# Assignment 1:

## Q1.

**Values:**

'hello' (string value)

-87.8 (floating-point value)

**Expressions:**

\* (multiplication operator)

- (subtraction operator)

/ (division operator)

+ (addition operator)

## Q2.

**String:**

A string represents a sequence of characters. Like an array.

Strings are enclosed in quotation marks. [“ “]

**Variable:**

A variable is used to assign a value. Variables are used to store and manipulate.

A variable can hold different types of values, including strings, numbers, or other data types.

Variables are typically assigned a value using an assignment operator (=).

## Q3.

**Integer (int):**

The integer data type represents whole numbers without any fractional part.

Examples : -5, 0, 42, 1000

Integers can be used in mathematical calculations and logical operations.

**String (str):**

The string data type represents a sequence of characters.

Strings are enclosed in quotation marks (single or double).

Examples : "Hello", 'Python', "42"

**Float:**

The float data type represents decimal numbers or numbers with fractional parts.

Floats are used when more precision is needed in numerical calculations or when representing real-world values with decimal places.

Example: 3.14

## Q4

An expression is made up of different components that work together to produce a value. It consists of variables, values, and operators.

Variables: Variables are like containers that hold values. They can store different types of data such as numbers or text.

Values: Values are the actual data that the expression operates on. They can be numbers (like 5 or 3.14) or text (like "Hello" or "Python").

Operators: Operators are symbols that perform operations on variables and values.

Expressions, when evaluated, produce a single value as the result. They can perform mathematical calculations or carry out logical operations.

## Q5

Expression:

An expression is a combination of variables, values, and operators that evaluates to a single value.

It produces a result or value when it is executed.

Examples of expressions: 2 + 3, x \* y, "Hello" + "World"

Statement:

A statement is a complete instruction or action that performs a specific task.

It represents a line or block of code that performs an action or controls the flow of a program.

Examples of statements: if statements, for loops, function definitions.

## Q6

After running the given code, the variable bacon would still contain its original value, which is 22. It doesn't modify the existing value.

## Q7

The values of the two terms would be:

'spam' + 'spamspam':

The resulting value is 'spamspamspam'.

'spam' \* 3:

The resulting value is 'spamspamspam'.

## Q8

In Python, variable names must follow certain rules to be considered valid.

"eggs" is a valid variable name because it starts with a letter and only contains valid characters (letters in this case).

"100" is an invalid variable name because it starts with a digit (1). Variable names cannot start with a digit in Python.

## Q9

To convert a value to different data types in Python, we can use the following three functions:

int():

The int() function is used to convert a value into an integer data type.

float():

The float() function is used to convert a value into a floating-point number data type.

str():

The str() function is used to convert a value into a string data type.

## Q10

The expression 'I have eaten ' + 99 + ' burritos.' causes an error because it attempts to concatenate a string with an integer value directly, which is not allowed in Python. The + operator is used for string concatenation when both operands are strings. However, in this case, the value 99 is an integer, not a string.

To fix the error and perform the concatenation correctly, we need to convert the integer value to a string before concatenating it with the other strings. we can achieve this by using the str() function to convert the integer to a string. Here's the corrected expression:

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

# Assignment 2:

## Q1

The Boolean data type in Python has two possible values: True and False. These values represent the two states of Boolean logic: true and false.

In Python, we write these Boolean values as True and False (Capital T and F).

## Q2

The three different types of Boolean operators in Python are:

**Logical AND (and):**

The logical AND operator returns True if both operands are True, and False otherwise.

**Logical OR (or):**

The logical OR operator returns True if at least one of the operands is True, and False if both operands are False.

**Logical NOT (not):**

The logical NOT operator negates the value of the operand. If the operand is True, the NOT operator returns False, and vice versa.

## Q3

Here are the truth tables for each Boolean operator:

Logical AND (and):

Operand 1 Operand 2 Result

True True True

True False False

False True False

False False False

Logical OR (or):

Operand 1 Operand 2 Result

True True True

True False True

False True True

False False False

Logical NOT (not):

Operand Result

True False

False True

## Q4

**(5 > 4) and (3 == 5):**

(5 > 4) evaluates to True.

(3 == 5) evaluates to False.

The expression becomes True and False, which evaluates to False.

**not (5 > 4):**

(5 > 4) evaluates to True.

The not operator negates the value, so True becomes False.

**(5 > 4) or (3 == 5):**

(5 > 4) evaluates to True.

(3 == 5) evaluates to False.

The expression becomes True or False, which evaluates to True.

**not ((5 > 4) or (3 == 5)):**

not((5 > 4)) evaluates to False.

(3 == 5) evaluates to False.

The expression becomes not (True or False), which evaluates to False.

**(True and True) and (True == False):**

(True and True) evaluates to True.

(True == False) evaluates to False.

The expression becomes True and False, which evaluates to False.

**(not False) or (not True):**

not False evaluates to True.

not True evaluates to False.

The expression becomes True or False, which evaluates to True.

The values of the given expressions are:

False

False

True

False

False

True

## Q5

The six comparison operators in Python are:

**Equal to (==):**

Checks if two operands are equal.

Example: 5 == 5 evaluates to True.

**Not equal to (!=):**

Checks if two operands are not equal.

Example: 5 != 3 evaluates to True.

**Greater than (>):**

Checks if the left operand is greater than the right operand.

Example: 5 > 3 evaluates to True.

