5 October 2018 (Tuple Review & Try/Except)

- Fall break remarks
- HW 6 will be released next Wednesday (due the Thursday after exam 2), no work over the break
- **Tuple Review**
  - Tuples are similar to lists, but are immutable !!
    - No .append(), and += returns a new tuple and reassigns it to the old tuple
    - Can't go in and change individual elements of a tuple like a_tup[0] = 5
  - unpacking
    - can assign all variables of a tuple in one line, but you have to know the number of vals in that tuple
    - a, b, c = some_tuple
      - now you can use the variables a, b, c however you want
    - **Mini-Quiz 1 (tuple_tracing_2)**
  - indexing and slicing works the same as with lists
    - a[num] accesses the value
    - a[num:num2] gives back a new tuple slice
  - Deep copies
    - since tuples are immutable, += returns a new tuple, unlike lists...
    - a = (1,2,3)
    - b = a
    - b += (4,) #only changes b, not a
    -
- **Try/Except**
  - Used to handle errors (exceptions)
  - Useful for preventing programs from stopping because of errors
    - Up to this point, only seen tracebacks and confusing red text
    - The customer wouldn't want to see that
  - 4 parts to a try-except block
    - Try
    - Except
    - Else
    - Finally
  - Try
    - Put the code that may cause an error in your try block

        - Usually used with accessing files or user input (Trying to access a file that doesn’t exist or dealing with type errors of user input)
        - Ex.
            - try:
                - x = 1 / 0
            - try:
                - x = input(“Give me a number please!”)
                - x = int(x)
    - Except
        - Code that is run if an exception occurs in the try block (catches all types of errors by default)
        - all trys must have an except!
        - Ex.
            - except:
                - print(“An error occurred”)
        - Types of exceptions that might be useful to know:
            - IndexError
            - ZeroDivisionError
        - To catch those specific exceptions, put the type after except
            - except IndexError:
                - print(“Index outside of bounds”)
            - except ZeroDivisionError:
                - print(“Can’t divide by 0”)
        - You can also check for multiple types of exceptions
            - Ex.
                - try:
                    - x = input(“Number please”)
                    - y = 1 / int(x)
                    - z = [1,2,3,4]
                    - print(z[4])
                - except ZeroDivisionError:
                    - print(“You entered 0! This is not valid.”)
                - except IndexError:
                    - print(“Index is out of bounds”)
    - Else
        - The else block is run only if there is no exception raised in the try block. Comes after all except blocks
        - Ex.
            - else:

            - print(“Yay, the code worked”)
    - Finally
        - The finally block will **ALWAYS** be run, whether or not an error occurs anywhere in the try-except block
        - Ex.
            - finally:
                - print(“Try-except block is finished”)
    - Else and Finally block are not necessary, but must have them in the correct order (can’t have finally then else)

coding example (if necessary?)

```
def func(x):
    try:
        x = x/(x-1)
    except:
        print(“an error occured!”)
    else:
        pring(“there wasn’t an error!”)
    final:
        print(“the code is complete”)

func(2) #doesn’t throw error
func(1) #throws error
```

Practice Problem

```
    try:
        x = 5.5
        print(x)
        x -= 4.5
        print(x)
        x = str(x / 0)
        print(x)
        x += 3
        print(x, end = " ")
    except ZeroDivisionError:
        print("error occurred!")
        x += 1
        print(x)
    except:
        print(“!!”)
    finally:
        print("done!" + str(x))
```

I guess lecture is over here! HAVE A GOOD FALL BREAK