Warm Up Project Exercises

It is time to get you to put together all your skills to start building usable projects! Before you jump into our full milestone project, we will go through some warm-up component exercises, to get you comfortable with a few key ideas we use in the milestone project and larger projects in general, specifically:

- Getting User Input
- Creating Functions that edit variables based on user input
- Generating output
- Joining User Inputs and Logic Flow

Function to Display Information

Creating a function that displays a list for the user

In [3]:
```
def display_list(mylist):
    print(mylist)
```

In [4]:
```
mylist = [0,1,2,3,4,5,6,7,8,9,10]
display_list(mylist)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Accepting User Input

Creating function that takes in an input from user and returns the result in the correct data type. Be careful when using the input() function, running that cell twice without providing an input value will cause python to get hung up waiting for the initial value on the first run. You will notice an In[*] next to the cell if it gets stuck, in which case, simply restart the kernel and re-run any necessary cells.

In [8]:
```
input('Please enter a value: ')
```

```
Please enter a value: 2
```

Out[8]:
```
'2'
```

In [10]:
```
result = input("Please enter a number: ")
```

```
Please enter a number: 2
```

In [11]:
```
result
```

Out[11]:
```
'2'
```

In [12]:
```
type(result)
```

Out[12]:
```
str
```

In [13]:
```
int(result)
```

Out[13]:
```
2
```

In [15]:
```
result = int(input("Please enter a number: "))
```

```
Please enter a number: 2
```

In [17]:
```
type(result)
```

Out[17]:
```
int
```

In [19]:
```
# Example of an error!
result = int(input("Please enter a number: "))
```

```
Please enter a number: two
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-19-202dd8101f66> in <module>()
      1 # Example of an error!
----> 2 result = int(input("Please enter a number: "))

ValueError: invalid literal for int() with base 10: 'two'
```

Creating a function to hold this logic:

In [20]:
```
def user_choice():
    '''
    User inputs a number (0-10) and we return this in integer form.
    No parameter is passed when calling this function.
    '''
    choice = input("Please input a number (0-10)")

    return int(choice)
```

In [21]:
```
user_choice()
```

```
Please input a number (0-10)2
```

Out[21]:
```
2
```

In [22]:
```
result = user_choice()
```

```
Please input a number (0-10)2
```

In [23]:
```
result
```

Out[23]:
```
2
```

In [24]:
```
type(result)
```

Out[24]:
```
int
```

Validating User Input

Check that input is valid before attempting to convert.

We'll use a simple method here.

As you get more advanced, you can start looking at other ways of doing this (these methods will make more sense later on in the course, so don't worry about them for now).

- Various Posts on This
- StackOverflow Post 1
- StackOverflow Post 2

In [31]:
```
some_input = '10'
```

In [32]:
```
# Lot's of .is methods availble on string
some_input.isdigit()
```

Out[32]:
```
True
```

Edit the function to confirm against an acceptable value or type

In [35]:
```
def user_choice():

    # This original choice value can be anything that isn't an integer
    choice = 'wrong'

    # While the choice is not a digit, keep asking for input.
    while choice.isdigit() == False:

        # we shouldn't convert here, otherwise we get an error on a wrong input
        choice = input("Choose a number: ")

    # We can convert once the while loop above has confirmed we have a digit.
    return int(choice)
```

In [38]:
```
user_choice()
```

```
Choose a number: hello
Choose a number: two
Choose a number: 2
```

Out[38]:
```
2
```

Let's try adding an error message within the while loop!

In [39]:
```
def user_choice():

    # This original choice value can be anything that isn't an integer
    choice = 'wrong'

    # While the choice is not a digit, keep asking for input.
    while choice.isdigit() == False:

        # we shouldn't convert here, otherwise we get an error on a wrong input
        choice = input("Choose a number: ")

        # Error Message Check
```

```
        if choice.isdigit() == False:
            print("Sorry, but you did not enter an integer. Please try again.")

    # We can convert once the while loop above has confirmed we have a digit.
    return int(choice)
```

In [40]:
```
user_choice()
```

```
Choose a number: two
Sorry, but you did not enter an integer. Please try again.
Choose a number: 2
```

Out[40]:
```
2
```

Now let's explore how to "clear" the output, that way we don't see the history of the "Choose a number" statements.

NOTE: Jupyter Notebook users will use the IPython method shown here. Other IDE users (PyCharm, VS, etc..) will use

In [3]:
```
from IPython.display import clear_output
clear_output()
```

In [4]:
```
def user_choice():

    # This original choice value can be anything that isn't an integer
    choice = 'wrong'

    # While the choice is not a digit, keep asking for input.
    while choice.isdigit() == False:

        # we shouldn't convert here, otherwise we get an error on a wrong input
        choice = input("Choose a number: ")

        if choice.isdigit() == False:
            # THIS CLEARS THE CURRENT OUTPUT BELOW THE CELL
            clear_output()

            print("Sorry, but you did not enter an integer. Please try again.")