**Less than (<):**

Checks if the left operand is less than the right operand.

Example: 3 < 5 evaluates to True.

**Greater than or equal to (>=):**

Checks if the left operand is greater than or equal to the right operand.

Example: 5 >= 5 evaluates to True.

**Less than or equal to (<=):**

Checks if the left operand is less than or equal to the right operand.

Example: 3 <= 5 evaluates to True.

## Q6

The equal to (==) operator and the assignment (=) operator serve different purposes in Python and can be distinguished as :

**Equal to (==) operator:**

The equal to operator (==) is used for comparison.

It checks if the values on both sides of the operator are equal.

It returns True if the values are equal and False otherwise.

**Assignment (=) operator:**

The assignment operator (=) is used for variable assignment.

It assigns a value on the right side to a variable on the left side.

When we want to compare two values for equality, we use the equal to (==) operator. For example, in an if statement, we might use if x == 5: to check if x is equal to 5 before executing a block of code based on that condition.

When we want to assign a value to a variable, we use the assignment (=) operator. For example, we might use x = 5 to assign the value 5 to the variable x.

## Q7

In the given code, the three blocks can be identified based on the indentation levels. In Python, indentation is used to define blocks of code. Here's the code with the blocks:

spam = 0

# Block 1

if spam == 10:

print('eggs')

# Block 2

if spam > 5:

print('bacon')

else:

# Block 3

print('ham')

print('spam')

print('spam')

## Q8

spam = 1

if spam == 1:

print("Hello")

elif spam == 2:

print("Howdy")

else:

print("Greetings!")

## Q9

Ctrl + C

Pressing Ctrl + C , causes it to terminate. This key combination is commonly used to break out of an infinite loop or stop the execution of a program that is not responding.

## Q10

In Python, break and continue are control flow statements used within loops to alter the flow of execution. They serve different purposes:

**break:**

When encountered within a loop (such as a for or while loop), the break statement immediately terminates the loop.

It allows we to exit the loop prematurely, regardless of any remaining iterations.

After encountering a break, the program execution continues from the next statement after the loop.

**continue:**

When encountered within a loop, the continue statement skips the remaining code inside the loop for the current iteration and moves to the next iteration.

It allows we to skip specific iterations and continue with the next iteration of the loop.

After encountering continue, the program execution jumps to the beginning of the loop for the next iteration.

## Q11

**range(10):**

In this case, range(10) generates a sequence of numbers starting from 0 (default) and ending at 10 (exclusive) with a default step of 1.

The start value is not explicitly mentioned, so it defaults to 0.

The stop value is 10, which means the sequence will include numbers up to 9.

The step value is not explicitly mentioned, so it defaults to 1.

**range(0, 10):**

In this case, range(0, 10) generates a sequence of numbers starting from 0 (inclusive) and ending at 10 (exclusive) with a step of 1.

The start value is explicitly mentioned as 0.

The stop value is 10, which means the sequence will include numbers up to 9.

The step value is not explicitly mentioned, so it defaults to 1.

**range(0, 10, 1):**

In this case, range(0, 10, 1) generates a sequence of numbers starting from 0 (inclusive) and ending at 10 (exclusive) with a step of 1.

The start value is explicitly mentioned as 0.

The stop value is 10, which means the sequence will include numbers up to 9.

The step value is explicitly mentioned as 1.

## Q12

**#** **Using a for loop**

for i in range(1, 11):

print(i)

**# Using a while loop**

i = 1

while i <= 10:

print(i)

i += 1

## Q13

After importing the module named spam, we can call the function named bacon() using the following syntax:

**import spam**

**spam.bacon()**

# Assignment 3:

## Q1

Functions are advantageous to have in wer programs for several reasons, especially for beginners. Here are some simple explanations:

Reuse of code: Functions allow we to write a block of code once and reuse it multiple times in wer program. Instead of repeating the same code over and over, we can define a function and call it whenever we need that specific functionality.

Modularity: Functions help in organising wer code into logical and manageable chunks. Each function can perform a specific task, making wer program easier to understand and maintain. we can focus on writing small, self-contained functions that work together to solve a larger problem.

Readability: By breaking wer code into functions, we can give meaningful names to each function, making wer code more readable and self-explanatory. Functions also help in dividing complex tasks into smaller, manageable parts, making the overall code easier to understand.

Code organisation: Functions help in organising wer code by separating different functionalities into modular units. This makes it easier to navigate through wer codebase, locate specific functionality, and make changes or improvements without affecting other parts of the program.

Code reusability: Functions can be reused in different programs or projects. Once we write a useful function, we can easily import it into other programs, saving time and effort by leveraging wer existing code.

Overall, functions provide structure, reusability, and simplicity to wer programs. They promote code reuse, improve readability, and make it easier to manage and maintain wer codebase. Functions are a fundamental building block in programming and play a vital role in creating efficient, organised, and scalable programs.

## Q2

The code inside a function runs when the function is called, not when it is specified or defined.

## Q3

In Python, the def statement is used to create a function. The def keyword stands for "define" and is followed by the name of the function, a pair of parentheses ().

## Q4

The difference between a function and a function call lies in their respective roles and actions within a program.

Function: A function is a block of reusable code that performs a specific task or a set of tasks. It is defined using the def statement and has a name, optional parameters, and a body of code. Functions allow we to organise wer code by encapsulating specific functionality.

