Luis F Lopez Penunuri
February 28, 2023
IT FDN 110 A
Assignment 07
https://github.com/fernandoPenunuri34/IntroToProg-Python-Mod07.git

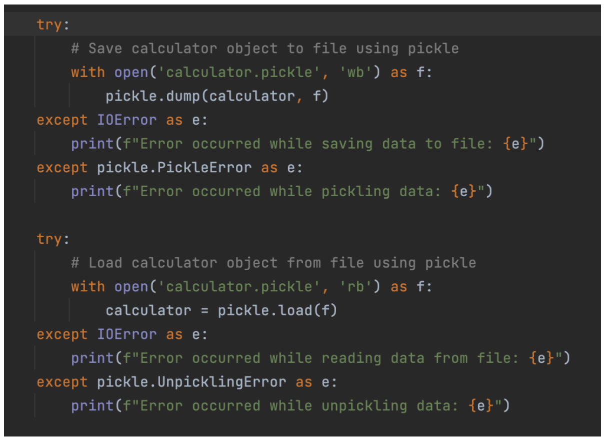
Assignment 07

Introduction

This assignment required us to implement exception management. Root gave us the opportunity to choose what we wanted to do to implement it. In my practice, I continued talking with Jose to understand his needs. He said that he was having issues with his clients since most of them use imperial system metrics (lbs and inches), but he only understand international metrics (mts and kg). So, I wanted to help him by creating a script that would assist him in making the conversions.

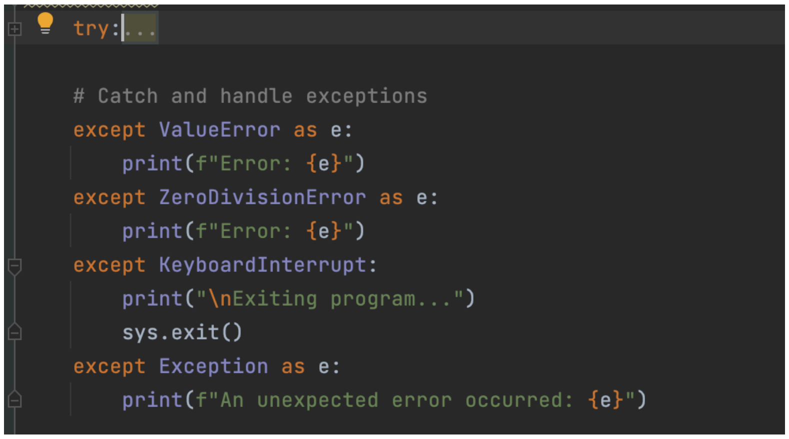
Practice

In this practice, we implemented 'Try-Catch' to handle exceptions. To understand it better, I implemented as many as I could. The try-catch block allows us to catch and handle exceptions so that the program can continue executing instead of crashing. The try block contains the code that may raise an exception, while the except block contains the code that should be executed when an exception occurs (Figure 7.1).



```python
try:
    # Save calculator object to file using pickle
    with open('calculator.pickle', 'wb') as f:
        pickle.dump(calculator, f)
except IOError as e:
    print(f"Error occurred while saving data to file: {e}")
except pickle.PickleError as e:
    print(f"Error occurred while pickling data: {e}")

try:
    # Load calculator object from file using pickle
    with open('calculator.pickle', 'rb') as f:
        calculator = pickle.load(f)
except IOError as e:
    print(f"Error occurred while reading data from file: {e}")
except pickle.UnpicklingError as e:
    print(f"Error occurred while unpickling data: {e}")
```

