**Python Assignment 8**

**By Abhishek Sachan**

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

No, PyInputPlus is a third-party library that is not part of the Python Standard Library. However, PyInputPlus is built on top of the Python Standard Library's **input()** function, and it provides additional functionality for getting user input that the standard **input()** function does not offer.

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

PyInputPlus is commonly imported with **import pyinputplus as pypi** to make it easier to reference the library's functions and classes in the code.

The **as** keyword allows you to specify an alias or shorthand for the imported module name. In this case, **pypi** is a shortened name for **pyinputplus**, making it quicker and easier to type out in your code. It's a common convention to use shorter module names or aliases, especially for modules with long names, to keep the code more readable and concise.

By importing PyInputPlus with the **as** keyword, you can use the shortened alias **pypi** instead of typing out the full module name **pyinputplus** each time you want to use one of its functions or classes.

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

**inputInt()** and **inputFloat()** are two functions provided by PyInputPlus for getting user input as integers and floating-point numbers, respectively. The main difference between the two functions is the type of number they accept as input:

* **inputInt()** only accepts integers as input. If the user enters a non-integer value, an error message is displayed and the user is prompted to enter a valid integer.
* **inputFloat()** accepts floating-point numbers as input. If the user enters an integer, it is automatically converted to a float. If the user enters a non-numeric value, an error message is displayed and the user is prompted to enter a valid floating-point number.

**4. 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**, **max**, and **greaterThan**/**lessThan** arguments set appropriately. Here's an example:

import pyinputplus as pypi  
number = pypi.inputInt(prompt='Enter a number between 0 and 99: ',  
 min=0,  
 max=99,  
 greaterThan=False,  
 lessThan=True)

In this example:

* The **prompt** argument specifies the message to be displayed to the user when prompting for input.
* The **min** argument specifies the minimum allowed value, which is set to 0.
* The **max** argument specifies the maximum allowed value, which is set to 99.
* The **greaterThan** argument is set to **False** to disallow values greater than 99.
* The **lessThan** argument is set to **True** to disallow negative values.

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

The **allowRegexes** and **blockRegexes** keyword arguments in PyInputPlus are used to specify regular expressions that allow or block certain patterns of user input.

The **allowRegexes** argument takes a list of regular expressions as input. When PyInputPlus prompts the user for input, it checks the user's input against each regular expression in the list. If the user's input matches any of the regular expressions in the **allowRegexes** list, the input is accepted. Otherwise, PyInputPlus displays an error message and prompts the user to enter a valid input.

For example, to allow only input that contains only letters and spaces, you could use the following **allowRegexes** argument:

import pyinputplus as pypi  
  
input\_str = pypi.inputStr(prompt='Enter only letters and spaces: ',  
 allowRegexes=['^[a-zA-Z]\*$'])

In this example, the regular expression **^[a-zA-****Z ]\*$** matches only strings that contain only letters and spaces.

The **blockRegexes** argument also takes a list of regular expressions as input. When PyInputPlus prompts the user for input, it checks the user's input against each regular expression in the list. If the user's input matches any of the regular expressions in the **blockRegexes** list, the input is rejected. Otherwise, PyInputPlus accepts the input.

For example, to block input that contains any digits, you could use the following **blockRegexes** argument:

import pyinputplus as pypi  
  
input\_str = pypi.inputStr(prompt='Enter a string with no digits: ',  
 blockRegexes=['\d'])

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

The **inputStr(limit=3)** function takes a user input as a string and allows the user to enter it up to a maximum of 3 times. If the user enters a blank input three times in a row, then the function will raise a **ValueError** with the message "Exceeded maximum number of tries" and terminate the program.

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

If a blank input is entered three times in a row while calling **inputStr(limit=3, default='hello')**, the function will return the default value of "hello" instead of raising a **ValueError** as in the previous case.

However, if the user enters a non-blank input before reaching the maximum limit of 3 tries, then the function will return the input provided by the user, as it would normally do in the absence of the **default** parameter.