1. Why does this expression cause an error? How can you fix it? 'I have eaten ' + 99 + ' burritos.'

To fix this error, you should convert the integer to a string using the `str()` function before concatenating it with other strings. Here's the corrected code
'I have eaten ' + str(99) + ' burritos.'

2. What can you press if your program is stuck in an infinite loop

Press ctrl+c

3. What statement creates a function?

In Python, the `def` statement is used to create a function.

4. What happens to variables in a local scope when the function call returns?

When a function call returns in most programming languages, including Python, the local variables defined within the function's scope cease to exist. This means that the variables created within the function are no longer accessible or usable outside of that function.

5. What does spam[int('3' * 2) / 11] evaluate to?

The expression `int('3' * 2) / 11` first calculates `int('3' * 2)` which is equivalent to `int('33')`, resulting in the integer value 33. Then, it divides 33 by 11, resulting in a floating-point value.

6. What does spam[-1] evaluate to?

The expression `spam[-1]` evaluates to the last element of the sequence (e.g., a list, string, or tuple) referenced by the variable `spam`. In Python, negative indices are used to count elements from the end of the sequence. `-1` refers to the last element, `-2` would refer to the second-to-last element

7. What does spam[:2] evaluate to?

`2` would refer to the second element of the sequence

8. How can you trim whitespace characters from the beginning or end of a string?

You can trim whitespace characters (such as spaces, tabs, and newlines) from the beginning or end of a string in Python using the `strip()` method. The `strip()` method removes leading and trailing whitespace characters from a string and returns a new string with the whitespace removed. Here's how you can use it:

9. What is the function that creates Regex objects?

In Python, the `re.compile()` function is used to create regular expression (regex) objects. Regular expressions are a powerful tool for pattern matching and text manipulation. The `re.compile()` function compiles a regular expression pattern into a regex object, which you can then use for matching and searching operations on strings.

10. What does the search() method return?

The `search()` method in Python's `re` (regular expression) module returns a special match object if a match is found in the searched string, and `None` if no match is found. The match object provides information about the match, including the matched text and the position where the match was found.

11. How do you get the actual strings that match the pattern from a Match object?

You can get the actual strings that match the pattern from a Match object in Python by using the `group()` method of the Match object. The `group()` method returns the matched substring as a string.

12. In the regex created from r'(\d\d\d)-(\d\d\d-\d\d\d\d)', what does group 0 cover? Group 1? Group 2

In the regular expression pattern `r'(\d\d\d)-(\d\d\d-\d\d\d\d)'`, which contains two pairs of parentheses for capture groups, here's what each group represents:

1. Group 0: Group 0 covers the entire matched string. In this case, it will be the entire string that matches the pattern `(\d\d\d)-(\d\d\d-\d\d\d\d)`. Group 0 represents the full match.

2. Group 1: Group 1 corresponds to the first set of parentheses `(\d\d\d)`. It captures and represents the first three digits in the string.

13. What is the difference between the + and * characters in regular expressions?

In regular expressions, the `+` and `*` characters are quantifiers that specify how many times a particular element should be repeated in the pattern. Here's the difference between them:

1. `+` (Plus Quantifier):

- The `+` quantifier means "one or more" of the preceding element.

- It indicates that the element must appear at least once, but it can also appear more than once.

- For example, the pattern `\d+` matches one or more consecutive digits, so it would match "123," "4567," and so on.

2. `*` (Asterisk Quantifier):

- The `*` quantifier means "zero or more" of the preceding element.

- It indicates that the element is optional and can appear zero times or more.

- For example, the pattern `\w*` matches zero or more word characters, so it would match an empty string (""), "hello," or "world."

14. What is the difference between {3} and {3,5} in regular expressions?

In regular expressions, `{n}` and `{m, n}` are quantifiers that specify how many times a preceding element should be repeated. Here's the difference between `{n}` and `{m, n}`:

1. `{n}`:

- `{n}` specifies that the preceding element must be repeated exactly `n` times.

- It matches when the element appears exactly `n` times and fails to match if there are more or fewer occurrences.

- For example, the pattern `\d{3}` matches exactly three consecutive digits, like "123" or "456," but not "12" or "1234."

2. `{m, n}`:

- `{m, n}` specifies a range for the number of times the preceding element can be repeated, allowing it to appear between `m` and `n` times.

- It matches when the element appears between `m` and `n` times (inclusive).

15. How do you make a regular expression case-insensitive?

In regular expressions, you can make a pattern case-insensitive by using the `re.IGNORECASE` or `re.I` flag when compiling the regular expression pattern. This flag tells the regular expression engine to ignore the distinction between uppercase and lowercase characters when matching**.**

16. What is the difference between .* and .*??

In regular expressions, `.*` and `.*?` both use the asterisk (`*`) quantifier, but they have different meanings and behaviors:

1. `.*` (Greedy Match):

- `.*` is a greedy match that tries to match as much text as possible while still allowing the rest of the pattern to match.