Function Call: A function call is the act of calling or executing a function. It is the point in the program where the function's code is actually executed. When a function is called, the program flow transfers to the function, executes the code inside the function body, and then returns to the point immediately after the function call.

## Q5

In a Python program, there is typically one global scope and multiple local scopes.

Global Scope: The global scope refers to the outermost level of the program. It is the scope in which variables, functions, and classes defined outside any function or class are accessible. The global scope exists throughout the entire program, and any variables defined in this scope can be accessed from any part of the program.

Local Scopes: Local scopes are created when a function is called or when a block of code, such as a loop or conditional statement, is executed. Each function or block of code creates its own local scope. Local scopes are temporary and exist only within the context of the function or block of code in which they are created. Variables defined in a local scope are accessible only within that scope and are not accessible outside of it.

## Q6

When a function call returns, the local variables defined within that function's local scope cease to exist. They are destroyed, and their values are no longer accessible.

In Python, each time a function is called, a new local scope is created for that function. Any variables created within that local scope, including parameters and local variables defined within the function, only exist within the context of that function call.

## Q7

In Python, the return is used in a function to specify the value that should be sent back or "returned" to the code that called the function. It marks the end of the function's execution and sends the specified value back as the result of the function call.

A return value is not an expression itself, but it can be used as part of an expression in the calling code. For example, we can directly use the return value in calculations or assign it to a variable for later use.

## Q8

If a function does not have a return statement, the return value of a call to that function is None.

## Q9

**Using the global keyword:** The global keyword tells Python that we want to refer to the global variable, even if there is a local variable with the same name.

## Q10

In Python, the data type of None is NoneType. None is a special value that represents the absence of a value or the lack of a specific value.

## Q11

The sentence import areallwerpetsnamederic imports a module named areallwerpetsnamederic. This module does not exist, so the import statement will fail.

## Q12

spam.bacon()

## Q13

To save a program from crashing when it encounters an error, we can use error handling techniques to catch and handle the errors. In Python, this is achieved using try and except blocks.

## Q14

The purpose of the try clause in Python is to enclose a block of code that may potentially raise an error. It allows we to specify code that might cause an exception and provides a structured way to handle potential errors. If an error occurs within the try block, the execution of the code within the try block is immediately stopped, and the program jumps to the corresponding except. The purpose of the except clause is to define how to handle specific types of errors that may occur within the associated try block. It allows we to catch and handle specific exceptions, providing alternative code or actions to be executed when a particular error occurs.

Assignment 4:

## Q1

In Python, [] represents an empty list. A list is a collection of items enclosed in square brackets [] and separated by commas. when there are no items between the brackets, it denotes an empty list.

## Q2

spam = [2, 4, 6, 8, 10]

# Insert 'hello' as the third value in spam

spam.insert(2, 'hello')

# Print spam

print(spam)

## Q3

The expression int(int('3' \* 2) / 11) evaluates to 3. This is because int('3' \* 2) first converts the string '3' to the integer 3, and then int(3 / 11) divides 3 by 11 and returns the integer 3.

The expression spam[3] then accesses the third element of the list spam, which is the value 'd'.

Therefore, the value of spam[int(int('3' \* 2) / 11)] is 'd'.

## Q4

The value of spam[-1] is 'd'.

The expression spam[-1] accesses the last element of the list spam. Since the list spam only has four elements, the expression spam[-1] will return the value 'd'.

## Q5

The value of spam[:2] is ['a', 'b'].

The expression spam[:2] returns a slice of the list spam starting from the first element and ending at the second element (not including the second element). Since the list spam only has four elements, the expression spam[:2] will return a list with the first two elements of the spam list.

## Q6

The value of bacon.index('cat') is 1.

The index() method of a list returns the index of the first occurrence of the specified value in the list. In this case, the first occurrence of the value 'cat' in the list bacon is at index 1.

## Q7

The append() method of a list adds the specified value to the end of the list. In this case, the append() method will add the value 99 to the end of the list bacon.

[3.14, 'cat,', 11, 'cat,', True, 99]

## Q8

The remove() method of a list removes the first occurrence of the specified value from the list. In this case, the remove() method will remove the first occurrence of the value 'cat' from the list bacon.

[3.14, 11, 'cat,', True]

## Q9

The list concatenation and list replication operators in Python are the + and \* operators, respectively.

The + operator concatenates two lists, creating a new list that contains the elements from both lists.

The \* operator replicates a list, creating a new list that contains the specified number of copies of the original list.

## Q10

**append() method:**

The append() method is used to add an element at the end of a list. It takes a single argument, which is the value to be added, and appends that value as a new element to the list.

**insert() method:**

The insert() method is used to insert an element at a specific index within the list. It takes two arguments: the index where the element should be inserted and the value to be inserted.

## Q11

**remove() method:**

The remove() method is used to remove the first occurrence of a specified value from a list. It takes a single argument, which is the value to be removed.

**pop() method:**

The pop() method is used to remove an item from a list at a specific index.

## Q12

List values and string values are identical in the following ways:

They are both sequences, meaning that they can be indexed, sliced, concatenated and iterated over.

They can both be used in for loops.

They can both be used in conditional statements.

## Q13

Tuples and lists are both sequence data types in Python, but they have some key differences.

**Mutability**

The main difference between tuples and lists is that tuples are immutable, while lists are mutable. This means that the elements of a tuple cannot be changed after it is created, while the elements of a list can be changed.

