**Python Assignment 8**

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 taking user input in a more controlled and validated manner compared to the built-in input() function.

The Python Standard Library is a collection of modules and packages that are included with Python itself and cover a wide range of functionality, including file I/O, networking, regular expressions, data structures, and more. It comes bundled with every standard Python installation and does not need to be separately installed.

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

While PyInputPlus can be imported using the standard import pyinputplus syntax, some developers choose to use the import pyinputplus as pypi syntax for brevity and convenience. This practice is not standard and depends on individual preferences or coding conventions within a specific project or team. Let's break down the reasoning behind using this alias:

Brevity: Using import pyinputplus as pypi allows us to refer to the library using a shorter alias, pypi, throughout our code. This can be especially helpful if we use PyInputPlus functions frequently, as it reduces the amount of typing required.

Readability: The alias pypi might be chosen as a mnemonic for PyInputPlus, making it easier to identify the library being used. This can improve the readability of our code, especially if we have multiple libraries with long names being imported.

Avoiding Name Conflicts: Sometimes, libraries or modules might have similar names to built-in or other third-party modules. Using an alias helps avoid potential naming conflicts. For example, import pyinputplus could potentially conflict with another library or module named pyinputplus.

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

Both inputInt() and inputFloat() are functions provided by the PyInputPlus library for taking user input in Python. The key distinction between them lies in the type of input they expect and the type of value they return:

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

This function is used to take user input and ensures that the input is a valid integer.

It displays the specified prompt to the user and waits for the user to input a value.

If the input provided by the user is not a valid integer (e.g., contains non-numeric characters), or if it falls outside the specified min and max range (if provided), the function will prompt the user to try again until valid input is provided.

The function returns the valid integer input provided by the user.

Example:

import pyinputplus as pypi

age = pypi.inputInt(prompt="Enter your age: ", min=0, max=150)

print(f"Your age is: {age}")

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

This function is used to take user input and ensures that the input is a valid floating-point number.

Like inputInt(), it displays the specified prompt to the user and waits for the user to input a value.

If the input provided by the user is not a valid floating-point number (e.g., contains non-numeric characters or multiple decimal points), or if it falls outside the specified min and max range (if provided), the function will prompt the user to try again until valid input is provided.

The function returns the valid floating-point number input provided by the user.

Example:

import pyinputplus as pypi

weight = pypi.inputFloat(prompt="Enter your weight (kg): ", min=0, max=500)

print(f"Your weight is: {weight} kg")

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

we can use the inputInt() function from the PyInputPlus library to ensure that the user enters a whole number between 0 and 99. Here's how we can achieve this:

import pyinputplus as pypi

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

print(f"You entered: {number}")

In this example, the inputInt() function is used with the min and max parameters to specify the allowable range for the input. The min parameter is set to 0 to ensure the input is greater than or equal to zero, and the max parameter is set to 99 to ensure the input is less than or equal to 99.

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

In the PyInputPlus library, the keyword arguments allowRegexes and blockRegexes are used to pass regular expressions (regexes) that define patterns for allowing or blocking specific types of input. These keyword arguments allow us to customize and control the types of input that are accepted or rejected by the PyInputPlus functions.

Here's a brief explanation of each:

allowRegexes:

The allowRegexes keyword argument accepts a list of regular expressions.

When provided, these regular expressions define patterns for input that are allowed.

If the user's input matches any of the provided regular expressions in allowRegexes, it is considered valid.

This can be useful when we want to restrict input to specific formats or patterns that we define.

blockRegexes:

The blockRegexes keyword argument accepts a list of regular expressions.

When provided, these regular expressions define patterns for input that are not allowed.

If the user's input matches any of the provided regular expressions in blockRegexes, it is considered invalid.

This can be useful when we want to prevent certain patterns or formats from being accepted as input.

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

The inputStr(limit=3) function from the PyInputPlus library allows the user to input a string, but with a limit on the number of attempts. If a blank input is entered three times consecutively (without providing any non-blank input), the function will raise a TimeoutException.

Here's how it works:

import pyinputplus as pypi

try:

user\_input = pypi.inputStr(prompt="Enter a non-blank string: ", limit=3)

print(f"You entered: {user\_input}")

except pypi.TimeoutException:

print("You exceeded the maximum number of attempts.")

In the above code, if the user enters a blank string three times in a row, the inputStr() function will raise a TimeoutException. This indicates that the user has reached the maximum number of attempts (specified by the limit parameter) without providing a valid non-blank input.

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

If blank input is entered three times, the inputStr(limit=3, default='hello') function from the PyInputPlus library will use the provided default value 'hello' after the third attempt, and the function will not raise an exception. The default value is used only when the user exceeds the specified limit of attempts without providing a valid input.