Figure 7.1 Handle Exception



```python
try:...

    # Catch and handle exceptions
    except ValueError as e:
        print(f"Error: {e}")
    except ZeroDivisionError as e:
        print(f"Error: {e}")
    except KeyboardInterrupt:
        print("\nExiting program...")
        sys.exit()
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

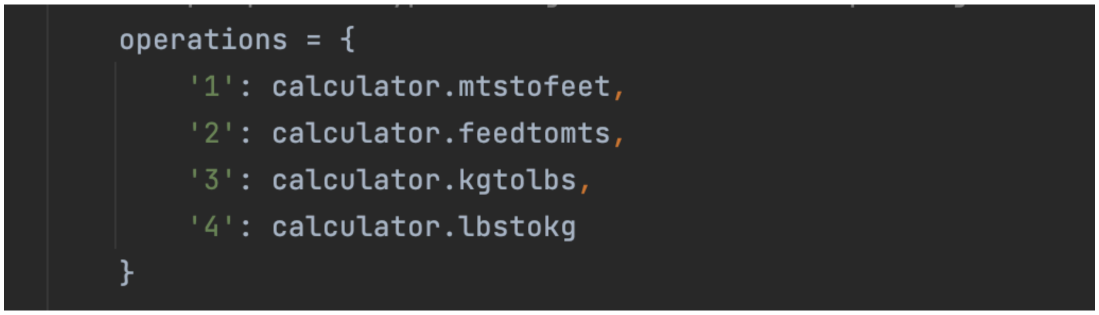
Figure 7.2 Handle Exception in the Main

Using Method:

Dictionary:

The menu is being handle by a dictionary called ‘operations’ that maps strings representing different types of conversions to the corresponding conversion functions defined in ‘calculator’ module.

The key is a string that represents the user’s choice of operation, while the value is the corresponding function that will be executed when the user selects that option (Figure 7.3).



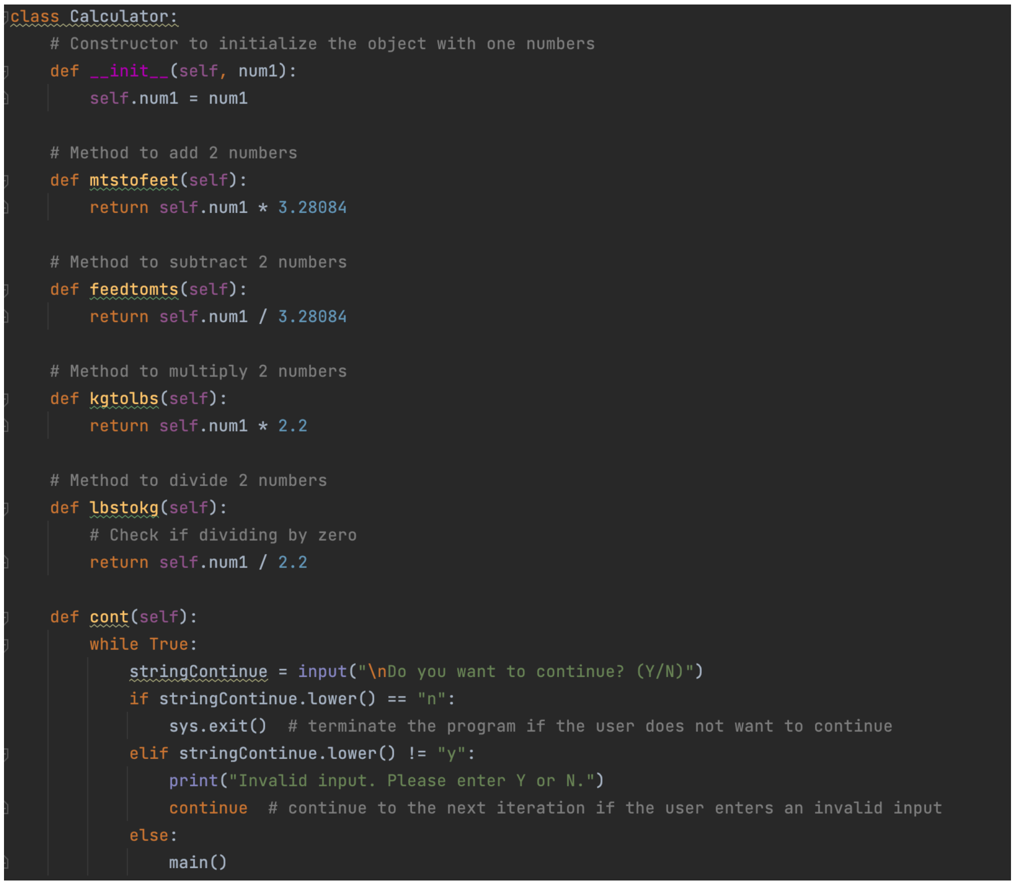
```python
operations = {
    '1': calculator.mtstofeet,
    '2': calculator.feedtomts,
    '3': calculator.kgtolbs,
    '4': calculator.lbstokg
}
```

Figure 7.3, Dictionary to handle the menu

self.num1

'Self' refers to the instance of the class that the method belongs to. In script, 'self.num1' refers to the 'num1' attribute of the instance of the 'Calculator' class.

In the same section of the contractor, the '_ _init_ _' method is a special method in Python classes that is called when an instance of the class is created. The `self.num1 = num1` statement in the `__init__` method initializes the num1 attribute of the instance with the value passed to the method as num1.
The other methods in the `Calculator` class also use `self.num1` to perform calculations based on the value of num1 stored in the instance.