    # Optionally you can clear everything after running the function
    # clear_output()

    # We can convert once the while loop above has confirmed we have a digit.
    return int(choice)
```

In [5]:
```
user_choice()
```

```
Choose a number: 2
```

Out[5]:
```
2
```

Checking Against Multiple Possible Values

In [1]:
```
result = 'wrong value'
acceptable_values = ['0','1','2']
```

In [2]:
```
result in acceptable_values
```

Out[2]:
```
False
```

In [7]:
```
result not in acceptable_values
```

Out[7]:
```
True
```

In [16]:
```
from IPython.display import clear_output
clear_output()
```

In [11]:
```
def user_choice():

    # This original choice value can be anything that isn't an integer
    choice = 'wrong'

    # While the choice is not a digit, keep asking for input.
    while choice not in ['0','1','2']:

        # we shouldn't convert here, otherwise we get an error on a wrong input
        choice = input("Choose one of these numbers (0,1,2): ")

        if choice not in ['0','1','2']:
            # THIS CLEARS THE CURRENT OUTPUT BELOW THE CELL
            clear_output()

            print("Sorry, but you did not choose a value in the correct range (0,1,2)")


    # Optionally you can clear everything after running the function
    # clear_output()

    # We can convert once the while loop above has confirmed we have a digit.
    return int(choice)
```

In [12]:
```
user_choice()
```

Out[12]:
```
Choose one of these numbers (0,1,2): 1
1
```

More Flexible Example

In [1]:
```
def user_choice():

    choice ='WRONG'
    within_range = False

    while choice.isdigit() == False or within_range == False:



        choice = input("Please enter a number (0-10): ")

        if choice.isdigit() == False:
            print("Sorry that is not a digit!")

        if choice.isdigit() == True:
            if int(choice) in range(0,10):
                within_range = True
            else:
                within_range = False
```

```
    return int(choice)
```

In [2]:
```
user_choice()
```

```
Please enter a number (0-10): 12
Please enter a number (0-10): 2
```

Out[2]:
```
2
```

Simple User Interaction

Finally let's combine all of these ideas to create a small game where a user can choose a "position" in an existing list and replace it with a value of their choice.

In [2]:
```
game_list = [0,1,2]
```

In [10]:
```
def display_game(game_list):
    print("Here is the current list")
    print(game_list)
```

In [11]:
```
display_game(game_list)
```

```
Here is the current list
['hi', 'no', 2]
```

In [12]:
```
def position_choice():

    # This original choice value can be anything that isn't an integer
    choice = 'wrong'

    # While the choice is not a digit, keep asking for input.
    while choice not in ['0','1','2']:

        # we shouldn't convert here, otherwise we get an error on a wrong input
        choice = input("Pick a position to replace (0,1,2): ")

        if choice not in ['0','1','2']:
            # THIS CLEARS THE CURRENT OUTPUT BELOW THE CELL
            clear_output()

            print("Sorry, but you did not choose a valid position (0,1,2)")


    # Optionally you can clear everything after running the function
    # clear_output()

    # We can convert once the while loop above has confirmed we have a digit.
    return int(choice)
```

In [13]:
```
def replacement_choice(game_list,position):

    user_placement = input("Type a string to place at the position")

    game_list[position] = user_placement

    return game_list
```

In [14]:
```
def gameon_choice():

    # This original choice value can be anything that isn't a Y or N
    choice = 'wrong'

    # While the choice is not a digit, keep asking for input.
    while choice not in ['Y','N']:

        # we shouldn't convert here, otherwise we get an error on a wrong input
        choice = input("Would you like to keep playing? Y or N ")


        if choice not in ['Y','N']:
            # THIS CLEARS THE CURRENT OUTPUT BELOW THE CELL
            clear_output()

            print("Sorry, I didn't understand. Please make sure to choose Y or N.")


    # Optionally you can clear everything after running the function
    # clear_output()

    if choice == "Y":
        # Game is still on
        return True
    else:
        # Game is over
        return False
```

Game Logic All Together

In [18]:
```
# Variable to keep game playing
game_on = True

# First Game List
game_list = [0,1,2]



while game_on:

    # Clear any historical output and show the game list
    clear_output()
    display_game(game_list)

    # Have player choose position
    position = position_choice()

    # Rewrite that position and update game_list
    game_list = replacement_choice(game_list,position)

    # Clear Screen and show the updated game list
    clear_output()
    display_game(game_list)

    # Ask if you want to keep playing
    game_on = gameon_choice()
```

```
Here is the current list
['34', 1, 'new value']
Would you like to keep playing? Y or N N
```

Great work! You now have an understanding of bringing functions and loop logics together to build a simple game. This will be expanded upon in the Milestone project!