) 6
```

Ans-1

The explaination is as below:

```
i)  *           => Expression
ii)  'hello'    => Value
iii) -87.8      => Value
iv)  -          => Expression
v)  /           => Expression
vi)  +          => Expression
vii)  6         => Value
```

Q-2. What is the difference between `string` and `variable` ?

Ans-2

`Difference between String & Variable`

`String:`

- String is a collection of alphabets, words or other characters.
- It is one of the primitive data structures and are the building blocks for data manipulation.
- Python has a built-in string class named str.
- Python strings are "immutable" which means they cannot be changed after they are created.
- For string manipulation, we create new strings as we go to represent computed values because of their immutable property.
- **Example:**

```
single_quote_string = 'Welcome to the !INeuron'.'
double_quote_string = 'I am  'Sachin', and I love to write codes.'
triple_quote_string = '''We are going to learn "ML" and "DL '''
```

`Variable:`

- A variable is a label or a name given to a certain location in memory. This location holds the value you want your program to remember, for use later on.
- To create a new variable in Python, you simply use the assignment operator (=, a single equals sign) and assign the desired value to it.
- Assigning an initial value to a variable is called initializing the variable.
- **Example:**

```
variable_int = 7777
variable_string = "8888"
variable_float = 99.01
variable_bool= True
```

Q-3. Describe three different data types.

Ans-3

- Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data.
- In Python, everything is an object; therefore data types are actually classes and variables are instance (object) of these classes.
- Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.
- The various data types in Python are :

```
1. int,                2. float,              3. complex,            4. bool,
5. str,                6. list,               7. tuple,              8. set,
9. Frozenset,      10. dict,              11. byte,              12.
bytearray,
13. range,         14. None
```

`1. Integer Data Type:`

- In Python, integers are zero, positive or negative whole numbers without a fractional part and having unlimited precision,
- Integer data type is represented by int class.
- It contains positive or negative whole numbers (without fraction or decimal).
- In Python there is no limit to how long an integer value can be.
- Integers can be binary, octal, and hexadecimal values.
- **Examples:**

```
  i. first_i = 10 (Normal positive integer value)
 ii. second_i = 15654223534887711  (Long positive integer value)
iii. third_i = -5255522   (short Negative integer value)
```

`2. Float Data Type:`

- In Python, floating point numbers (float) are positive and negative real numbers with a fractional part denoted by the decimal symbol . or the scientific notation E or e WITH BASE 10.
- Floats has the maximum size depends on system. The float beyond its maximum size referred as "inf", "nan", "INFINITY".
- Float 2e400 will be considered as infinity for most systems.
- Scientific notation is used as a short representation to express floats having many digits. For example: 345.567(8) is represented as 3.45567(8)e2 or 3.45567(8)E2
- **Examples:**

```
  i. float_1 = 2.45 (valid)
 ii. float_1=23_42_222_813 (valid)
iii. float_1=2e400  (valid, but its actual value is "inf", "Inf", "INFINITY", or
"infinity")
iv. float_4=3e3 (valid, but its actual value is 3*10to3 )
 v. float_5= 2.5e2  (valid, but its actual value is 2.5*10pow2)
```

`3. String Data Type:`

- The most commonly used data type is 'str' data type.
- The string can be defined as the sequence of characters represented in the quotation marks.
- In Python, we can use single, double, or triple quotes to define a string.
- In Python, strings are arrays of bytes representing Unicode characters.
- In python, there is no character (i.e. char) data type, a character is a string of length one. It is represented by str class.
- The operator (+) is used to concantenate two strings and the asterisk (*) operator is the used for repetition of string.
- The string is a sequence of Unicode characters
- **Examples:**

```
  i.  string_1 = 'sachin'
 ii.  string_2 = "virat"
iii.  string_3 = '''rohit'''
```

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

Ans-4

- An expression is a combination of operators and operands that is interpreted to produce some other value.
- An expression is a made up of variables, operators, and method invocations, which are constructed according to the syntax, that evaluates to a single value.
- The expression in Python produces some value or result after being interpreted by the Python interpreter.
- **Examples:**

```
In [3]:
```

```
        x = 25                  # a statement
        x = x + 10                # an expression
```

```
In [3]:
```

```
# assigning values to variable x and y
x = 10  # variable x
y = 5   # variable y

addition = x + y          # expression for sum
subtraction = x - y       # expression for difference
product = x * y           # expression for  product
division = x / y          # expression for division
power = x**y              # expression for power

print("The sum of x and y is: ", addition);
print("The difference between x and y is: ", subtraction)
print("The product of x and y is: ", product)
print("The division of x and y is: ", division)
print("x to the power y is: ", power)
```

```
The sum of x and y is:  15
The difference between x and y is:  5
The product of x and y is:  50
The division of x and y is:  2.0
x to the power y is:  100000
```

`What do all expressions do?`

- The expression in Python produces some value or result after being interpreted by the Python interpreter. An expression in Python is a combination of operators and operands.
- **Examples:**

```
        x = 25                  # a statement
        x = x + 10                # an expression
```

- 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'.

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

Ans-5

- The general thumb rule to define difference between statement and expression is : If you can print it, or assign it to a variable, it's an expression. If you can't, it's a statement.

`Statement:`

- A statement is an instruction that a Python interpreter can execute. So, in simple words, we can say anything written in Python is a statement. Python statement ends with the token NEWLINE character. It means each line in a Python script is a statement.
- Statements are used to perform actions or change the state of the program. They are used to execute code and do not return a value. Examples of statements include assignment statements, function calls, and control statements.
- **Examples:** In the following statement, the variable x is assigned the value 5