```python
class Calculator:
    # Constructor to initialize the object with one numbers
    def __init__(self, num1):
        self.num1 = num1

    # Method to add 2 numbers
    def mtstofeet(self):
        return self.num1 * 3.28084

    # Method to subtract 2 numbers
    def feedtomts(self):
        return self.num1 / 3.28084

    # Method to multiply 2 numbers
    def kgtolbs(self):
        return self.num1 * 2.2

    # Method to divide 2 numbers
    def lbstokg(self):
        # Check if dividing by zero
        return self.num1 / 2.2

    def cont(self):
        while True:
            stringContinue = input("\nDo you want to continue? (Y/N)")
            if stringContinue.lower() == "n":
                sys.exit()  # terminate the program if the user does not want to continue
            elif stringContinue.lower() != "y":
                print("Invalid input. Please enter Y or N.")
                continue  # continue to the next iteration if the user enters an invalid input
            else:
                main()
```

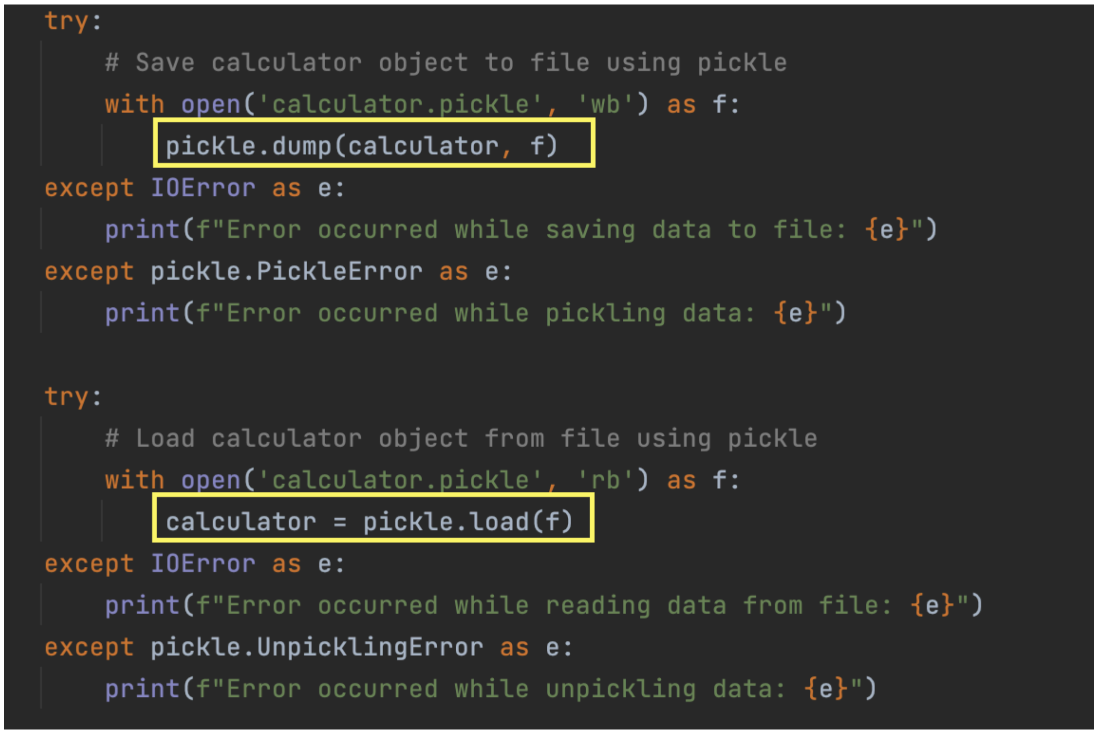
Figure 7.4 Init method (https://www.w3schools.com/python/gloss_python_class_init.asp)

pickle

The pickle module implements binary protocols for serializing and de-serializing a Python object structure (taken from https://docs.python.org/3/library/pickle.html).

In this code, I'm using the dump function to serialize an object hierarchy. This function writes the pickle representation of the object obj to the open file object file.

Pickle.loads returns the reconstituted object hierarchy of the pickled representation data of an object. The data must be a bytes-like object. The protocol version of the pickle is detected automatically, so no protocol argument is needed (taken from https://docs.python.org/3/library/pickle.html) (Figure 7.5).

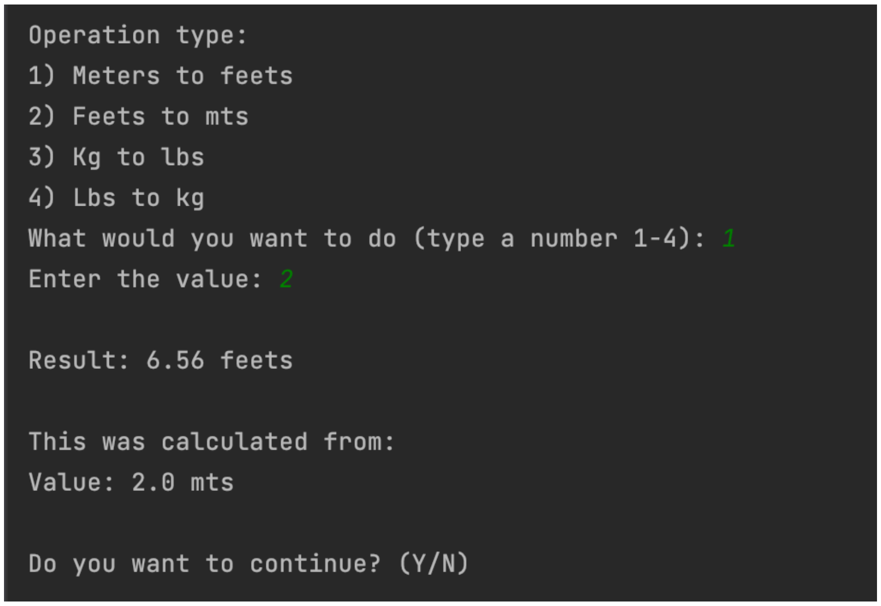


