1. Is the Python Standard Library included with PyInputPlus?

Ans: The Python Standard Library is not included with PyInputPlus. PyInputPlus is a third-party library that provides additional features and functionality for input validation and error handling in Python. The Python Standard Library, on the other hand, is a built-in collection of modules that provide a wide range of common functions and utilities that are used in Python programs. To use the functions and modules in the Python Standard Library, we will need to import them into your Python code, just like you would with any other third-party library.

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

Ans: It is common to import PyInputPlus with import pyinputplus as pypi because it allows us to use a shorter and more convenient alias for the library. This can make your code more readable and easier to write, because you can use the pypi alias instead of typing out the full pyinputplus name every time you want to use a function or method from the library.

```
import pyinputplus as pypi

# Get a string input from the user

response = pypi.inputStr("Enter a string: ")

#Using an alias like pypi can make your code more concise and easier to read.
```

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

Ans: We can distinguish between inputInt() and inputFloat() by the type of input they accept and the type of value they return.

inputInt() is a function that prompts the user to enter an integer value. It will only accept input that is a valid integer, and it will return the input as an int type. For example:

```
import pyinputplus as pypi

# Prompt the user to enter an integer

i = pypi.inputInt("Enter an integer: ")

# The value of i will be an int type

print(type(i))  # Output: <class 'int'>
```

inputFloat() is a function that prompts the user to enter a floating-point number. It will accept input that is a valid floating-point number, and it will return the input as a float type. For example:

```
import pyinputplus as pypi

# Prompt the user to enter a floating-point number

f = pypi.inputFloat("Enter a floating-point number: ")
```

```
# The value of f will be a float type

print(type(f))  # Output: <class 'float'>
```

We can use inputInt() to get integer input from the user, and inputFloat() to get floating-point input from the user. The main difference between the two functions is the type of input they accept and the type of value they return.

4. 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, we can use the inputInt() function with the limit and blank keyword arguments. The limit argument allows you to specify the minimum and maximum values that the user is allowed to enter, and the blank argument allows you to specify whether the user is allowed to enter an empty string as their input.

Here is an example that shows how to use inputInt() to get a whole number between 0 and 99 from the user:

```
import pyinputplus as pypi

# Prompt the user to enter a number between 0 and 99

num = pypi.inputInt("Enter a number between 0 and 99: ", limit=(0, 99), blank=False)

# The value of num will be an int type and will be between 0 and 99

print(type(num))  # Output: <class 'int'>

print(num)  # Output: (a number between 0 and 99)
```

In this example, the inputInt() function will only accept input that is a whole number between 0 and 99. If the user enters a value that is not within this range, they will be prompted to enter a valid value. The blank=False argument ensures that the user is not allowed to enter an empty string as their input.

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

Ans: The allowRegexes and blockRegexes keyword arguments are used to specify regular expressions that the user's input must or must not match, respectively. These arguments are used with the inputStr() function in PyInputPlus to validate the user's input and ensure that it meets certain criteria.

The allowRegexes argument is used to specify one or more regular expressions that the user's input must match in order to be considered valid. For example, if you want to ensure that the user enters a string that is a valid email address, you could use allowRegexes to specify a regular expression that matches email addresses.

The blockRegexes argument is used to specify one or more regular expressions that the user's input must not match in order to be considered valid. For example, if you want to ensure that the user does not enter a string that contains profanity, you could use blockRegexes to specify regular expressions that match profane words or phrases.

```
import re

import pyinputplus as pypi

# Create a regular expression that matches email addresses

emailRegex = re.compile(r"[\w.-]+@[\w.-]+\.[\w]+")

# Create a regular expression that matches profane words or phrases

profanityRegex = re.compile(r"\b(damn|hell|crap)\b", re.IGNORECASE)

# Prompt the user to enter a string

response = pypi.inputStr("Enter a string: ",

                allowRegexes=[emailRegex],

                blockRegexes=[profanityRegex])

# The value of response will be a string that is a valid email address and does not contain profanity

print(response)
```

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

Ans: If a blank input is entered three times while using the inputStr(limit=3) function from the pyinputplus module, the function will return an error because the maximum number of blank inputs allowed has been reached. This behavior is determined by the limit parameter, which sets the maximum number of blank inputs that are allowed before the function returns an error.

```
import pyinputplus as pyip

# Prompt the user for a string of input, and allow them to enter

# a blank input up to three times before returning an error.

response = pyip.inputStr(prompt='Enter a string of text:', limit=3)

print(response)
```

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

Ans: If a blank input is entered three times while using the inputStr(limit=3, default='hello') function from the pyinputplus module, the function will return the default value specified by the default parameter, which in this case is 'hello'. This is because the limit parameter is used to set the maximum number of times the user can enter a blank input before the function returns the default value specified by the default parameter. So, if the limit parameter is set to 3, then the user will be allowed to enter a blank input up to three times before the function will return the default value of 'hello'.

```
import pyinputplus as pyip

# Prompt the user for a string of input, and allow them to enter

# a blank input up to three times before returning the default

# value of 'hello'.

response = pyip.inputStr(prompt='Enter a string of text:', limit=3, default='hello')

print(response)
```