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'	: Value
-87.8	: Value
-	: Expression
/	: Expression
+	: Expression
6	: Value

2. What is the difference between string and variable?

A string is a sequence of characters enclosed within either single ('') or double ("") quotes. It can contain letters, numbers, symbols, and spaces. Strings are used to represent textual data.

A variable is a name that refers to a value stored in the computer's memory. It acts as a container for holding different types of data, including strings. Variables allow you to store and manipulate data, making your code more dynamic and flexible.

3. Describe three different data types.

- Boolean Type: bool   Example : x = True
- Numeric Types: int   Example : x = 20
- Text Type: str   Example : x = "Hello World"

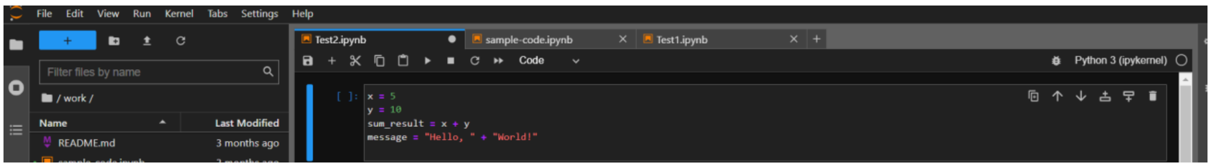
4. What is an expression made up of? What do all expressions do?

In Python, an expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a result. Expressions are the building blocks of Python programs and are used to perform computations, make decisions, and generate values.

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

An expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a single value.

Example:

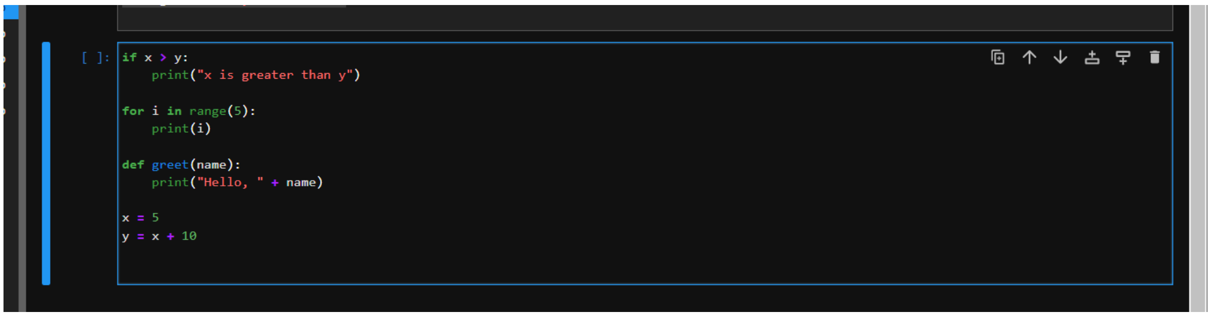


```
[ ]: x = 5
     y = 10
     sum_result = x + y
     message = "Hello, " + "World!"
```

In these examples, x + y and "Hello, " + "World!" are expressions. They produce values and can be used in assignments or as parts of larger expressions.

A statement is a line of code that performs an action, creates an effect, or changes the state of the program. Statements can include expressions, but they often involve more than just computation. Statements can define control structures (like if statements and loops), create functions, assignments, imports, and more.

Example:



```
[ ]: if x > y:
         print("x is greater than y")

     for i in range(5):
         print(i)

     def greet(name):
         print("Hello, " + name)

     x = 5
     y = x + 10
```

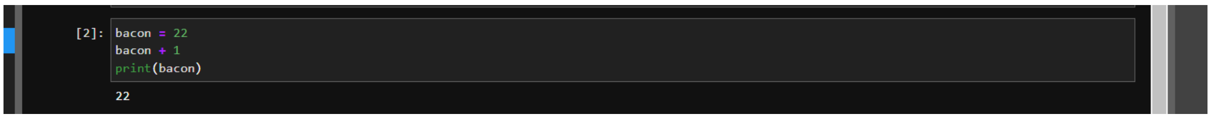
In these examples, if, for, def, and variable assignments are statements. They perform actions beyond just computing a value.

In summary, expressions are smaller units of code that produce values, while statements are more comprehensive and perform actions or control the flow of a program. Expressions can be part of statements (like in assignments), and statements often contain expressions, but the main distinction lies in their primary purpose and the level of activity they carry out within the program.