```
        x = 5
```

- The various types of statements in Python are listed below:

```
        i) Print statements,
        ii) Assignment statements,
        iii) Compound statements,
        iv) Expression statements:
        v) Multi-line Statements,
        vi) The 'pass' statement,
        vi) The 'del' statement,
        vii) The 'return' statement,
        viii) The 'import' statement,
        ix) The continue and break statement.
```

`Expression:`

- An expression is a combination of operators and operands that is interpreted to produce some other value. An expression is a made up of variables, operators, and method invocations, which are constructed according to the syntax, that evaluates to a single value.
- The expression in Python produces some value or result after being interpreted by the Python interpreter.
- Expressions are used to evaluate to a single value. They can be used in a variety of ways, such as in assignment statements, function arguments, and conditional statements. Expressions return a value and are often used to compute a value and assign it to a variable.
- **Examples:** In the following expression, the values 2 and 3 are added together and the result is stored in the variable y:

```
        x = 4                   # a statement
        y = 2 + 3                 # an expression
```

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

```
bacon = 22
bacon + 1
```

Ans-6

- In first line of given code , we are assigning the integer value 22 to the variable 'bacon', while in second line we are incrementing the value of 'bacon'.
- After execution of this line, the variable 'bacon' still contain the value '22' as show in code below:

```
In [10]:
```

```
# assigning int 22 to 'bacon'a variable
bacon = 22
# incrementing the value stored in bacon variable
bacon + 1

# printing the value stored in 'bacon'
print("The value of variable 'bacon' after execution is : {bacon}")
```

```
The value of variable 'bacon' after execution is : 22
```

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

```
        'spam' + 'spamspam'
        'spam' * 3
```

Ans-7

- The given two terms are the examples of concatenation.
- In first term, we are trying to concatenate the two strings **`spam`** & **`spamspam`** using addition operator whose result will be **`spamspamspam`** as show in code below.
- In second term, we are trying to concatinate the one strings **`spam`** & one int value **`3`** using multiplication operator , whose result will be **`spamspamspam`** as show in code below.

```
In [18]:
```

```
# First term is a concitation of two string values
'spam' + 'spamspam'
```

```
Out[18]:
```

```
'spamspamspam'
```

```
In [19]:
```

```
# Second term is a concitation of  string values and int value
'spam' * 3
```

```
Out[19]:
```

```
'spamspamspam'
```

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

`Variables:`

- Variable is a name that is used to refer to memory location i.e. variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Python variable is also known as an identifier and used to hold value.

`Properties of Variable`

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case (a-z), upper-case (A-Z), underscore, or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive; for example, my name, and MyName is not the same.

`Difference between 'eggs' & '100' variables`

- The **`eggs`** variable name is variable name, because variable names can begin with a alphabets i.e. it satisfied all the properties mentioned above.
- The **`100`** variable name is variable name, because variable names cannot begin with a number i.e. it does not satisfied all the properties mentioned above.

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

- In Python, **`int( )`** , **`float( )`** , and **`str( )`** functions can be used get integer, floating-point number, and string versions of the value passed to them.

`1. int( )`

- Python int() function returns an integer from a given object or converts a number in a given base to a decimal.
- **Example:**

```
# convert string value to integer
    age = "21"  # Here, 21 is in string format (i.e. "21").
    print("age =", int(age))  # This line will print the output in integeer format
(i.e. 10).
```

`ii. float( )`

- Python float() function is used to return a floating-point number from a number or a string representation of a numeric value.
- **Example:**

```
# convert integer value to float
    num = float(10)  # Here, 10 is in integer format (i.e. 10).
    print(num)  #  This line will print the output in floating point format (i.e.
10.0).
```

`iii. str( )`

- Python str() function returns an integer from a given object or converts a number in a given base to a decimal.
- **Example:**

```
# convert integer value to string
    num = 100  # Here, 10 is in integer format (i.e. 10).
    s = str(num)  # This line will convert int value in string value
    print(s, type(s))  #  This line will print the output in string format (i.e.
"100").
```

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

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

- Let's try to run the given code

```
In [23]:
```

```
'I have eaten ' + 99 + ' burritos.'     # this will show error.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [23], in <cell line: 1>()
----> 1 'I have eaten ' + 99 + ' burritos.'

TypeError: can only concatenate str (not "int") to str
```

- Here, we are trying to concatinate the `string` value with `integers` value using plus operator (i.e. `+`) But, we can not concatinate `string` value with `integers` value using plus operator (i.e. `+`)
- In order to concatinate he `string` value with `integers`, first we need to convert the `integer` value into `string` value using `str()` function, and then we can concatinate them.
- The detail explanation is in the code below:

In [14]:
```
# given statement  is 'I have eaten ' + 99 + ' burritos.'

term_1 = 'I have eaten'     # in string format.
term_2 = 99                 # in int format.
term_3 = 'burritos.'        # in string format.

# converting second term into string
term_22 = str(term_2)

# concatination of  all terms
output = f"The final string will be :{term_1 + ' ' + term_22 + ' ' + term_3}"  # + ' ' is used to add

# printing output string
print(output)
```

```
The final string will be : I have eaten  99  burritos.
```