**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

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

pypi is alias of PyInputPlus. The as pypi 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

**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

pyip.inputInt(min = 0, max =99)

**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.

response = pyip.inputNum(allowRegexes=[r'(I|V|X|L|C|D|M)+', r'zero']) :- it allows roman letters as numbers too.

response = pyip.inputNum(blockRegexes=[r'[02468]$']) :- blocks the even numbers

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

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)

/usr/local/lib/python3.7/dist-packages/pyinputplus/\_\_init\_\_.py in \_genericInput(prompt, default, timeout, limit, applyFunc, validationFunc, postValidateApplyFunc, passwordMask)

**167** possibleNewUserInput = validationFunc(

--> 168 userInput

**169** ) # If validation fails, this function will raise an exception. Returns an updated value to use as user input (e.g. stripped of whitespace, etc.)

/usr/local/lib/python3.7/dist-packages/pyinputplus/\_\_init\_\_.py in <lambda>(value)

**243** validationFunc = lambda value: pysv.\_prevalidationCheck(

--> 244 value, blank=blank, strip=strip, allowRegexes=allowRegexes, blockRegexes=blockRegexes, excMsg=None,

**245** )[1]

/usr/local/lib/python3.7/dist-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 == "":

/usr/local/lib/python3.7/dist-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)

<ipython-input-11-af00d69b8cd2> in <module>()

----> 1 response = pyip.inputStr(limit=3)

/usr/local/lib/python3.7/dist-packages/pyinputplus/\_\_init\_\_.py in inputStr(prompt, default, blank, timeout, limit, strip, allowRegexes, blockRegexes, applyFunc, postValidateApplyFunc)

**252** applyFunc=applyFunc,

**253** postValidateApplyFunc=postValidateApplyFunc,

--> 254 validationFunc=validationFunc,

**255** )

**256**

/usr/local/lib/python3.7/dist-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:

**7. 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