BASIC_ASSIGNMENT_6

1. What are escape characters, and how do you use them?

Escape characters are special characters used in strings to represent characters that cannot be represented directly. They are used to escape from the normal interpretation of the character, and represent a special character or sequence of characters. Escape characters are typically represented by a backslash () followed by a letter or a combination of letters. The backslash tells the interpreter to treat the following character or characters differently than they would normally be treated.

2. What do the escape characters n and t stand for?

The escape character ‘n’ stands for new line and ‘t’ stands for tab space.

```
# Using escape characters in Python

print("Hello\nWorld")

print("This is a tab:\t|")
```

3. What is the way to include backslash characters in a string?

To include a backslash character in a string, you can use an escape character \ before the backslash. This is because the backslash is itself an escape character in Python, so you need to escape it to use it as a regular character.

For example, if you want to include a file path that contains backslashes in a string, you can use the following code:

```
# Using backslashes in a string

path = "C:\\Users\\Username\\Documents\\file.txt"

print(path)
```

In the example above, the backslashes in the file path are escaped with another backslash, so they are treated as regular characters in the string. Note that if you don't escape the backslashes, you will get a syntax error, because Python will interpret them as escape characters.

4. The string "Howl's Moving Castle" is a correct value. Why isn't the single quote character in the word Howl's not escaped a problem?

In Python, you can use single quotes or double quotes to define a string literal. If you use single quotes to define a string, you can include double quotes inside the string without escaping them, and if you use double quotes to define a string, you can include single quotes

inside the string without escaping them. This is because Python treats the other type of quote as a regular character inside the string.

So, in the string "Howl's Moving Castle", the single quote in the word "Howl's" is not a problem because the string is defined using double quotes. Python treats the single quote as a regular character inside the string, and doesn't interpret it as the end of the string.

Here's an example that shows how you can include single quotes inside a string defined using double quotes:

```
# Using single quotes inside a string

string_with_single_quotes = "I'm a string with 'single quotes' inside."

print(string_with_single_quotes)  # Output: I'm a string with 'single quotes' inside.
```

5. How do you write a string of newlines if you don't want to use the n character?

If you don't want to use the '\n' character to represent a new line in a string, you can use triple quotes to define a multi-line string, and press the Enter key to create new lines in the string. In a multi-line string, any whitespace, including newlines, that you include inside the triple quotes will be included in the string.

Here's an example that shows how to define a multi-line string in Python:

```
# Defining a multi-line string

multi_line_string = """This is a string

that spans multiple lines.

It doesn't use the \n character to create newlines."""

print(multi_line_string)
```

6. What are the values of the given expressions?

'Hello, world!'[1] = 'e'

'Hello, world!'[0:5] = 'Hello'

'Hello, world!'[:5] = 'Hello'

'Hello, world!'[3:] = 'lo, world'

7. What are the values of the following expressions?

'Hello'.upper() = “HELLO”

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

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

8. What are the values of the following expressions?

'Remember, remember, the fifth of July.'.split() = [‘Remember, ’, ‘remember, ‘, ‘the’, ‘fifth’, ‘of’, ‘July’]

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

9. What are the methods for right-justifying, left-justifying, and centering a string?

In Python, you can use the following string methods to adjust the justification of a string:

ljust(width, fillchar) - This method left-justifies a string within a field of a specified width. The method takes two arguments: width, which is the width of the field, and fillchar, which is the character used to fill any remaining space in the field. If fillchar is not specified, it defaults to a space character. Example: "hello".ljust(10, '-') returns 'hello-----'.

rjust(width, fillchar) - This method right-justifies a string within a field of a specified width. The method takes two arguments: width, which is the width of the field, and fillchar, which is the character used to fill any remaining space in the field. If fillchar is not specified, it defaults to a space character. Example: "hello".rjust(10, '-') returns '-----hello'.

center(width, fillchar) - This method centers a string within a field of a specified width. The method takes two arguments: width, which is the width of the field, and fillchar, which is the character used to fill any remaining space in the field. If fillchar is not specified, it defaults to a space character. Example: "hello".center(10, '-') returns '--hello---'.

All of these methods return a new string that has been justified to the specified width with the specified fill character. The original string is not modified.

10. What is the best way to remove whitespace characters from the start or end?

In Python, you can use the strip() method to remove whitespace characters from the start and end of a string. The strip() method returns a new string with all leading and trailing whitespace characters removed. The original string is not modified.

```
s = "  hello  "

new_s = s.strip()

print(new_s)  # prints "hello"
```