```python
try:
    # Save calculator object to file using pickle
    with open('calculator.pickle', 'wb') as f:
        pickle.dump(calculator, f)
except IOError as e:
    print(f"Error occurred while saving data to file: {e}")
except pickle.PickleError as e:
    print(f"Error occurred while pickling data: {e}")

try:
    # Load calculator object from file using pickle
    with open('calculator.pickle', 'rb') as f:
        calculator = pickle.load(f)
except IOError as e:
    print(f"Error occurred while reading data from file: {e}")
except pickle.UnpicklingError as e:
    print(f"Error occurred while unpickling data: {e}")
```

Figure 7.5 pickle example

Prove

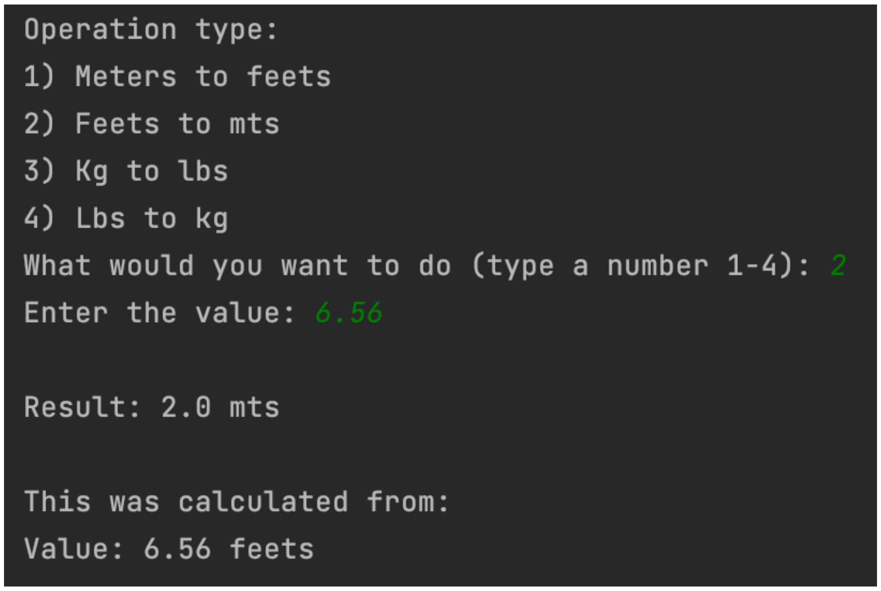
Option number 1 will do a conversion to feet from meters (Figure 7.6).



