1. Is the Python Standard Library included with PyInputPlus?

Ans:

No, the Python Standard Library is not included with PyInputPlus. PyInputPlus is a third-party library used for taking input from the user in a convenient and robust way, especially for validating and handling various types of input. It provides functionalities beyond what the Python Standard Library offers for input handling, such as input validation, retrying on invalid input, and handling timeouts.

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

Ans:

PyInputPlus is commonly imported with import pyinputplus as pypi for convenience and brevity. Using an alias such as pypi allows developers to refer to PyInputPlus with a shorter name in their code, reducing the amount of typing needed and making the code more concise.

For example, instead of writing pyinputplus.inputInt() to call the inputInt() function from PyInputPlus, you can simply write pypi.inputInt(), which is shorter and easier to read.

Additionally, using an alias like pypi helps prevent naming conflicts with other modules or packages that may have similar names. It provides a clear and distinct identifier for PyInputPlus within the codebase.

Overall, using an alias like pypi enhances code readability and maintainability while reducing the potential for naming conflicts.

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

Ans:

In PyInputPlus, inputInt() and inputFloat() are functions used to take input from the user and ensure that the input is of integer or floating-point type, respectively.

Here's how you can distinguish between inputInt() and inputFloat():

inputInt(): This function is used to prompt the user to enter an integer value. If the user enters a non-integer value, the function will continue to prompt until a valid integer is provided. It returns the integer value entered by the user.

import pyinputplus as pypi

# Prompt the user to enter an integer

integer\_value = pypi.inputInt(prompt='Enter an integer: ')

inputFloat(): This function is used to prompt the user to enter a floating-point value. Similar to inputInt(), if the user enters a non-floating-point value, the function will continue to prompt until a valid floating-point number is provided. It returns the floating-point value entered by the user.

import pyinputplus as pypi

# Prompt the user to enter a floating-point number

float\_value = pypi.inputFloat(prompt='Enter a floating-point number: ')

In summary, inputInt() is used to ensure that the input is an integer, while inputFloat() is used to ensure that the input is a floating-point number. They differ in the type of input they expect and return.

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, you can use the inputInt() function along with the min and max parameters to specify the allowable range of values. Additionally, you can set the limit parameter to restrict the number of attempts the user has to enter a valid value.

Here's how you can do it:

import pyinputplus as pypi

# Prompt the user to enter a whole number between 0 and 99

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

print('You entered:', number)

In this code:

inputInt() prompts the user to enter an integer.

min=0 specifies that the minimum allowed value is 0.

max=99 specifies that the maximum allowed value is 99.

If the user enters a value outside the specified range or a non-integer value, PyInputPlus will continue to prompt until a valid input is provided.

This ensures that the user enters a whole number between 0 and 99 and handles any invalid input accordingly.

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

Ans:

In PyInputPlus, the keyword arguments allowRegexes and blockRegexes are used to specify regular expressions that allow or block certain input patterns, respectively. These keyword arguments are used when calling functions like inputStr(), inputNum(), inputChoice(), etc., to customize the input validation behavior based on regular expression patterns.

Here's a brief overview of what is transferred to these keyword arguments:

allowRegexes: This keyword argument accepts a list of regular expression patterns. If provided, PyInputPlus will allow input values that match any of the specified regular expression patterns. If a user input matches any pattern in allowRegexes, it will be accepted, regardless of other validation criteria.

blockRegexes: This keyword argument also accepts a list of regular expression patterns. If provided, PyInputPlus will block input values that match any of the specified regular expression patterns. If a user input matches any pattern in blockRegexes, it will be rejected, even if it satisfies other validation criteria.

Here's an example demonstrating how to use these keyword arguments:

import pyinputplus as pypi

# Allow inputs that start with 'hello' or 'world'

input\_str = pypi.inputStr(prompt='Enter "hello" or "world": ', allowRegexes=[r'^hello', r'^world'])

# Block inputs that contain digits

input\_num = pypi.inputNum(prompt='Enter a number: ', blockRegexes=[r'\d+'])

print('Input (allowed):', input\_str)

print('Input (blocked):', input\_num)

In this example:

allowRegexes is used to allow inputs that start with either 'hello' or 'world'.

blockRegexes is used to block inputs that contain any digits.

These keyword arguments provide a flexible way to customize input validation based on specific regular expression patterns.

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

Ans:

If a blank input is entered three times while using inputStr(limit=3) in PyInputPlus, the function will raise a pyinputplus.RetryLimitException after the third attempt. This exception indicates that the user has exceeded the maximum number of retries allowed by the limit parameter.

Here's how it works:

import pyinputplus as pypi

# Prompt the user to enter a string with a limit of 3 retries

try:

input\_str = pypi.inputStr(prompt='Enter a string: ', limit=3)

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

except pypi.RetryLimitException:

print('Exceeded maximum number of retries.')

If the user enters a blank input three times in a row, the function will raise a pyinputplus.RetryLimitException, and the message "Exceeded maximum number of retries." will be printed.

This behavior ensures that the user is limited to a certain number of retries when entering input, preventing infinite loops or excessive waiting for user input.

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

Ans:

If blank input is entered three times while using inputStr(limit=3, default='hello') in PyInputPlus, the function will return the default value 'hello' after the third attempt. This behavior occurs because the default parameter specifies the value to return if the user enters blank input for the maximum number of retries allowed by the limit parameter.

Here's how it works:

import pyinputplus as pypi

# Prompt the user to enter a string with a limit of 3 retries and a default value of 'hello'

input\_str = pypi.inputStr(prompt='Enter a string: ', limit=3, default='hello')

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

If the user enters a blank input three times in a row, the function will return the default value 'hello', and the message "You entered: hello" will be printed.

This behavior ensures that the function always returns a valid input value, either provided by the user or the default value, even if the user exceeds the maximum number of retries for entering input.