**Syntax:**

Tuples are defined using parentheses () or without any brackets, separating the elements with commas. Lists, on the other hand, are defined using square brackets [ ].

## Q14

To type a tuple value that only contains the integer 42, we can use the following code:

tuple\_value = (42,)

## Q15

To get a list value's tuple form, we can use the tuple() function. The tuple() function takes an iterable, such as a list, and returns a tuple containing the elements of the iterable.

To get a tuple value's list form, we can use the list() function. The list() function takes an iterable, such as a tuple, and returns a list containing the elements of the iterable.

## Q16

Variables that "contain" list values in Python do not actually contain the lists themselves. Instead, they contain references to the lists.

In Python, variables are essentially labels or names that refer to objects in memory. When we assign a list to a variable, the variable holds a reference to the memory location where the list is stored. This means that the variable points to the list rather than directly containing the list itself.

## Q17

The copy.copy() and copy.deepcopy() functions are both used to copy objects in Python. However, they work in different ways.

copy.copy() creates a shallow copy of an object. This means that the new object is a reference to the original object. If we change the original object, the change will be reflected in the new object.

copy.deepcopy() creates a deep copy of an object. This means that the new object is a complete copy of the original object. If we change the original object, the change will not be reflected in the new object.

Assignment 5

## Q1

An empty dictionary in Python is represented by a pair of curly braces {}.

my\_dict = {}

## Q2

The value of a dictionary with the key 'foo' and the value 42 would be 42.

my\_dict = {'foo': 42}

## Q3

The most significant distinction between a dictionary and a list is the way they store and organize data.

**List**: A list is an ordered collection of items, where each item is assigned an index based on its position in the list. Lists maintain the order of elements as they are added.

**Dictionary**: A dictionary is an unordered collection of key-value pairs. Each key is unique within the dictionary and is used to access its corresponding value. Unlike a list, a dictionary does not maintain any specific order of its elements.

## Q4

If we try to access spam['foo'] and spam is {'bar': 100}, it will raise a KeyError.

## Q5

The expressions 'cat' in spam and 'cat' in spam.keys() check for the presence of the key 'cat' in the dictionary spam, but they differ in their implementation.

'cat' in spam:

This expression checks if the key 'cat' exists directly in the dictionary spam. It returns a boolean value (True or False) indicating whether the key is present as a direct key in the dictionary.

'cat' in spam.keys():

This expression retrieves all the keys of the dictionary spam using the keys() method, which returns a view object containing all the keys in the dictionary.

The expression then checks if the key 'cat' exists in this view object of keys. It also returns a boolean value indicating whether the key is present in the keys of the dictionary.

## Q6

The difference between the expressions 'cat' in spam and 'cat' in spam.values() is that the first expression checks whether the string 'cat' is a key in the dictionary spam, while the second expression checks whether the string 'cat' is a value in the dictionary spam.

## Q7

The shortcut for the code if 'color' not in spam: spam['color'] = 'black' is spam.setdefault('color', 'black').

The setdefault() method of a dictionary takes two arguments: a key and a default value. If the key does not exist in the dictionary, then the key is added to the dictionary with the default value. If the key already exists in the dictionary, then the value of the key is not changed.

## Q8

To "pretty print" dictionary values in Python, we can use the pprint module and the pprint() function. The pprint() function takes an object as input and returns a string representation of the object with appropriate indentation and line breaks.

Assignment 6:

## Q1

Escape characters are special characters used in strings to represent certain characters or symbols that are difficult to include directly. Escape characters are preceded by a backslash (\) in Python.

Ex- \n: Represents a newline character. It is used to insert a line break.

## Q2

The escape character \n stands for a newline character, and the escape character \t stands for a tab character.

\n: It is used to insert a newline or line break in a string. When encountered in a string, it moves the cursor to the beginning of the next line.

\t: It is used to insert a horizontal tab or tabulation in a string. When encountered in a string, it adds horizontal spacing equivalent to a tab.

## Q3

To include a backslash character (\) in a string, we need to use the escape character \\. The double backslash represents a single literal backslash in the resulting string.

## Q4

The single quote character in the string "Howl's Moving Castle" is not causing a problem because the string itself is enclosed in double quotes ("). In Python, strings can be enclosed in either single quotes or double quotes, and they can contain the opposite type of quote character without causing any issues.

## Q5

Using triple quotes (''' or """) to create a multiline string, where line breaks are automatically preserved.

## Q6

'Hello, world!'[1] is 'e'.

'Hello, world!'[0:5] is 'Hello'.

'Hello, world!'[:5] is 'Hello'.

'Hello, world!'[3:] is 'lo, world!'.

## Q7

'Hello'.upper() is 'HELLO'.

'Hello'.upper().isupper() is True.

'Hello'.upper().lower() is 'hello'.

## Q8

'Remember, remember, the fifth of July.'.split() is ['Remember,', 'remember,', 'the', 'fifth', 'of', 'July.'].

'-'.join('There can only one.'.split()) is 'There-can-only-one.'.

## Q9

The rjust() method right-justifies a string, meaning that it pads the string with spaces on the left side until it reaches the specified width. The ljust() method left-justifies a string, meaning that it pads the string with spaces on the right side until it reaches the specified width. The centre() method centres a string, meaning that it pads the string with spaces on both the left and right sides until it reaches the specified width.

## Q10

The best way to remove whitespace characters from the start or end of a string is to use the strip() method. The strip() method removes leading (start) and trailing (end) whitespace characters from a string.