```
Operation type:
1) Meters to feets
2) Feets to mts
3) Kg to lbs
4) Lbs to kg
What would you want to do (type a number 1-4): 1
Enter the value: 2

Result: 6.56 feets

This was calculated from:
Value: 2.0 mts

Do you want to continue? (Y/N)
```

Figure 7.6 Option 1

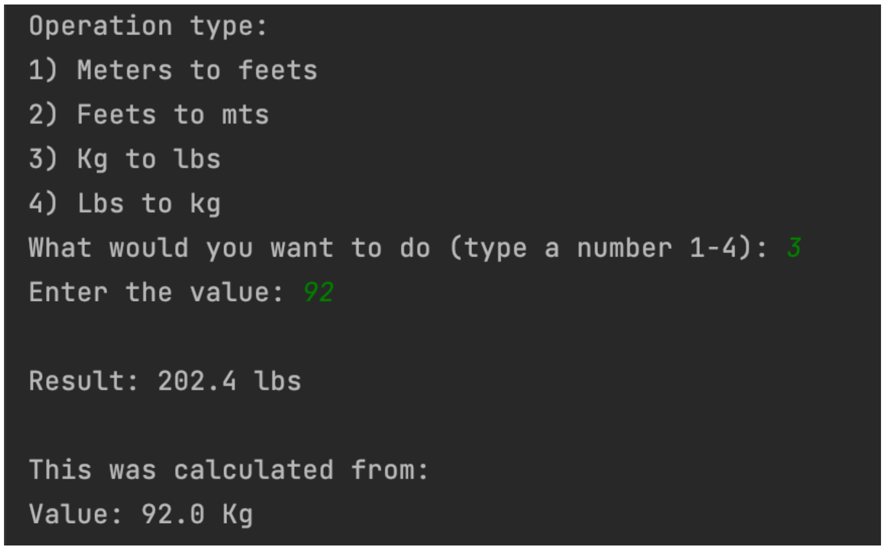
Option number 2 will do a conversion to meters from feet (Figure 7.7)



```
Operation type:
1) Meters to feets
2) Feets to mts
3) Kg to lbs
4) Lbs to kg
What would you want to do (type a number 1-4): 2
Enter the value: 6.56

Result: 2.0 mts

This was calculated from:
Value: 6.56 feets
```

Figure 7.7 Option 2

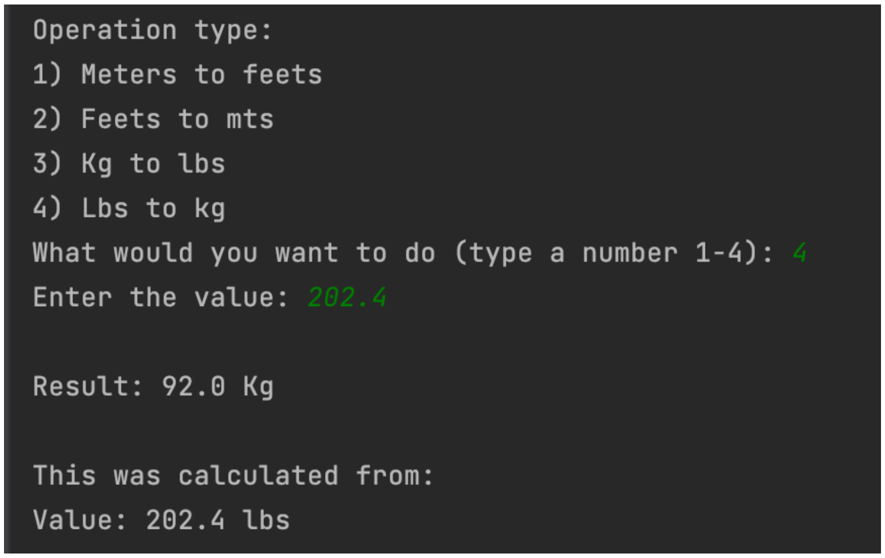
Option number 3 will do a conversion to lbs from Kg (Figure 7.8).



```
Operation type:
1) Meters to feets
2) Feets to mts
3) Kg to lbs
4) Lbs to kg
What would you want to do (type a number 1-4): 3
Enter the value: 92

Result: 202.4 lbs

This was calculated from:
Value: 92.0 Kg
```

Figure 7.8 Option 3

Option number 4 will do a conversion to Kg from lbs (Figure 7.9)



```
Operation type:
1) Meters to feets
2) Feets to mts
3) Kg to lbs
4) Lbs to kg
What would you want to do (type a number 1-4): 4
Enter the value: 202.4

Result: 92.0 Kg

This was calculated from:
Value: 202.4 lbs
```

Figure 7.9 Option 4

Conclusion

This was a very interesting practice. I wasn't that creative, but it was nice to search for this information and understand it. On the other hand, handling exceptions was a good way to prevent errors in the system. I really like how it works and how easy it was to implement.

Jose was amazed by the program and how easy it was use. Now he has a tool to continue with his work. However, he was not sure if it is good enough, so I made a deal with him.

Grade	Cost (Avocados)
100	8
>90 <100	6
=>80 <=90	3
<80	1

Whether or not I can make a good amount of guacamole for the carne asada is up to you.