Assignment_8

April 16, 2023

1. Is the Python Standard Library included with PyInputPlus? No, the Python Standard Library is not included with PyInputPlus. PyInputPlus is a third-party library that provides additional input validation and handling features beyond what is available in the Python Standard Library.

However, PyInputPlus may use some modules from the Python Standard Library, such as the re module for regular expressions, to implement its features. 2. Why is PyInputPlus commonly imported with import pyinputplus as pypi? PyInputPlus is commonly imported with the alias pypi using the following syntax:

```
import pyinputplus as pypi
```

This alias is commonly used because the name "PyInputPlus" is quite long, and using pypi as an alias is shorter and more convenient. The alias also makes the code more readable and easier to understand.

By using the alias, we can access the PyInputPlus functions and methods using the pypi prefix. 3. How do you distinguish between inputInt() and inputFloat()? The inputInt() and inputFloat() functions in PyInputPlus are used to get integer and floating-point number inputs from the user, respectively. Here's how you can distinguish between them:

The first argument: inputInt() and inputFloat() have different first arguments. inputInt() requires a prompt string that is displayed to the user, while inputFloat() requires a prompt string and can also take optional arguments for the minimum and maximum values allowed.

The returned value: inputInt() always returns an integer, while inputFloat() always returns a floating-point number.

[3]:
```
import pyinputplus as pypi

# inputInt() example
age = pypi.inputInt("Enter your age: ")
print("Your age is:", age)
print("Type of age is:", type(age))

# inputFloat() example
price = pypi.inputFloat("Enter the price: ", min=0, max=1000)
print("The price is:", price)
print("Type of price is:", type(price))
```

```
Enter your age:

 3
```

```
Your age is: 3
Type of age is: <class 'int'>
Enter the price:

 34

The price is: 34.0
Type of price is: <class 'float'>
```

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 and max arguments.

[4]:
```python
import pyinputplus as pypi

# Get a whole number between 0 and 99
num = pypi.inputInt("Enter a number between 0 and 99: ", min=0, max=99)

# Display the input
print("You entered:", num)
```

```
Enter a number between 0 and 99:

 5

You entered: 5
```

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 are either allowed or blocked as input from the user.

The allowRegexes argument takes a list of regular expressions, and only inputs that match one of the regular expressions in the list will be accepted.

[6]:
```python
import pyinputplus as pypi

# Allow inputs that consist of digits and spaces
num = pypi.inputStr("Enter a number: ", allowRegexes=[r'^\d+\s*\d*$'])

# Display the input
print("You entered:", num)

"""In this example, we use the allowRegexes argument to specify a regular␣
 ↪expression that matches one or more digits, optionally followed by spaces,␣
 ↪followed by zero or more digits. This ensures that the input only consists␣
 ↪of digits and spaces."""
```

```
Enter a number:

 5

You entered: 5
```

[6]:
```
'In this example, we use the allowRegexes argument to specify a regular
expression that matches one or more digits, optionally followed by spaces,
followed by zero or more digits. This ensures that the input only consists of
digits and spaces.'
```

[8]:
```
import pyinputplus as pypi

# Block inputs that contain any letter characters
num = pypi.inputStr("Enter a number: ", blockRegexes=[r'[a-zA-Z]'])

# Display the input
print("You entered:", num)
"""In this example, we use the blockRegexes argument to specify a regular␣
 ↪expression that matches any letter characters. This ensures that the input␣
 ↪does not contain any letter characters. If the user enters a value that␣
 ↪contains a letter character, inputStr() will raise a BlockedInputException␣
 ↪and prompt the user again until a valid input is entered."""
```

```
Enter a number:
 43
You entered: 43
```

[8]:
```
'In this example, we use the blockRegexes argument to specify a regular
expression that matches any letter characters. This ensures that the input does
not contain any letter characters. If the user enters a value that contains a
letter character, inputStr() will raise a BlockedInputException and prompt the
user again until a valid input is entered.'
```

6. If a blank input is entered three times, what does inputStr(limit=3) do? If a blank input is entered three times in a row when using the inputStr() function with the limit argument set to 3, PyInputPlus will raise a TimeoutException.

The limit argument specifies the maximum number of attempts the user has to provide valid input before PyInputPlus raises an exception. By default, limit is set to 3, which means that the user has three attempts to provide valid input. If the user enters invalid input three times in a row, PyInputPlus will raise a ValidationException or TimeoutException, depending on the function used and the value of other arguments such as timeout.

In the case of inputStr(limit=3), if the user enters a blank input three times in a row, the function will raise a TimeoutException because no valid input was provided within the specified limit. 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 when using the inputStr() function with the limit argument set to 3 and the default argument set to 'hello', the function will return the default value 'hello'.

The default argument specifies a default value to return if the user enters an empty input, which is an input that consists only of whitespace characters. By default, default is set to None, which means that if the user enters an empty input, the function will keep prompting the user until a non-empty input is provided or the limit is reached.

In the case of inputStr(limit=3, default='hello'), if the user enters a blank input three times in a row, the function will return the default value 'hello' instead of raising a TimeoutException. If the user enters a non-empty input within the limit of three attempts, the function will return the input as a string.

[ ]: