1. Is the Python Standard Library included with PyInputPlus?

PyInputPlus is not a part of the Python Standard Library, so you must install it separately using Pip

#!pip install pyinputplus

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

pypi is alias of PyInputPlus. The as pyip code in the import statement saves us from typing pyinputplus each time we want to call a PyInputPlus function. Instead we can use the shorter pyip name#

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

inputInt() : Accepts an integer value, and returns int value

inputFloat() : Accepts integer/floating point value and returns float value

Both takes additional parameters 'min', 'max', 'greaterThan' and 'lessThan' for bounds

In [2]:
```
pip install pyinputplus
```

```
Requirement already satisfied: pyinputplus in c:\users\upasa\anaconda3\lib\site-packages (0.2.12)
Requirement already satisfied: stdiomask>=0.0.3 in c:\users\upasa\anaconda3\lib\site-packages (from pyinputplu
s) (0.0.6)
Requirement already satisfied: pysimplevalidate>=0.2.7 in c:\users\upasa\anaconda3\lib\site-packages (from pyi
nputplus) (0.2.12)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
import pyinputplus as pyip
pyip.inputInt()
pyip.inputFloat()
```

```
4
3.5
```

Out[5]:
```
3.5
```

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

In the inputint function we can set the min = 0 and max =99 to ensure user enters number between 0 and 99

In [6]:
```
pyip.inputInt(min = 0, max =99)
```

```
98
```

Out[6]:
```
98
```

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

we can also use regular expressions to specify whether an input is allowed or not. 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.

In [7]:
```
response = pyip.inputNum(allowRegexes=[r'(I|V|X|L|C|D|M)', r'zero'])
```

```
IV
```

In [9]:
```
response = pyip.inputNum(blockRegexes=[r'[13579]$'])
```

```
35
This response is invalid.
44
```

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

it will throw RetryLimitException exception.

In [14]:
```
1 response = pyip.inputStr(limit=3)
```

```
Blank values are not allowed.

Blank values are not allowed.

Blank values are not allowed.

---------------------------------------------------------------------------
ValidationException                       Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pyinputplus\__init__.py:167, in _genericInput(prompt, default, timeout, lim
it, applyFunc, validationFunc, postValidateApplyFunc, passwordMask)
    166 try:
--> 167     possibleNewUserInput = validationFunc(
    168         userInput
    169     )  # If validation fails, this function will raise an exception. Returns an updated value to use a
s user input (e.g. stripped of whitespace, etc.)
    170     if possibleNewUserInput is not None:

File ~\anaconda3\lib\site-packages\pyinputplus\__init__.py:243, in inputStr.<locals>.<lambda>(value)
    241 pysv._validateGenericParameters(blank, strip, allowRegexes, blockRegexes)
--> 243 validationFunc = lambda value: pysv._prevalidationCheck(
    244     value, blank=blank, strip=strip, allowRegexes=allowRegexes, blockRegexes=blockRegexes, excMsg=None
,
    245 )[1]
    247 return _genericInput(
    248     prompt=prompt,
    249     default=default,
   (...)
    254     validationFunc=validationFunc,
    255 )

File ~\anaconda3\lib\site-packages\pysimplevalidate\__init__.py:250, in _prevalidationCheck(value, blank, stri
p, allowRegexes, blockRegexes, excMsg)
    248 if not blank and value == "":
    249     # value is blank but blanks aren't allowed.
--> 250     _raiseValidationException(_("Blank values are not allowed."), excMsg)
    251 elif blank and value == "":

File ~\anaconda3\lib\site-packages\pysimplevalidate\__init__.py:222, in _raiseValidationException(standardExcM
sg, 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)
Input In [14], in <cell line: 1>()
----> 1 response = pyip.inputStr(limit=3)

File ~\anaconda3\lib\site-packages\pyinputplus\__init__.py:247, in inputStr(prompt, default, blank, timeout, l
imit, strip, allowRegexes, blockRegexes, applyFunc, postValidateApplyFunc)
    241 pysv._validateGenericParameters(blank, strip, allowRegexes, blockRegexes)
    243 validationFunc = lambda value: pysv._prevalidationCheck(
    244     value, blank=blank, strip=strip, allowRegexes=allowRegexes, blockRegexes=blockRegexes, excMsg=None
,
    245 )[1]
--> 247 return _genericInput(
    248     prompt=prompt,
    249     default=default,
    250     timeout=timeout,
    251     limit=limit,
    252     applyFunc=applyFunc,
    253     postValidateApplyFunc=postValidateApplyFunc,
    254     validationFunc=validationFunc,
    255 )

File ~\anaconda3\lib\site-packages\pyinputplus\__init__.py:188, in _genericInput(prompt, default, timeout, lim
it, applyFunc, validationFunc, postValidateApplyFunc, passwordMask)
    185         return default
    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 user enter input again.
    191     continue

RetryLimitException:
```

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

When you use limit keyword arguments and also pass a default keyword argument, the function returns the default value instead of raising an exception

In [16]:
```
response = pyip.inputStr(limit=3,default='hello')
response
```

```
Blank values are not allowed.

Blank values are not allowed.

Blank values are not allowed.
```

Out[16]:
```
'hello'
```

In [ ]:
```

```