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
```

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

```
Collecting pyinputplus
  Downloading PyInputPlus-0.2.12.tar.gz (20 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
    Preparing wheel metadata: started
    Preparing wheel metadata: finished with status 'done'
Collecting stdiomask>=0.0.3
  Downloading stdiomask-0.0.6.tar.gz (3.6 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
    Preparing wheel metadata: started
    Preparing wheel metadata: finished with status 'done'
Collecting pysimplevalidate>=0.2.7
  Downloading PySimpleValidate-0.2.12.tar.gz (22 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
    Preparing wheel metadata: started
    Preparing wheel metadata: finished with status 'done'
Building wheels for collected packages: pyinputplus, pysimplevalidate, stdiomask
  Building wheel for pyinputplus (PEP 517): started
  Building wheel for pyinputplus (PEP 517): finished with status 'done'
  Created wheel for pyinputplus: filename=PyInputPlus-0.2.12-py3-none-any.whl size=11319 sha256=70577e00285a30b943a5f5c87c908daae478eefcb265549e4d1bf1929
3a0d91c
  Stored in directory: c:\users\rosha\appdata\local\pip\cache\wheels\b4\6e\2f\8a852732646cabec36c3fe8fc060ec5bea1c1be711432c47f7
  Building wheel for pysimplevalidate (PEP 517): started
  Building wheel for pysimplevalidate (PEP 517): finished with status 'done'
  Created wheel for pysimplevalidate: filename=PySimpleValidate-0.2.12-py3-none-any.whl size=16204 sha256=1084989fe190a2d911710475e8e5b82c191f7b8f83b5db2
c5ae5b69915706406
  Stored in directory: c:\users\rosha\appdata\local\pip\cache\wheels\b1\44\4a\043a4f4c4512c7cdfb0c2b8408b18b0de5fd45cac57f5dfa02
  Building wheel for stdiomask (PEP 517): started
  Building wheel for stdiomask (PEP 517): finished with status 'done'
  Created wheel for stdiomask: filename=stdiomask-0.0.6-py3-none-any.whl size=3322 sha256=9337426ad141899d7b4ca84512647e2367fd34d32f5f188f6bc3a3cc5571d75
f
  Stored in directory: c:\users\rosha\appdata\local\pip\cache\wheels\1d\aa\47\f41f117d22c5de82e95d9342f44da578c80610739a2d5ebec4
Successfully built pyinputplus pysimplevalidate stdiomask
Installing collected packages: stdiomask, pysimplevalidate, pyinputplus
Successfully installed pyinputplus-0.2.12 pysimplevalidate-0.2.12 stdiomask-0.0.6

WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -ython (c:\users\rosha\anaconda3\lib\site-packages)
```

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
```

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
```

In [3]:
```
import pyinputplus as pyip
pyip.inputInt()
pyip.inputFloat()
```

```
7.8
'7.8' is not an integer.
7
9.5
```

Out[3]: 9.5

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 [4]:
```
pyip.inputInt(min=0, max=99)
```

```
100
Number must be at maximum 99.
5
```

Out[4]: 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 [5]:
```
response = pyip.inputNum(allowRegexes=[r'(I|V|X|L|C|D|M)+', r'zero'])
```

```
E
'E' is not a number.
X
```

In [6]:
```
response = pyip.inputNum(blockRegexes=[r'[02468]$'])
```

```
8
This response is invalid.
l
'l' is not a number.
1
```

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

```
- It will throw RetryLimitException exception.
```

In [7]:
```
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)
~\anaconda3\lib\site-packages\pyinputplus\__init__.py in _genericInput(prompt, default, timeout, limit, applyFunc, validationFunc, postValidateApplyFunc, passwordMask)
    166         try:
--> 167             possibleNewUserInput = validationFunc(
    168                 userInput

~\anaconda3\lib\site-packages\pyinputplus\__init__.py in <lambda>(value)
    242
--> 243     validationFunc = lambda value: pysv._prevalidationCheck(
    244         value, blank=blank, strip=strip, allowRegexes=allowRegexes, blockRegexes=blockRegexes, excMsg=None,

~\anaconda3\lib\site-packages\pysimplevalidate\__init__.py in _prevalidationCheck(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 == "":

~\anaconda3\lib\site-packages\pysimplevalidate\__init__.py in _raiseValidationException(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)
~\AppData\Local\Temp/ipykernel_2192/1250714940.py in <module>
----> 1 response = pyip.inputStr(limit=3)

~\anaconda3\lib\site-packages\pyinputplus\__init__.py in inputStr(prompt, default, blank, timeout, limit, strip, allowRegexes, blockRegexes, applyFunc, postValidateApplyFunc)
    245     )[1]
    246
--> 247     return _genericInput(
    248         prompt=prompt,
    249         default=default,

~\anaconda3\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 user enter input again.

RetryLimitException:
```

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

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

In [10]:
```
pyip.inputStr(limit=3, default='hello')
```

```
Blank values are not allowed.

Blank values are not allowed.

Blank values are not allowed.
```

Out[10]:
```
'hello'
```

In [ ]:
```

```