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 that provides additional functionality for taking input from users, such as input validation and automatic retrying. It builds upon the functionality provided by the Python Standard Library's `input()` function but does not include the entire standard library. If you want to use functions or modules from the Python Standard Library, you will need to import them separately in your code.

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

Ans - The practice of importing PyInputPlus as `pypi` (or any other chosen alias) is purely a matter of personal preference or convention within the codebase or community. It allows developers to use a shorter and more concise name when referencing PyInputPlus in their code.

Using `import pyinputplus as pypi` allows you to access the PyInputPlus functions and classes by prefixing them with `pypi`. For example, instead of writing `pyinputplus.inputStr()`, you can use `pypi.inputStr()`. This can save typing effort and make the code more readable.

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

Ans - In PyInputPlus, `inputInt()` and `inputFloat()` are two separate functions used for taking user input as integers and floats, respectively. Here's how you can distinguish between them:

1. `inputInt(prompt=None, default=None, limit=None, timeout=None, greaterThan=None, lessThan=None)`:

- This function is used to prompt the user for an integer input.

- It displays an optional prompt to the user and expects an integer value as the input.

- If the user enters a non-integer value, it prompts them to enter a valid integer.

- It also provides additional optional parameters like `default`, `limit`, `timeout`, `greaterThan`, and `lessThan` for input validation and control.

2. `inputFloat(prompt=None, default=None, limit=None, timeout=None, greaterThan=None, lessThan=None)`:

- This function is used to prompt the user for a floating-point number input.

- It displays an optional prompt to the user and expects a float value as the input.

- If the user enters a non-float value, it prompts them to enter a valid float.

- It also provides optional parameters like `default`, `limit`, `timeout`, `greaterThan`, and `lessThan` for input validation and control.

To summarize, `inputInt()` is used when you expect the user to provide an integer input, while `inputFloat()` is used when you expect a floating-point number as input.

1. **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 utilize the `inputInt()` function with the `min` and `max` parameters. Here's an example:

```python

import pyinputplus as pypi

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

min=0, max=99)

print("You entered:", number)

```

In the code above:

1. We import `pyinputplus` and alias it as `pypi`.

2. The `inputInt()` function is called with the following parameters:

- `prompt`: The message displayed to the user as a prompt.

- `min`: The minimum allowed value, which is 0 in this case.

- `max`: The maximum allowed value, which is 99 in this case.

3. The user's input, which must be a whole number between 0 and 99, is stored in the `number` variable.

4. Finally, we print the entered number.

With this code, PyInputPlus will enforce that the user enters a valid whole number within the specified range. If the user enters a value outside the range or a non-integer value, PyInputPlus will continue to prompt them until a valid input is provided.

1. **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 control which input patterns are allowed or blocked, respectively. These arguments are passed to various input functions such as `inputStr()`, `inputInt()`, `inputFloat()`, etc.

Here's what is transferred to these keyword arguments:

1. `allowRegexes`:

- This argument accepts a list of regular expressions (as strings) or a compiled regular expression pattern.

- If provided, PyInputPlus will only accept user input that matches at least one of the regular expressions in the `allowRegexes` list.

- If the user's input does not match any of the specified regular expressions, PyInputPlus will continue to prompt for valid input until a matching pattern is entered.

2. `blockRegexes`:

- This argument accepts a list of regular expressions (as strings) or a compiled regular expression pattern.

- If provided, PyInputPlus will block user input that matches any of the regular expressions in the `blockRegexes` list.

- If the user's input matches any of the specified regular expressions, PyInputPlus will reject the input and continue to prompt for valid input until a non-matching input is provided.

By using these arguments, you can define custom patterns to allow or block specific input formats based on regular expressions. This allows for flexible input validation and control.

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

Ans - If a blank input is entered three times consecutively while using `inputStr(limit=3)`, the behavior depends on the specific version of PyInputPlus being used.

In versions up to and including PyInputPlus 0.3.0:

- If the user enters a blank input three times in a row, `inputStr(limit=3)` will raise a `ValidationException`. This exception indicates that the input limit has been reached, and no further input is accepted.

However, starting from PyInputPlus 0.4.0:

- If the user enters a blank input three times in a row, `inputStr(limit=3)` will return an empty string (`''`). It will not raise an exception.

The behavior change was introduced in PyInputPlus 0.4.0 to provide more flexibility and avoid raising an exception in cases where an empty input is acceptable or desired.

Therefore, it is important to consider the specific version of PyInputPlus being used when determining the behavior of `inputStr(limit=3)` in response to blank inputs.

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

Ans - If a blank input is entered three times consecutively while using `inputStr(limit=3, default='hello')`, the behavior depends on the specific version of PyInputPlus being used.

In versions up to and including PyInputPlus 0.3.0:

- If the user enters a blank input three times in a row, `inputStr(limit=3, default='hello')` will raise a `ValidationException`. This exception indicates that the input limit has been reached, and no further input is accepted. The provided default value `'hello'` will not be used in this case.

However, starting from PyInputPlus 0.4.0:

- If the user enters a blank input three times in a row, `inputStr(limit=3, default='hello')` will return the default value `'hello'` without raising an exception. This behavior change allows the default value to be used when the input limit is reached.

Therefore, it's important to consider the specific version of PyInputPlus being used when determining the behavior of `inputStr(limit=3, default='hello')` in response to blank inputs.