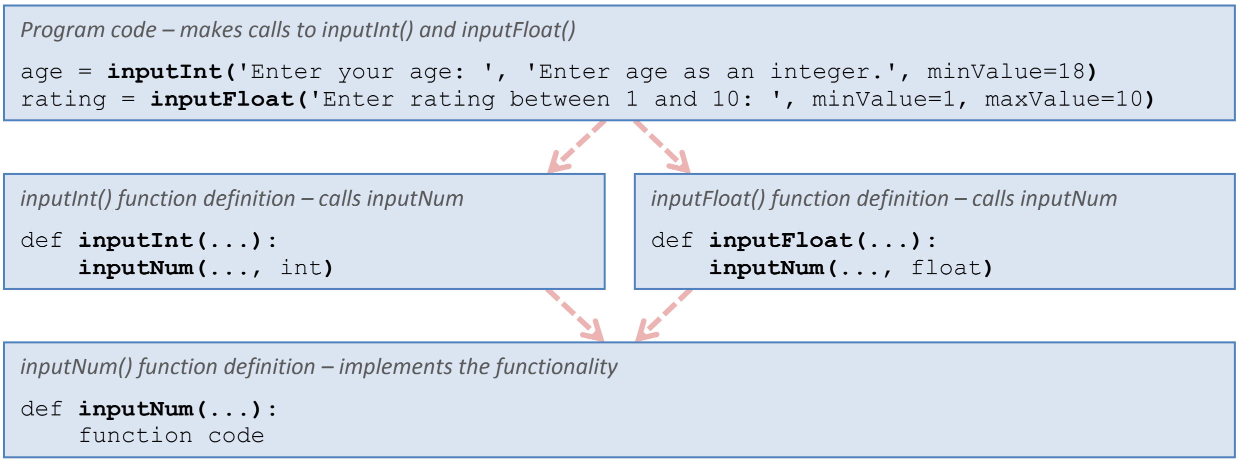
CSP1150/CSP5110: Programming Principles
Reading 4.3: Generalising inputInt() and inputFloat()

This reading follows on from Task 4 of Workshop 4. If you have not yet completed this task, please do so before continuing this reading. After completing the task, you are likely to have realised that the code of "`inputInt()`" and "`inputFloat()`" are almost exactly the same apart from what data type it converts the input to. You may also have realised that it is becoming cumbersome to make changes to two almost identical functions whenever we want to enhance them.

It would be more efficient to turn them into one function that can convert to either integer or float, using an extra parameter to allow the desired data type to be specified. To maintain convenience and compatibility, the "`inputInt()`" and "`inputFloat()`" functions will still exist and be used as normal – however behind the scenes they will both be calling the same function.



Program code – makes calls to inputInt() and inputFloat()

```
age = inputInt('Enter your age: ', 'Enter age as an integer.', minValue=18)
rating = inputFloat('Enter rating between 1 and 10: ', minValue=1, maxValue=10)
```

inputInt() function definition – calls inputNum

```
def inputInt(...):
    inputNum(..., int)
```

inputFloat() function definition – calls inputNum

```
def inputFloat(...):
    inputNum(..., float)
```

inputNum() function definition – implements the functionality

```
def inputNum(...):
    function code
```

As the diagram above shows, the program code still calls "`inputInt()`" and "`inputFloat()`". However, instead of each implementing the functionality themselves, those functions will now be calling "`inputNum()`", which implements the functionality. The "`inputNum()`" function requires an additional parameter to specify whether to require an integer or a float. When "`inputInt()`" calls "`inputNum()`", it specifies this parameter as `int`. When "`inputFloat()`" calls "`inputNum()`", it specifies this parameter as `float`.

To implement this new structure, first **copy and paste the code of your "`inputFloat()`" function** into the same file, and **rename the new copy to "inputNum"**. It does not matter whether the code for "`inputNum()`" is below or above the code for "`inputInt()`" and "`inputFloat()`".

Next, make the following changes to the parameters of "`inputNum()`":

1. **Remove the default values of the errorMessage, minValue and maxValue parameters**. Since "`inputNum()`" will only ever be called via "`inputInt()`" or "`inputFloat()`", the default values will be defined in those functions rather than in "`inputNum()`".
2. **Add a new parameter named "`requiredType`"**, which also doesn't have a default value. This additional parameter will tell "`inputNum()`" which data type to convert to/require.

The function definition line for "`inputNum()`" should end up looking like this:

Python

```
def inputNum(prompt, errorMessage, minValue, maxValue, requiredType):
```

Once the parameters have been changed, **change the code in the "try" part of "`inputNum()`"** that converts the data type of the input so that it matches the code below (you may need to change variable names so that they match the variable names you used in the rest of your function code):

Python

```
try:
    if requiredType == int:
        numResponse = int(response)

    elif requiredType == float:
        numResponse = float(response)
```

The lack of quote marks around "`int`" and "`float`" are not mistakes – they are references to the int and float data types, not strings. When "`inputInt()`" and "`inputFloat()`" call "`inputNum()`", they will specify int or float in the same way for the `requiredType` parameter.

The last thing to do is **delete the function code from "`inputInt()`" and "`inputFloat()`" and replace it with a call to "`inputNum()`"**.

Python

```
def inputInt(prompt, errorMessage='Invalid input: int required.', minValue=None, maxValue=None):
    return inputNum(prompt, errorMessage, minValue, maxValue, int)

def inputFloat(prompt, errorMessage='Invalid input: float required.', minValue=None, maxValue=None):
    return inputNum(prompt, errorMessage, minValue, maxValue, float)
```

As you can see, all that these functions now do is call "`inputNum()`", passing it all of the parameters they were called with (as well as either int or float), and return the result.

While this exercise has been relatively small-scale, we are left with a useful set of functions that reuse code in an efficient manner. Hopefully it also helped you to visualise how specific tasks within a program can be separated into functions.

The full code for the functions is on the next page – if you find them useful, feel free to save the code as a ".py" file and use it as a module – remember to remove the testing code from the bottom!

Python

```python
# inputNum is called by inputInt and inputFloat
# it prompts for input, converts to int or float checks min and max
# if there is an error converting or with the min or max, it re-prompts for input
def inputNum(prompt, errorMessage, minValue, maxValue, requiredType):
    while True:
        response = input(prompt)

        try:
            if requiredType == int:
                numResponse = int(response)

            elif requiredType == float:
                numResponse = float(response)

        except ValueError:
            print(errorMessage)
            continue

        if minValue is not None and numResponse < minValue:
            print('Invalid input: minimum allowable value is ', minValue, '.', sep='')
            continue

        if maxValue is not None and numResponse > maxValue:
            print('Invalid input: maximum allowable value is ', maxValue, '.', sep='')
            continue

        return numResponse


# inputInt allows users to specify a prompt, error message, min and max value for an integer input
def inputInt(prompt, errorMessage='Invalid input: int required.', minValue=None, maxValue=None):
    return inputNum(prompt, errorMessage, minValue, maxValue, int)


# inputFloat allows users to specify a prompt, error message, min and max value for a float input
def inputFloat(prompt, errorMessage='Invalid input: float required.', minValue=None, maxValue=None):
    return inputNum(prompt, errorMessage, minValue, maxValue, float)


# testing code / examples (remove this if using the functions as a module)
age = inputInt('Enter your age: ', 'Enter age as an integer.', minValue=18)
rating = inputFloat('Enter rating between 1 and 10: ', minValue=1, maxValue=10)
print('You entered an age of', age, 'and a rating of', rating)
```