Number 1 -

- Question -

Is the Python Standard Library included with PyInputPlus?

- Answer -

PyInputPlus is a Python module used for taking inputs with additional validation features. PyInputPlus will keep asking the user for text until they enter valid input.

Number 2 -

- Question -

Why is PyInputPlus commonly imported with import pyinputplus as pypi?

- Answer -

The 'as pypi' code in the import statement saves us from typing pyinputplus each time we want to call a PyInputPlus function.

Number 3 -

- Question -

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

- Answer -

"inputInt()" : Accepts an integer value. This also takes additional parameters ‘min’, ‘max’, ‘greaterThan’ and ‘lessThan’ for bounds. Returns an int. "inputFloat()" : Accepts a floating-point numeric value. Also takes additional ‘min’, ‘max’, ‘greaterThan’ and ‘lessThan’ parameters. Returns a float.

Number 4 -

- Question -

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

- Answer -

In [10]:

```
import pyinputplus as pypi

# integer input with
# specific bounds
inp = pypi.inputInt(prompt = "Enter an Integer... ",min = 0, max = 99 )

print(inp)
```

```
Enter an Integer... 5.5
'5.5' is not an integer.
Enter an Integer... -9
Number must be at minimum 0.
Enter an Integer... 600
Number must be at maximum 99.
Enter an Integer... 95
95
```

Number 5 -

- Question -

What is transferred to the keyword arguments allowRegexes and blockRegexes?

- Answer -

The allowRegexes and blockRegexes keyword arguments take a list of regular expression strings to determine what the PyInputPlus function will accept or reject as valid input.

Number 6 -

- Question -

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

- Answer -

It will raise RetryLimitException.

In [12]:

```
inp = pypi.inputStr(limit=3)
```

```
Blank values are not allowed.

Blank values are not allowed.

Blank values are not allowed.
```

```
ValidationException                         Traceback (most recent call last)
```

```
D:\Anaconda\lib\site-packages\pyinputplus\__init__.py in _genericInput(prompt
, default, timeout, limit, applyFunc, validationFunc, postValidateApplyFunc,
passwordMask)
    166         try:
--> 167             possibleNewUserInput = validationFunc(
    168                 userInput

D:\Anaconda\lib\site-packages\pyinputplus\__init__.py in <lambda>(value)
    242
--> 243     validationFunc = lambda value: pysv._prevalidationCheck(
    244         value, blank=blank, strip=strip, allowRegexes=allowRegexes, b
lockRegexes=blockRegexes, excMsg=None,

D:\Anaconda\lib\site-packages\pysimplevalidate\__init__.py in _prevalidationC
heck(value, blank, strip, allowRegexes, blockRegexes, excMsg)
    249         # value is blank but blanks aren't allowed.
--> 250         _raiseValidationException(_("Blank values are not allowed."),
excMsg)
    251     elif blank and value == "":

D:\Anaconda\lib\site-packages\pysimplevalidate\__init__.py in _raiseValidatio
nException(standardExcMsg, customExcMsg)
    221     if customExcMsg is None:
--> 222         raise ValidationException(str(standardExcMsg))
    223     else:

ValidationException: Blank values are not allowed.

During handling of the above exception, another exception occurred:

RetryLimitException                       Traceback (most recent call last)
<ipython-input-12-9acf02496cc9> in <module>
----> 1 inp = pypi.inputStr(limit=3)
      2
      3

D:\Anaconda\lib\site-packages\pyinputplus\__init__.py in inputStr(prompt, def
ault, blank, timeout, limit, strip, allowRegexes, blockRegexes, applyFunc, po
stValidateApplyFunc)
    245     )[1]
    246
--> 247     return _genericInput(
    248         prompt=prompt,
    249         default=default,

D:\Anaconda\lib\site-packages\pyinputplus\__init__.py in _genericInput(prompt
, default, timeout, limit, applyFunc, validationFunc, postValidateApplyFunc,
passwordMask)
    186                 else:
    187                     # If there is no default, then raise the timeout/
limit exception.
--> 188                     raise limitOrTimeoutException
    189             else:
```

```
    190                 # If there was no timeout/limit exceeded, let the use
r enter input again.

RetryLimitException:
```

Number 7 -

- Question -

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

- Answer -

The function returns the default value instead of raising an exception.

In [13]:

```
inp = pypi.inputStr(limit=3, default='hello')
```

```
Blank values are not allowed.

Blank values are not allowed.

Blank values are not allowed.
```

In [14]:

```
inp
```

Out[14]:

```
'hello'
```

In [ ]: