1. Is the Python Standard Library included with PyInputPlus?

The Python Standard Library is not included with PyInputPlus. PyInputPlus is a separate third-party library that provides additional functionality for taking user input in Python. It is not part of the Python Standard Library.

The Python Standard Library refers to the collection of modules and packages that are included with Python itself. It provides a wide range of functionality for various tasks, including file I/O, networking, mathematics, data structures, and more. The Python Standard Library is installed by default when you install Python.

PyInputPlus, on the other hand, is an external library that you need to install separately using tools like pip before you can use it in your Python programs. It offers features like input validation, automatic retry on invalid input, timeout functionality, and other convenient input handling capabilities.

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

PyInputPlus is commonly imported with the alias pypi (or any other preferred alias) for brevity and convenience. It's not a required import style, but it can make the code shorter and more readable. Here are a few reasons why this aliasing practice is commonly used:

1. Conciseness: The name "PyInputPlus" is quite long, and using a shorter alias like pypi reduces the amount of typing needed when referencing PyInputPlus functions or classes throughout the code. This can lead to more concise and readable code.
2. Avoiding Name Conflicts: By importing PyInputPlus with an alias, such as pypi, you can prevent potential name conflicts if your code already has a variable or module with a similar name. It helps avoid namespace collisions and ensures clarity in your code.
3. Convention and Familiarity: The use of aliases, especially when importing third-party libraries, is a common convention in Python programming. It allows developers to quickly identify the origin of a function or class while maintaining code readability.
4. Customization: Using an alias gives you the flexibility to choose an alias that fits your preferences or aligns with the overall style of your codebase. You can select an alias that makes sense in the context of your project or team's coding guidelines.
5. How do you distinguish between inputInt() and inputFloat()?

inputInt(prompt=None, min=None, max=None):

* inputInt() is used to prompt the user for integer input.
* It displays an optional prompt message to the user.
* It validates that the user's input is a valid integer.
* It provides additional validation options such as specifying minimum and maximum values for the input.
* If the user enters a non-integer value, an error message is displayed, and the user is prompted again until a valid integer is provided.

inputFloat(prompt=None, min=None, max=None):

* inputFloat() is used to prompt the user for floating-point input.
* It displays an optional prompt message to the user.
* It validates that the user's input is a valid floating-point number.
* It provides additional validation options such as specifying minimum and maximum values for the input.
* If the user enters a non-floating-point value, an error message is displayed, and the user is prompted again until a valid float is provided.

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

To ensure that the user enters a whole number between 0 and 99 using PyInputPlus, you can use the inputInt() function with the min and max parameters set to enforce the desired range. Here's an example:

import pyinputplus as pypi

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

print("Number:", number)

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

allowRegexes:

* The allowRegexes argument accepts a list of regular expression patterns as strings.
* These patterns define the allowed input that matches the specified regular expressions.
* If any of the patterns in allowRegexes match the user's input, it is considered valid

blockRegexes:

* The blockRegexes argument also accepts a list of regular expression patterns as strings.
* These patterns define the blocked input that should be rejected based on the specified regular expressions.
* If any of the patterns in blockRegexes match the user's input, it is considered invalid and will be rejected.

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

When inputStr(limit=3) is used and a blank input is entered three times consecutively, it raises a TimeoutException.

The limit parameter in inputStr() specifies the number of times the function will allow a RetryLimitException to be raised. If the user fails to provide valid input within the specified number of retries, a TimeoutException is raised, indicating that the maximum number of retries has been exceeded.

In the case of inputStr(limit=3), if the user enters a blank input three times in a row, without providing any valid non-blank input, the function will raise a TimeoutException.

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

If blank input is entered three times consecutively and inputStr(limit=3, default='hello') is used, the function will return the default value 'hello' instead of raising a TimeoutException.

The default parameter in inputStr() allows you to specify a default value that will be returned if the user fails to provide valid input within the specified number of retries (limit).

In the case of inputStr(limit=3, default='hello'), if the user enters a blank input three times in a row, the function will return the default value 'hello' instead of raising a TimeoutException.