1. Is the Python Standard Library included with PyInputPlus?

ANS : - No, the Python Standard Library is not included with PyInputPlus. PyInputPlus is a separate third-party library that provides additional functionalities for taking user input in Python. It is not part of the Python Standard Library, which consists of modules and packages that are included with the Python programming language by default. PyInputPlus needs to be installed separately using tools like pip.

2. Why is PyInputPlus commonly imported with import pyinputplus as pypi?

ANS : - PyInputPlus is commonly imported with the alias "pypi" for brevity and to avoid naming conflicts.

3. How do you distinguish between inputInt() and inputFloat()?

ANS : - The `inputInt()` function is used to specifically accept integer input from the user, while the `inputFloat()` function is used to specifically accept floating-point (decimal) input from the user. The `inputInt()` function validates and returns an integer value, while the `inputFloat()` function validates and returns a floating-point value.

4. Using PyInputPlus, how do you ensure that the user enters a whole number between 0 and 99?

ANS : - To ensure that the user enters a whole number between 0 and 99 using PyInputPlus, we can use the `inputInt()` function with additional arguments for the `min` and `max` constraints. Here's an example:

```python

import pyinputplus as pypi

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

print("You entered:", number)

```

The `min` argument specifies the minimum allowed value (0 in this case), and the `max` argument specifies the maximum allowed value (99 in this case). PyInputPlus will validate the input and reprompt the user if an invalid value is entered.

5. What is transferred to the keyword arguments allowRegexes and blockRegexes?

ANS : - The `allowRegexes` and `blockRegexes` keyword arguments in PyInputPlus are used to transfer regular expressions for pattern matching.

The `allowRegexes` argument accepts a list of regular expressions as patterns that are allowed or accepted as valid input. If the user's input matches any of the regular expressions in `allowRegexes`, it is considered valid.

On the other hand, the `blockRegexes` argument accepts a list of regular expressions as patterns that are blocked or considered invalid input. If the user's input matches any of the regular expressions in `blockRegexes`, it is rejected as invalid input.

By providing appropriate regular expressions to these keyword arguments, we can customize the pattern matching behavior of PyInputPlus and restrict or allow specific input patterns based on our requirements.

6. If a blank input is entered three times, what does inputStr(limit=3) do?

ANS : - The `inputStr(limit=3)` function in PyInputPlus allows the user to enter a string input with a limit of three attempts. If a blank input (empty string) is entered three times consecutively, the function will raise the `ValidationException` indicating that the input limit has been reached.

In other words, if the user enters a blank input three times in a row, PyInputPlus will raise an exception rather than allowing further input. This behavior helps enforce a limit on the number of consecutive blank inputs.

7. If blank input is entered three times, what does inputStr(limit=3, default='hello') do?

ANS : - The `inputStr(limit=3, default='hello')` function in PyInputPlus behaves similarly to `inputStr(limit=3)`, but with an additional default value provided. If a blank input (empty string) is entered three times consecutively, instead of raising a `ValidationException`, the function will return the default value `'hello'` as the result.

In other words, if the user enters a blank input three times in a row, PyInputPlus will return the default value `'hello'` instead of raising an exception. This allows us to provide a fallback value when the limit of consecutive blank inputs is reached.