Computing

Lesson 4: Data Validation

Programming Part 3: Iteration

Rebecca Franks

[1] *Materials from the Teach Computing Curriculum created by the National Centre for Computing Education*

All likely inputs

All likely inputs

The following code is used for data entry:

```
print("Enter a number between 1 and 10:")
number = int(input())
```

Here is a list of some of the possible entries that the user could make:

```
Five
7
7.5
12
-5
Enter key pressed
8]
```

```
Twelve
twelve
9
10
83
0
3]
#
```

All likely inputs

Sort the likely inputs from the previous slide into the following four categories:

- Correct
- Out of range
- ValueError
- Empty string

Adding validation checks

Enter a number program

For this task you will need the program that you just created in the previous task.

You should have already modified this program by incorporating a while loop:

oaknat.uk/comp-validationstartercode

```
try:
  print("Enter a number between 1 and 10:")
  number = int(input())
except ValueError:
  print("You must enter a number between 1 and 10:")
  number = int(input())
```

Task 1: Adding a range check

Step 1

An if statement is required to check if the number entered is within the correct range. Some incomplete code has been created below to assist you with this:

```
if                  and                     :
    not_validated = False
else:
    print("Number entered out of range")
```

Place a tick ✓ next to the instruction when you have completed it.

- [ ] Complete the code and incorporate it within the while loop.

Step 2

Test your code to make sure that it works as expected. Use the table on the next slide to help.

Task 1: Adding a range check

Example

Note: Use this example to check your program. Given the input you see in this sample interaction, this is the output your program should produce.

The user is prompted to enter a number	`Enter a number between 1 and 10:`
The user enters a letter	`F`
The user is reminded to enter a number between 1 and 10	`You must enter a number between 1 and 10:` `Enter a number between 1 and 10:`
The user enters a number	`12`
The user is told that the number is out of range and asked to enter the number again	`Number entered out of range` `Enter a number between 1 and 10:`
The user enters a number	`7`
The program ends	`>>>`

Task 2: Check for all likely inputs

Your code should now work when all likely inputs are entered. Test your code to make sure that it doesn't break when:

- [ ] The correct data is entered
- [ ] The data entered is out of range
- [ ] The data entered is not an integer
- [ ] No data is entered (the user presses enter without entering any input)

Task 3: Enter a name program

Step 1

Create a new file in Repl.it and enter the code below:

```
print("Enter a name:")
name = input()
print(f"Stored name: {name}")
```

Step 2

The user must not be able to leave this question blank.

Create a while loop that will continue to ask for the name if the user presses the enter key.

Tip: If nothing is entered in `name` then it will be equal to `""`

Task 3: Enter a name program

Step 3

Test your program to make sure that the user is prompted if they press the enter key without entering any data.

Use the table on the next slide to help you with this.

Task 3: Enter a name program

Example

Note: Use this example to check your program. Given the input you see in this sample interaction, this is the output your program should produce.

The user is prompted to enter a name	`Enter a name:`
The user presses the enter key	
The user is reminded that the name cannot be left blank and is prompted to enter a name	`Name cannot be left blank` `Enter a name:`
The user presses the enter key	
The user is reminded that the name cannot be left blank and is prompted to enter a name	`Name cannot be left blank` `Enter a name:`
The user enters a name	`George`
The name is displayed for the user	`Stored name: George`

Task 4: Adding validation checks to your times table program

Step 1

- Find and open a copy of your completed times table quiz from Lesson 3. If you do not have access to this then use this repl.it (**oaknat.uk/comp-ks4-timestablequiz**). It should already have some basic validation techniques used.
- Save the file under a different name

Step 2

There are three opportunities for user input in this quiz.

- Using the table on the next slide, make a list of the three variables that are assigned an input
- For each variable, decide what the likely inputs might be

Task 4: Adding validation checks to your times table program

Copy and complete the table below using the instructions on the previous slide.

Inputs	Likely inputs

Task 4: Adding validation checks to your times table program

Step 3

- [ ] Incorporate validation checks for each variable that requires user input.

Step 4

- [ ] Test your program using all the likely inputs as test data

Parson’s puzzle

Parson’s puzzle

Take a look at the code on the next slide. It contains all of the code needed to complete a validation check. The program should only allow a number to be entered as input.

- Your job is to rearrange the lines of code so that the program will:
- Prompt the user to enter an integer
- If the user doesn’t enter an integer then it should remind the user to enter a number
- It should continue to check for an integer and remind the user until an integer is entered

Note: You might need to add indents if they are needed

Parson’s puzzle

```
print("You must enter a number:")
try:
except ValueError:
print("Enter a number:")
not_validated = True
while not_validated:
number = int(input())
not_validated = False

```