Assignment 7:

## Q1

The feature responsible for generating regular expression (Regex) objects in Python is the re module. The re module provides functions and methods for working with regular expressions.

## Q2

Raw strings are often used in Regex objects to handle backslashes (\) effectively. In Python, backslashes are used as escape characters to represent special characters or character sequences (e.g., \n for newline, \t for tab). However, in regular expressions, backslashes have their own special meaning, and using them as escape characters in string literals can lead to unintended behaviour.

To avoid potential issues and to ensure that backslashes are treated as literal characters within regular expressions, raw strings are commonly used.

## Q3

The search() method from the re module in Python returns a match object if it finds a match for the specified pattern within the given string. If no match is found, the method returns None.

## Q4

To get the actual strings that match the pattern from a Match object in Python, we can use the group() method. The group() method returns the substring of the input string that was matched by the regular expression pattern.

## Q5

In the regular expression pattern r'(\d\d\d)-(\d\d\d-\d\d\d\d)', the group zero (group(0)) covers the entire match, including all the captured groups.

Group 1 (group(1)) corresponds to the first capturing group (\d\d\d), which matches three consecutive digits.

Group 2 (group(2)) corresponds to the second capturing group (\d\d\d-\d\d\d\d), which matches a group of three digits followed by a hyphen and then four digits.

## Q6

To match literal parentheses and periods in a regex pattern, we need to escape them using a backslash (\).

To tell a regex that we want to match literal parentheses, we can use \( to match an opening parenthesis and \) to match a closing parenthesis.

## Q7

The findall() method of the re module in Python returns a list of all non-overlapping matches of a pattern in a string. The structure of the returned list depends on the presence of capturing groups (parentheses) in the regular expression pattern.

## Q8

In standard regular expressions, the | character is used as the alternation or OR operator. The | operator is used to create a logical OR condition.

## Q9

The | operator is used to create a logical OR condition in regular expressions. It matches either the pattern on the left side of the | or the pattern on the right side. It allows we to specify multiple alternative patterns, and it matches any one of the patterns.

## Q10

The + and \* characters in regular expressions are both quantifiers, which means that they indicate how many times the preceding expression should be matched.

The "+" character means "one or more" and is used to match one or more occurrences of the preceding pattern.

The \* character indicates that the preceding expression can be matched zero or more times.

## Q11

The difference between {4} and {4,5} in regular expressions is the number of times the preceding expression must be matched. The curly braces are used to represent quantifiers in regular expressions. The quantifier {4} indicates that the preceding expression must be matched exactly 4 times.

The quantifier {4,5} indicates that the preceding expression must be matched at least 4 times but not more than 5 times.

## Q12

In regular expressions, the shorthand character classes \d, \w, and \s are special patterns that match specific sets of characters:

\d represents any digit character (0-9). It matches a single digit.

\w represents any word character. It matches alphanumeric characters (a-z, A-Z, 0-9) and underscores (\_).

\s represents any whitespace character. It matches spaces, tabs, newlines, and other whitespace characters.

## Q13

In regular expressions, the shorthand character classes \D, \W, and \S are negations of the corresponding shorthand character classes \d, \w, and \s. They represent the inverse sets of characters:

\D represents any non-digit character. It matches any character that is not a digit (0-9).

\W represents any non-word character. It matches any character that is not an alphanumeric character or underscore.

\S represents any non-whitespace character. It matches any character that is not a space, tab, newline, or other whitespace character.

## Q14

The main difference between them is the preference for matching shorter or longer sequences. If there are multiple occurrences of the pattern, the non-greedy .\*? will match the shortest possible substring, while the greedy .\* will match the longest possible substring.

## Q15

To match both numbers and lowercase letters in a regular expression using a character class, we can use the following syntax:

[0-9a-z]

[0-9] matches any digit from 0 to 9.

[a-z] matches any lowercase letter from a to z.

## Q16

To make a regular expression case insensitive in Python, we can use the re.IGNORECASE flag as an argument to the matching function. The IGNORECASE flag is passed to the re.compile() function when we are compiling the regular expression.

## Q17

In regular expressions, the . (dot) character normally matches any character except for a newline character (\n). It matches any single character in the input string.

However, if the re.DOTALL flag is passed as the second argument to the re.compile() function then the . character will match any character including newline characters (\n). It will match across multiple lines in the input string.

## Q18

If numReg = re.compile(r'\d+'), then numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') will return the string 'X drummers, X pipers, five rings, X hen'.

## Q19

Passing re.VERBOSE as the second argument to re.compile() allows we to add comments and whitespace to the regular expression pattern for better readability without affecting its functionality.

## Q20

(r'^\d{1,3}(,\d{3})\*$')

^ matches the start of the string.

\d{1,3} matches 1 to 3 digits.

(,\d{3})\* matches a comma followed by exactly 3 digits, and this group can repeat zero or more times.

$ matches the end of the string.

## Q21

(r'^[A-Z][a-zA-Z]\* Watanabe$')

^ matches the start of the string.

[A-Z] matches an uppercase letter (the first letter of the first name).

[a-zA-Z]\* matches zero or more lowercase or uppercase letters (the rest of the first name).

matches a space between the first name and last name.

Watanabe matches the last name.

$ matches the end of the string.

## Q22

(r'^(Alice|Bob|Carol) (eats|pets|throws) (apples|cats|baseballs)\.$', re.IGNORECASE)

^ matches the start of the string.

