1. Is the Python Standard Library included with PyInputPlus?

PyInputPlus is not included by default with Python and needs to be installed separately using tools like pip.

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

The purpose of importing PyInputPlus with an alias is to provide a shorter and more readable name to refer to the module when using its functions and classes

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

1. inputInt(): This function is used to prompt the user for an integer input. It ensures that the user enters a valid integer value. If the user enters a non-integer value, it will prompt again until a valid integer is provided. The function returns the validated integer value.

Eg:

import pyinputplus as pypi

num = pypi.inputInt("Enter an integer: ")

print("Entered integer:", num)

1. inputFloat(): This function is used to prompt the user for a floating-point input. Similar to inputInt(), it validates that the user enters a valid float value. If the user enters a non-float value, it will prompt again until a valid float is provided. The function returns the validated float value.

Eg:

import pyinputplus as pypi

num = pypi.inputFloat("Enter a float: ")

print("Entered float:", num)

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

import pyinputplus as pypi

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

print("Entered number:", num)

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 patterns that are either allowed or blocked for user input validation.

The allowRegexes keyword argument takes a list of regular expressions patterns as its value. These patterns define the allowed input formats. If a user input matches any of the patterns in allowRegexes, it is considered valid.

On the other hand, the blockRegexes keyword argument also takes a list of regular expressions patterns as its value. These patterns define the blocked input formats. If a user input matches any of the patterns in blockRegexes, it is considered invalid.

By providing appropriate regular expressions patterns to allowRegexes and blockRegexes, you can control and restrict the input formats allowed or blocked during user input validation in PyInputPlus.

Eg:

import pyinputplus as pypi

# Allow only even numbers between 0 and 99

input\_number = pypi.inputInt(prompt='Enter an even number between 0 and 99: ',

allowRegexes=[r'^[02468]$', r'^[0-9]{2}$'],

blockRegexes=[r'^[13579]$'])

print('You entered:', input\_number)

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

If a blank input is entered three times when using inputStr(limit=3) in PyInputPlus, it will raise a pyinputplus.RetryLimitException error. This means that the user has exceeded the maximum number of allowed retries specified by the limit parameter.

The limit parameter in inputStr() defines the maximum number of retries or attempts allowed for the user to input a valid value. In this case, with limit=3, the user is given three chances to provide a non-blank input. If the user fails to provide a non-blank input within the specified number of retries, PyInputPlus will raise the RetryLimitException to indicate that the limit has been reached.

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

If a blank input is entered three times when using inputStr(limit=3, default='hello') in PyInputPlus, it will return the default value 'hello' instead of raising an error.

The limit parameter in inputStr() specifies the maximum number of retries or attempts allowed for the user to input a valid value. In this case, with limit=3, the user is given three chances to provide a non-blank input. However, if the user enters a blank input for all three attempts, PyInputPlus will return the default value 'hello' as specified by the default parameter.

Using the default parameter allows you to provide a fallback value that will be returned when the user does not provide a valid input within the specified number of retries. In this case, if the user fails to input a non-blank value within three attempts, the function will return 'hello' as the default value