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

bacon = 22

bacon + 1



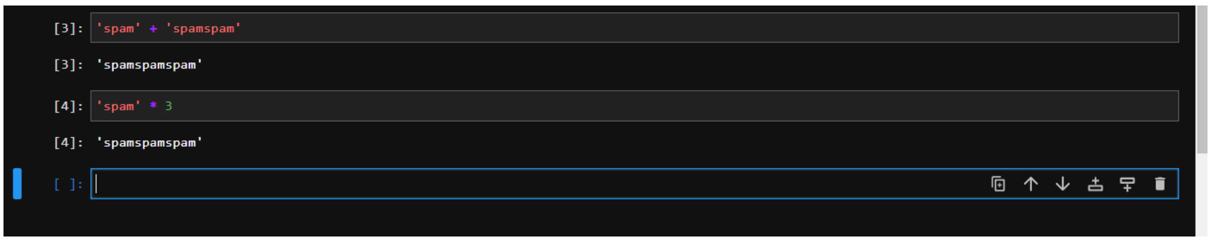
```
[2]: bacon = 22
     bacon + 1
     print(bacon)

     22
```

The variable bacon will still contain the value 22

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

'spam' + 'spamspam'

'spam' * 3



```
[3]: 'spam' + 'spamspam'
[3]: 'spamspamspam'
[4]: 'spam' * 3
[4]: 'spamspamspam'
[ ]:
```

- 'spam' + 'spamspam': This expression will result in the string 'spamspamspam'. The + operator performs string concatenation, which joins the two strings together.
- 'spam' * 3: This expression will result in the string 'spamspamspam'. The * operator, when used with a string and an integer, repeats the string a specified number of times. In this case, the string 'spam' is repeated three times.

So, both expressions yield the same result 'spamspamspam'

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

In Python, variable names must follow certain rules and conventions. This is to ensure clarity, consistency, and to avoid confusion with language elements like numbers, operators, and keywords. Let's break down why `eggs` is a valid variable name while `100` is invalid:

Valid Variable Name: `eggs`

- Variable names must start with a letter (a-z, A-Z) or an underscore (_).

- After the first character, variable names can contain letters, underscores, and digits.

- The name `eggs` starts with a letter and doesn't include any invalid characters. It adheres to the rules for variable naming.

Invalid Variable Name: `100`

- Variable names cannot start with a digit (0-9).

- Variable names cannot consist solely of digits.

- The name `100` starts with a digit, violating the rule that variable names must start with a letter or underscore.

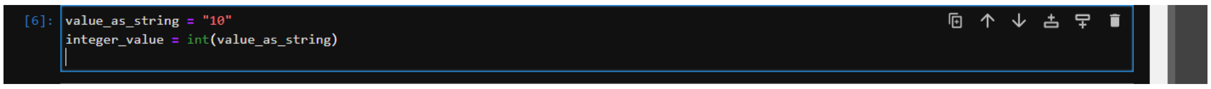
In short, variable names in Python must follow specific rules to be valid, and `eggs` meets these rules, while `100` does not.

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

Integer Conversion: int()

The int() function is used to convert a value to an integer (whole number). It can handle strings representing integers as well as floating-point numbers.

Example:

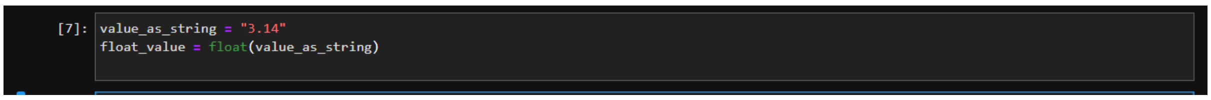


```
[6]: value_as_string = "10"
     integer_value = int(value_as_string)
```

Floating-Point Conversion: float()

The float() function is used to convert a value to a floating-point number (decimal number). It can handle strings representing floating-point numbers as well as integers.

Example:

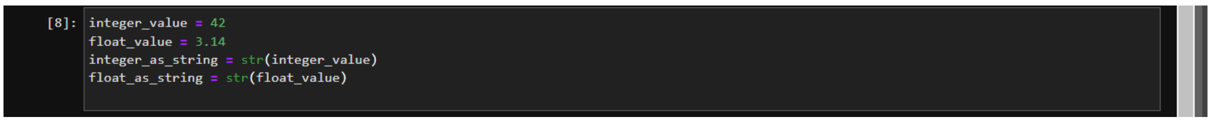


```
[7]: value_as_string = "3.14"
     float_value = float(value_as_string)
```

String Conversion: str()

The str() function is used to convert a value to its string representation. This is particularly useful when you want to concatenate a value with other strings or display it as text.

Example:



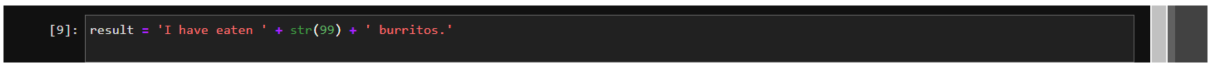
```
[8]: integer_value = 42
     float_value = 3.14
     integer_as_string = str(integer_value)
     float_as_string = str(float_value)
```

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

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

The expression 'I have eaten ' + 99 + ' burritos.' causes an error because we are trying to concatenate a string ('I have eaten ') with an integer (99) using the + operator. Python doesn't automatically convert the integer to a string in this context.

To fix this, you need to ensure that all values being concatenated are of the same type (strings). You can achieve this by explicitly converting the integer 99 to a string using the str() function:



```
[9]: result = 'I have eaten ' + str(99) + ' burritos.'
```