(Alice|Bob|Carol) matches one of the given names: Alice, Bob, or Carol.

(eats|pets|throws) matches one of the given actions: eats, pets, or throws.

(apples|cats|baseballs) matches one of the given objects: apples, cats, or baseballs.

\. matches the period at the end of the sentence.

$ matches the end of the string.

re.IGNORECASE enables case-insensitive matching.

Assignment 8:

## Q1

The Python Standard Library is separate from PyInputPlus. PyInputPlus is not part of the Python Standard Library. It is an external library, specifically designed for input validation and handling user input in a convenient way.

## Q2

The import statement import pyinputplus as pypi is a way to import the PyInputPlus library and assign it an alias pypi. This alias allows we to refer to the library using a shorter and more convenient name in wer code. By using the pypi alias, we can save typing effort and make wer code more compact when referencing PyInputPlus functions and classes.

## Q3

inputInt(): This function is used when we specifically want the user to input an integer value. It will prompt the user for input and continue to re-prompt until a valid integer is entered. If the user enters a non-integer value, an error message will be displayed, and the user will be prompted again.

inputFloat(): This function is used when we want the user to input a floating-point number. It works similar to inputInt(), but it allows the user to input decimal numbers. It will validate the input and re-prompt until a valid float value is entered. If the user enters a non-numeric value an error message will be displayed.

## Q4

To ensure that the user enters a whole number between 0 and 99 using PyInputPlus, we can use the inputInt() function along with the min and max arguments.

num = pypi.inputInt("Enter a number between 0 and 99: ", min=0, max=99)

## Q5

allowRegexes: This argument accepts a list of regular expressions. PyInputPlus will only allow inputs that match any of the regular expressions specified in allowRegexes. If the user enters input that doesn't match any of the allowed patterns, an error message will be displayed.

blockRegexes: This argument also accepts a list of regular expressions. PyInputPlus will block inputs that match any of the regular expressions specified in blockRegexes. If the user enters input that matches any of the blocked patterns, an error message will be displayed.

## Q6

When using inputStr(limit=3) in PyInputPlus, it means that the user is allowed to enter a blank input (an empty string) up to three times before an exception is raised.

## Q7

If inputStr(limit=3, default='hello') is used in PyInputPlus and the user enters a blank input (an empty string) three times consecutively, the function will return the default value specified, which is 'hello'.

Assignment 9:

## Q1

A relative path refers to the path of a file or directory relative to the current working directory or another reference point. It specifies the location of a file or directory relative to the current location rather than specifying the complete path from the root directory. Relative paths are useful when referring to files or directories within the same project or directory structure without the need to specify the complete path from the root directory.

## Q2

My OS = MacOS

On the Mac operating system, an absolute path starts with a forward slash (/). The forward slash represents the root directory of the file system.

## Q3

The functions os.getcwd() and os.chdir() are part of the os module in Python and are used for working with the current working directory.

os.getcwd(): This function returns a string representing the current working directory. It returns the path of the directory from which the script is being executed.

os.chdir(): This function is used to change the current working directory to the specified path. It takes a string argument path which represents the new directory to be set as the current working directory.

## Q4

In file systems, the . and .. folders are special entries that represent relative directories.

. : This represents the current directory. It is used to refer to the directory itself.

.. : This represents the parent directory. It is used to refer to the directory one level above the current directory.

## Q5

In the file path C:\bacon\eggs\spam.txt:

The directory name (dir name) is C:\bacon\eggs. It refers to the path of the directory that contains the file.

The base name is spam.txt. It refers to the name of the file itself, without the directory path.

## Q6

The three "mode" arguments that can be passed to the open() function in Python are:

"r": This mode is used for reading from a file. It opens the file in read-only mode, and if the file does not exist, it raises a FileNotFoundError.

"w": This mode is used for writing to a file. It opens the file in write mode, creates a new file if it doesn't exist, or truncates the file if it exists. If the file already exists, its contents will be overwritten.

"a": This mode is used for appending to a file. It opens the file in append mode, creating a new file if it doesn't exist. If the file exists, new data will be added to the end of the file.

## Q7

If an existing file is opened in write mode ("w"), the file's contents will be completely overwritten. Opening a file in write mode truncates the file to zero length before writing new data to it. If the file does not exist, a new file with the specified name will be created.

## Q8

read(): This method reads the entire contents of a file as a single string. It reads from the current position in the file until the end of the file. The resulting string includes all the characters in the file, including newline characters ('\n').

readlines(): This method reads the contents of a file line by line and returns a list of strings, where each string represents a line from the file. The newline characters ('\n') are included in each line. This method is useful when we want to process the file line by line or when we need to access individual lines.

## Q9

A shelf value resembles a dictionary value. It has keys and values, along with keys() and values() methods that work similarly to the dictionary methods of the same names.

Assignment 10:

## Q1

The shutil.copy() and shutil.copytree() functions in Python's shutil module are used for copying files and directories, respectively.

shutil.copy(src, dst): This function is used to copy a single file from the source location to the destination location. It takes two arguments: the path of the source file (src) and the destination path (dst) where the file will be copied.

shutil.copytree(src, dst): This function is used to copy an entire directory tree from the source location to the destination location. It takes two arguments: the path of the source directory (src) and the destination path (dst) where the directory tree will be copied.