- It matches any character (except a newline) zero or more times until it cannot match any further or until the rest of the pattern requires a match.

- It will consume as much text as it can before backtracking, which can lead to a match that spans a larger portion of the input text.

Example: In the pattern `a.*b`, when applied to the input "axxxbyyyb," it matches the entire string "axxxbyyyb" because it greedily matches everything between 'a' and the last 'b'.

2. `.*?` (Non-Greedy or Lazy Match):

- `.*?` is a non-greedy or lazy match that matches as little text as necessary to allow the rest of the pattern to match.

- It matches any character (except a newline) zero or more times but tries to consume as little text as possible before allowing the rest of the pattern to match.

- It will stop matching as soon as the rest of the pattern can successfully match.

Example: In the pattern `a.*?b`, when applied to the input "axxxbyyyb," it matches only "axxxb" because it matches as little as needed to allow the 'b' to match.

17. What do the os.getcwd() and os.chdir() functions do? In Python, the `os.getcwd()` and

1. `os.getcwd()`:

`os.getcwd()` is a function that returns the current working directory as a string. The current working directory is the directory (folder) in which your Python script or application is currently running.

- It does not take any arguments and simply returns the current directory path.
- Example:

```
import os
current_directory = os.getcwd()
print("Current working directory:", current_directory)
```

2. `os.chdir(path)`:

- `os.chdir()` is a function used to change the current working directory to the specified path. It takes a single argument, `path`, which is the new directory to which you want to switch.

- After calling `os.chdir(path)`, your Python script or application will be executed in the specified directory.

- Example:

```
import os

new_directory = '/path/to/new/directory'

os.chdir(new_directory)
```

18. What is the difference between the read() and readlines() methods?

1. `read()`:

- The `read()` method is used to read the entire contents of a file as a single string.

- It reads the entire file into memory, so if the file is large, it may consume a significant amount of memory.

- It does not preserve the line structure of the file; it returns the entire content as a single string.

- You can specify an optional argument to limit the number of characters read, e.g., `read(100)` reads the first 100 characters of the file.

- Example

```
with open('myfile.txt', 'r') as file:
    content = file.read
(
print(content)
```

2. `readlines()`:

- The `readlines()` method is used to read the lines of a file and return them as a list of strings.

- It reads the file line by line and stores each line as a separate string element in a list.

- It's useful when you want to work with the individual lines of a file.

- Example

```
with open('myfile.txt', 'r') as file:
    lines = file.readlines()
    for line in lines:
        print(line)
```

19. What is an RGBA value?

In Python, RGBA values are typically represented as tuples or lists of four integer values in the range of **0 to 255**, where each value corresponds to the Red, Green, Blue, and Alpha components, respectively. The format is `(R, G, B, A)` or `[R, G, B, A].

20. How can you get the RGBA value of 'CornflowerBlue' from the Pillow module?

```
from PIL import ImageColor

color_name = 'CornflowerBlue'

rgba_value = ImageColor.getcolor(color_name, 'RGBA')

print(f"The RGBA value of '{color_name}' is {rgba_value}.")
```

21. What is a box tuple?

In the context of the Python Imaging Library (PIL) and its successor, the Pillow library, **a "box tuple" refers to a tuple that specifies a rectangular region within an image. A box tuple is used to define a bounding box,** which can be used for various image manipulation operations, such as cropping or pasting a portion of an image.

22. What function returns an Image object for, say, an image file named zophie.png?

To open an image file and obtain an `Image` object using the Pillow (PIL) library in Python, you can use **the `Image.open()` function**. You provide the filename as an argument to this function, and it returns an `Image` object that represents the image. For example, to open an image file named "zophie.png":

23. How can you find out the width and height of an Image object's image?

To find out the width and height of an `Image` object's image in the Python Pillow (PIL) library, you can use the `.size` attribute of the **`Image` object.** The **`.size`** attribute returns a tuple containing the width and height of the image in pixels. Here's how you can use it:

24. What method would you call to get Image object for a 100×100 image, excluding the lower left quarter of it?

To obtain an `Image` object for a 100x100 image that excludes the lower-left quarter, you can use the **`crop()`** method from the Python Pillow (PIL) library. Here's how you can do it:

25. After making changes to an Image object, how could you save it as an image file?

After making changes to an `Image` object in the Python Pillow (PIL) library, you can save it as an image file using the **`save()` method**. The `save()` method allows you to specify the filename and the format of the image. Here's how you can save an `Image` object as an image file: