**ANSWERS**

**Q1.** **Is the Python Standard Library included with PyInputPlus?**

**Ans.** No, the Python Standard Library is not included with PyInputPlus. PyInputPlus is a separate library that provides additional functionality and validation for user inputs, building upon the Python Standard Library's input() function.

**Q2. Why is PyInputPlus commonly imported with import pyinputplus as pypi?**

**Ans.** The import statement import pyinputplus as pypi is commonly used to import the PyInputPlus library and assign it an alias for convenience. By using the alias pypi, developers can refer to the library using a shorter and more readable name in their code.

**Q3. How do you distinguish between inputInt() and inputFloat()?**

**Ans.** The inputInt() and inputFloat() functions in PyInputPlus are used to handle user inputs for integer and floating-point numbers, respectively. The main distinction between them lies in the type of input they accept.

* inputInt() expects the user to enter a whole number without any decimal places. If the user enters a non-integer value, an error will be raised, and PyInputPlus will continue to prompt the user until a valid integer is provided.
* inputFloat() allows the user to enter a number with decimal places. If the input is not a valid floating-point number, an error will be raised, and PyInputPlus will continue to prompt the user until a valid float is entered.

**Q4. 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 with additional keyword arguments. Here's an example:

import pyinputplus as pypi

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

print("You entered:", number)

In the above code, the min argument specifies the minimum value allowed, which is 0, and the max argument specifies the maximum value allowed, which is 99. If the user enters a number outside this range or a non-integer value, PyInputPlus will keep prompting until a valid input is provided.

**Q5. What is transferred to the keyword arguments allowRegexes and blockRegexes?**

**Ans.** The keyword arguments allowRegexes and blockRegexes in PyInputPlus are used to define regular expressions that match inputs allowed or blocked, respectively.

* allowRegexes is a list of regular expressions. If any of the regular expressions in this list matches the user's input, it is considered valid and accepted.
* blockRegexes is also a list of regular expressions. If any of the regular expressions in this list matches the user's input, it is considered invalid and rejected.

These keyword arguments provide a way to customize the validation of user inputs based on specific patterns or conditions.

**Q6. If a blank input is entered three times, what does inputStr(limit=3) do?**

**Ans.** If a blank input is entered three times when using inputStr(limit=3), PyInputPlus will raise a ValidationException with an error message indicating that the input limit has been reached. The program will not proceed beyond this point, and the user will not have another chance to enter a valid input.

**Q7. If blank input is entered three times, what does inputStr(limit=3, default='hello') do?**

**Ans.** If a blank input is entered three times when using inputStr(limit=3, default='hello'), PyInputPlus will return the default value 'hello' instead of raising a ValidationException. The program will continue executing with the default value, allowing the code to handle the absence of valid user input gracefully.