## Q2

The os.rename() function is used to rename files in Python. The os.rename() function takes two arguments: the current name of the file (including the path) and the new name we want to assign to the file (including the path if desired). It renames the file to the new name specified.

## Q3

send2trash: This function from the send2trash module moves a file or directory to the system's trash or recycle bin instead of permanently deleting it.

shutil.rmtree: This function from the shutil module is used to remove a directory and its contents. It permanently deletes the specified directory.

## Q4

The ZipFile method equivalent to File objects' open() method is ZipFile.open(). The ZipFile.open() method takes two arguments: the name of the file to open and the mode in which to open it. The mode can be "r" for read, "w" for write, "a" for append.

## Q5

import os

import shutil

def search\_and\_copy\_files(source\_folder, target\_folder, file\_extension):

for root, dirs, files in os.walk(source\_folder):

for file in files:

if file.endswith(file\_extension):

source\_path = os.path.join(root, file)

target\_path = os.path.join(target\_folder, file)

shutil.copy2(source\_path, target\_path)

print(f"Copied {file} to {target\_folder}")

# Example usage

source\_folder = "/path/to/source/folder"

target\_folder = "/path/to/target/folder"

file\_extension = ".pdf"

search\_and\_copy\_files(source\_folder, target\_folder, file\_extension)

Assignment 11:

## Q1

assert spam >= 0, "spam cannot be a negative integer"

## Q2

assert eggs.lower() != bacon.lower(), "eggs and bacon are the same"

## Q3

assert False, "This assertion always triggers an AssertionError"

## Q4

import logging

logging.basicConfig(level=logging.DEBUG)

## Q5

import logging

logging.basicConfig(filename='programLog.txt', level=logging.DEBUG)

## Q6

DEBUG: Detailed information for debugging purposes.

INFO: General information about the program's execution.

WARNING: Indication of potential issues or unexpected behaviour that is not necessarily an error.

ERROR: An error has occurred, but the program can still continue.

CRITICAL: A critical error has occurred, and the program may not be able to continue.

## Q7

logging.disable(logging.CRITICAL)

## Q8

There are several reasons why using logging messages is better than using print() to display the same message.

Logging messages are more flexible. we can control the level of logging, which means that we can only see the messages that we are interested in. we can also configure logging to send messages to different destinations, such as a file or a database.

. we can use the same logging configuration in different environments, such as development, staging, and production. Logging messages are more secure. we can configure logging to encrypt messages, which can help to protect sensitive information. print() statements can print a lot of output, which can slow down wer code. Logging messages are only printed when they are needed, which can improve the performance of wer code.

## Q9

Step Over- Executes the current line of code and then proceeds to the next line. If the current line of code is a function call, the function call will be executed as a single step.

Step In- Executes the current line of code and then steps into any function calls that are made on that line.

Step Out- Executes the current line of code and then steps out of any function calls that are currently active.

## Q10

After clicking the Continue button in a debugger, the debugger will stop if there is a Breakpoint, Exception, End of Program or Interruption.

## Q11

A breakpoint is an essential aspect of debugging. A breakpoint is a designated point in wer code where we want the debugger to pause execution, allowing we to inspect the program's state, analyse variables, and step through the code line by line.Breakpoints are useful for isolating specific sections of code that we want to investigate more closely or for pinpointing the source of an issue.

Assignment 12:

## Q1

For PdfFileReader(): The PDF file object should be opened in "rb" mode, which stands for "read binary." This mode is used to indicate that the file will be read as a binary file, which is necessary for reading PDF files.

For PdfFileWriter(): The PDF file object should be opened in "wb" mode, which stands for "write binary." This mode is used to indicate that the file will be written as a binary file. Opening the file in "wb" mode allows the PyPDF2 library to write PDF content into the file.

## Q2

To get a Page object for page 5 from a PdfFileReader object, we can use the getPage() method and pass the page number as an argument.

page\_5 = PdfFileReader.getPage(4)

## Q3

The numPages variable in the PdfFileReader object stores the number of pages in the PDF document.

## Q4

Before obtaining Page objects from a PdfFileReader object that has an encrypted PDF with the password "swordfish", we need to decrypt the PDF by calling the decrypt() method of the PdfFileReader object and passing the password as an argument.

## Q5

The rotate() method is used to rotate the page in the PDF file. Can use the rotateClockwise() or rotateCounterClockwise() methods of the PageObject.

## Q6

A Paragraph object represents a single paragraph of text in a document. It typically consists of multiple runs of text. A paragraph is a logical unit that separates blocks of text and is often delimited by line breaks or paragraph breaks.

A Run object represents a continuous span of text within a paragraph that shares the same formatting properties. It is a smaller unit within a paragraph that can be styled independently.

## Q7

To obtain a list of Paragraph objects from a Document object in python, we can use the paragraphs attribute. **.paragraphs** returns a list of all the Paragraph objects in the document.

## Q8

The Run object in python-docx has the variables bold, underline, italic, strike, and outline to represent different formatting options for text. The bold, underline, italic, strike, and outline variables control the formatting of the text in the Run object.

## Q9

True means that the text in the Run object will be bold.

False means that the text in the Run object will not be bold.

None means that the text in the Run object will inherit the bold setting from the style of the paragraph.

## Q10

To create a Document object for a new Word document using the python, we can simply call the **Document()** constructor. **Document()** creates a new Document object, which represents a new, empty Word document.

