1. Is the Python Standard Library included with PyInputPlus?

No, the Python Standard Library is not included with PyInputPlus. PyInputPlus is a third-party library that provides enhanced input functions for user prompts and validation. It builds upon the functionality of the Python Standard Library's input() function by adding features like time limits, retries, and various input type validation functions.

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

The alias pypi is a common convention used when importing pyinputplus. It makes the code shorter and more readable when you want to use the library's functions. Instead of writing out pyinputplus.inputInt() or pyinputplus.inputChoice(), you can use pypi.inputInt() or pypi.inputChoice() if you've imported it as import pyinputplus as pypi.

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

Both inputInt() and inputFloat() are functions provided by PyInputPlus for taking integer and floating-point input from the user, respectively.

* inputInt(): This function is used to take input from the user and ensures that the input provided is a valid integer. If the user enters something that is not a valid integer (e.g., a string, a floating-point number, or any other non-integer input), PyInputPlus will keep prompting the user until a valid integer is entered.
* inputFloat(): Similarly, this function takes input from the user and ensures that the input provided is a valid floating-point number. It also handles cases where the user enters an invalid input and continues to prompt until a valid float is entered.

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

You can use the inputInt() function with additional arguments to achieve this. Specifically, you can set the min and max arguments to define the acceptable range of values. Here's an example:

import pyinputplus as pypi

user\_input = pypi.inputInt(prompt="Enter a whole number between 0 and 99: ", min=0, max=99)

print("You entered:", user\_input)

In this example, the min argument ensures that the user's input is greater than or equal to 0, and the max argument ensures that the input is less than or equal to 99. If the user enters an invalid input (outside the specified range or not an integer), PyInputPlus will keep prompting until a valid input is provided.

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

In PyInputPlus, the keyword arguments allowRegexes and blockRegexes are used to specify regular expressions that control what kind of input is allowed or blocked from the user. Regular expressions are patterns that define sets of strings. Here's what each of these arguments does:

* allowRegexes: This argument accepts a list of regular expression patterns. If any of the patterns in this list match the user's input, the input is considered valid and accepted. For example, if you pass allowRegexes=[r'\d{3}-\d{2}-\d{4}'], it would allow Social Security Number format input like "123-45-6789".
* blockRegexes: This argument also accepts a list of regular expression patterns. If any of the patterns in this list match the user's input, the input is considered invalid and rejected. For instance, passing blockRegexes=[r'[A-Z]'] would block input containing uppercase letters.

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

If a blank input (an empty string) is entered three times consecutively, and you're using inputStr(limit=3), then PyInputPlus will raise a ValidationException with an error message indicating that the limit of three attempts has been reached. The default error message is "Blank values are not allowed," but you can customize the message using the limitMessage parameter.

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

If blank input is entered three times consecutively and you're using inputStr(limit=3, default='hello'), then on the third attempt, the function will return the default value 'hello'. This is because the limit argument specifies the maximum number of attempts, and after three unsuccessful attempts (blank inputs), the function returns the provided default value. So, if the user doesn't provide any input within those three attempts, the function will return 'hello'.