**Assignment 8**

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

**Ans:** No, the Python Standard Library is separate from PyInputPlus. However, PyInputPlus may use some modules from Python Standard Library to Implement certain features.

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

**Ans:** PyInputPlus is commonly imported with ‘import pyinputplus as pypi’ to provide a shorter alias for the module name. This makes it quicker and easier to type the module name when using it in code. The ‘as’ keyword is used to give the imported module a new alias, in this case ‘pypi’. It is a common convention to use a short, lowercase alias for commonly used modules to reduce typing and improve code readability.

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

**Ans:** **inputInt ()** and **inputFloat ()** are two different PyInputPlus functions used to take user input as integers and floats, respectively.

* **inputInt()** function ensures that the user enters an integer value. If the user enters an invalid
* input, it will reprompt them until they enter a valid integer.
* **inputFloat ()** function ensures that the user enters a floating-point value. It also accepts scientific notation (e.g., 3.14e7) as a valid input. If the user enters an invalid input, it will reprompt them until they enter a valid floating-point number.

You can distinguish between these functions based on the type of input you want to accept from the user. Use **inputInt ()** if you want to ensure that the user enters an integer value, and **inputFloat ()** if you want to ensure that the user enters a floating-point value.

**Q4. Using PyInputPlus, how do you ensure that the user enters a whole number between 0 and 99?**

**Ans:**   
You can use the **inputInt()** function from PyInputPlus to ensure that the user enters a whole number and set the **min** and **max** arguments to limit the range of allowed inputs. Here is an example of code that prompts the user to enter a whole number between 0 and 99:

‘’’ **import pyinputplus as pyip**

**num = pyip.inputInt(prompt='Enter a whole number between 0 and 99: ', min=0, max=99)**

**print('You entered:', num) ’’’**

In this code, the **inputInt ()** function prompts the user to enter a whole number between 0 and 99. If the user enters a number outside of this range, **inputInt()** will reprompt them until a valid number is entered.

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

**Ans:** The **allowRegexes** and **blockRegexes** keyword arguments in PyInputPlus are lists of regular expression strings.

* **allowRegexes** is a list of regular expressions that match the input that is allowed.
* **blockRegexes** is a list of regular expressions that match the input that is not allowed.

When using PyInputPlus, if a value matches a regular expression in the **allowRegexes** list and doesn't match any regular expression in the **blockRegexes** list, it is accepted as valid input.

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

**Ans:** If a blank input is entered three times, **inputStr(limit=3)** will raise a **TimeoutException** error indicating that the user has exceeded the maximum number of allowed attempts. This is because **limit=3** specifies that the user can only enter input 3 times before an error is raised. The default behaviour is to keep prompting the user indefinitely until valid input is provided.

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

**Ans:** If blank input is entered three times, **inputStr (limit=3, default='hello')** will return the default value **'hello'**.

The **limit** parameter specifies the number of attempts the user has to enter valid input before raising a **TimeoutException**. In this case, if the user enters blank input three times, the function will raise a **TimeoutException** with a message indicating that the limit has been reached.

However, since the **default** parameter is set to **'hello'**, the function will return **'hello'** instead of raising an exception. This is because the **default** parameter specifies the value to return if the user enters blank input or if the **TimeoutException** is raised.