## Q11

To add a paragraph with the text 'Hello, there!' to a Document object stored in a variable named doc, we can use the **add\_paragraph()** method. **add\_paragraph()** is called on the Document object to add a new paragraph with the specified text. The method returns a Paragraph object, which can be stored in a variable if needed.

## Q12

In Word documents, the levels of headings available are represented by integers from 0 to 4. The integer 0 represents the Title style, which is used for the top of the document. Integers 1 to 4 are for various heading levels, with 1 being the main heading and 4 the lowest subheading. The level of a heading can be set using the **add\_heading()** method of the Document object. The **add\_heading()** method takes two arguments: the text of the heading and the level of the heading.

Assignment 13:

## Q1

Excel spreadsheets offer several advantages over CSV (Comma-Separated Values) spreadsheets. One of the key advantages is the ability to format data. Excel provides a wide range of formatting options such as cell borders, colours, fonts, and alignment, allowing for visually appealing and easy-to-read presentations of data. Another advantage of Excel is its support for formulas and functions. Excel offers a vast library of built-in formulas and functions that enable complex calculations, data analysis, and automation within the spreadsheet. Excel provides a range of data analysis tools such as sorting, filtering, pivot tables, and conditional formatting.

## Q2

To create a reader object using the **csv.reader()** function, we pass a file object as the argument. This can be achieved by opening a CSV file using the open() function and passing it to csv.reader().

To create a writer object using the **csv.writer()** function, we pass a file object and optionally specify the delimiter and quotechar parameters.

## Q3

For reader objects in the csv module, the File object needs to be opened in 'r' mode. This allows the reader to read data from the CSV file.

For writer objects in the csv module, the File object needs to be opened in 'w' mode. This enables the writer to write data to the CSV file.

## Q4

The **writerow()** method in the **csv.write**r object is used to write a list of values as a row to a CSV file. This method takes a single argument, which is a list containing the values to be written as a row in the CSV file.

## Q5

The keyword argument delimiter is used to specify the character or string that separates the fields in a CSV file. By default, it is set to a comma (,)

The keyword argument lineterminator is used to specify the character or string that represents the line terminator in the CSV file. It indicates the end of a row.

## Q6

The json.loads() function takes a string of JSON data and returns a Python data structure. The json.loads() function is a built-in function in Python that can be used to parse JSON data.

The json.loads() function takes a string of JSON data as an argument and returns a Python data structure. The Python data structure can be a dictionary, a list, or a nested structure of dictionaries and lists.

## Q7

The json.dumps() function takes a Python data structure and returns a string of JSON data. The json.dumps() function is a built-in function in Python that can be used to serialize Python data structures into JSON format.

The json.dumps() function takes a Python data structure as an argument and returns a string of JSON data.

Assignment 14:

## Q1

RGBA stands for Red Green Blue Alpha. It is a colour model that represents colours using combinations of red, green, and blue primary colours, along with an alpha channel that represents the transparency or opacity of the colour.

In RGBA, each component (red, green, blue, and alpha) is represented by an integer value ranging from 0 to 255. The alpha channel determines the degree of transparency, where 0 represents fully transparent and 255 represents fully opaque.

## Q2

Open the image file by using the **Image.open()** function.

Convert it to RGBA mode using the **Image.convert()** method.

Once the image is in RGBA mode, we can access the pixel data using the Image.getdata() method. This method returns a sequence of pixel values.

To retrieve the RGBA values of each pixel, we can iterate over the pixel data using a loop.

## Q3

A box tuple refers to a tuple that defines a rectangular region or bounding box within an image. It is commonly represented as a tuple of four integers (left, upper, right, lower).

The left value represents the x-coordinate of the leftmost pixel in the box, while the upper value represents the y-coordinate of the topmost pixel. The right value represents the x-coordinate of the rightmost pixel, and the lower value represents the y-coordinate of the bottommost pixel.

The box tuple is used to specify a region of interest within an image, such as cropping or extracting a specific area.

## Q4

To find out the width and height of an Image object using the Pillow library in Python, we can use the size attribute of the Image object.

from PIL import Image

# Open the image file

image = Image.open('image.jpg')

# Get the width and height of the image

width, height = image.size

# Print the width and height

print(f"Width: {width}")

print(f"Height: {height}")

## Q5

To get an Image object for a 100x100 image, excluding the lower-left quarter of it, we can use the crop() method of the Image object from the Pillow library.

from PIL import Image

# Open the original image

image = Image.open('image.jpg')

# Define the coordinates for the upper-right corner of the desired region

upper\_right = (100, 100)

# Crop the image to exclude the lower-left quarter

cropped\_image = image.crop((0, 0, upper\_right[0], upper\_right[1]))

# Display the cropped image

cropped\_image.show()

## Q6

After making changes to an Image object using the Pillow library, we can save it as an image file using the **save()** method.

image.save('modified\_image.jpg')

## Q7

The module that contains Pillow’s shape-drawing code is the ImageDraw module. The ImageDraw module provides a variety of methods for drawing shapes on images. These methods include drawing lines, rectangles, circles, ellipses, polygons, and text.

## Q8

Image objects do not have drawing methods because they are immutable. This means that they cannot be changed once they have been created. If we want to draw on an image, we need to create a Draw object.

A Draw object is a mutable object that can be used to draw on images. we can create a Draw object by calling the **draw()** method on an Image object.

Assignment 15:

## Continued in Python\_Basics.ipynb notebook.