Welcome Back

- We will answer the question asked in the last video
- And look at…
  - Setting an alias for imported module functions
  - A new function
  - More docstings
  - Try and Except Block

```
import validate_module
user_value = input("Enter a number:")
an_int = validate_module.is_integer(user_value)
if (an_int):
    print("yep, an integer")
else:
    print("Not an integer")
```

The result will be: “Not an integer”

All user input are strings. Thus, 42 is actually “42”.

validate_module namespace

- a namespace with the name of the module name was created that held all the names within that module
- we need to access the name space first using dot notation.

```
an_int = validate_module.is_integer(user_input)
```

- time consuming typing
  `validate_module.function_name()`
  for each and every function that we wish to call

Import module as an alias

- We could use the **`as`** keyword to set an alias for the imported module
- Wouldn't it be nice if we could do something like this instead?

```
an_int = check.is_integer(user_value)
```

- If we used the as keyword with the import, we can do just that.
- This is quite a common approach.
- Let's take a look. We will create a new function in our module to convert a string to an integer
  - We don't want the program to crash if it can't convert.

Objective

- Create a new function in the validate_module named convert_to_int with one variable in the parameter list
- Create a docstring explain the purpose of the function
- Try to convert the value to an integer
- If successful in converting to an integer, return the integer value
- Otherwise, return the original value

```
def convert_to_int(value):
    """ Returns the integer representation of
        the value, or the original value if it
        can't be converted to an integer """
    try:
        value = int(value)
    except:
        pass
    return value
```

pass is a null operation -- when it is executed, nothing happens. It is useful as a placeholder when a statement is required syntactically, but no code needs to be executed

Objective

- Import our module using **`import`** and **`as`** keywords
- Keep getting input from the user until they enter a valid integer
- The program should not crash on bad input

```
import validate_module as check


while True:
    user_input = input("Enter a number: ")
    user_input = check.convert_to_int(user_input)
    if (check.is_integer(user_input)):
        break
```

```
Validate User Input Imported.
Enter a number: No
Enter a number: Why should I?
Enter a number: hmm
Enter a number: ok
Enter a number: 42!!!!!
Enter a number: 4 2
Enter a number: 42

Process returned 0 (0x0)
execution time : 10.187 s
Press any key to continue . . .
```

End of Part 3

- Time for a break. Cup of tea?
- Click on the next video when refreshed and ready and we will look at other ways to import modules and functions