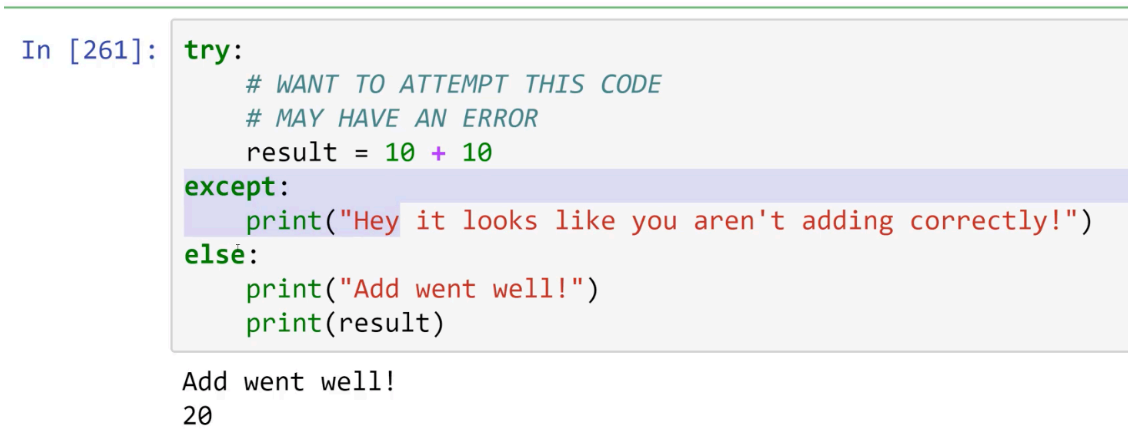
## SECTION 10: ERRORS AND EXCEPTIONS HANDLING - 46 minutes, 6 parts

### 1/6 Errors and Exception Handling

- **Errors**
    - -> the code will have errors
    - -> error handling is used to prepare for this
    - -> for example, file permissions
    - -> the code will stop and return an error statement

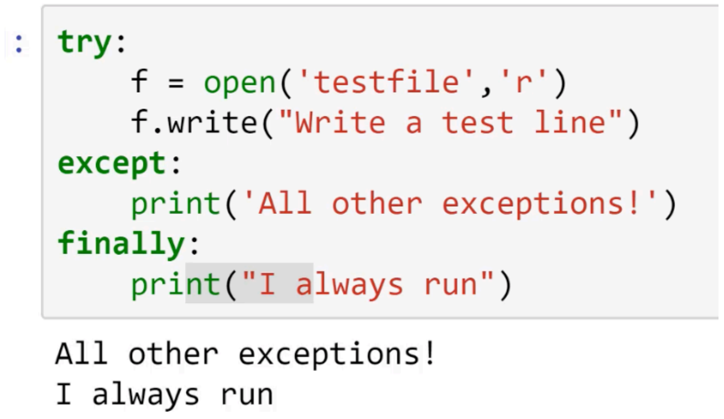
- **Keywords**
    - -> try:, except: and finally: <- these all have a block of code associated with them
    - -> try <- the block of code which is attempted, that might lead to an error
    - -> except <- a block of code which will execute is there is an error in the try block
    - -> finally <- this is what try is to except, but to try

- **Adding function example <- try, except else error handling**
    - -> he has defined a summing function
    - -> he has entered an example input into the function which deliberately returns an error
    - -> adding different types of errors
    - -> try <- try this code, it may have an error
    - -> except
    - -> he has then added in an else statement
    - -> this is a try, except, else statement



```
In [261]: try:
              # WANT TO ATTEMPT THIS CODE
              # MAY HAVE AN ERROR
              result = 10 + 10
          except:
              print("Hey it looks like you aren't adding correctly!")
          else:
              print("Add went well!")
              print(result)

          Add went well!
          20
```

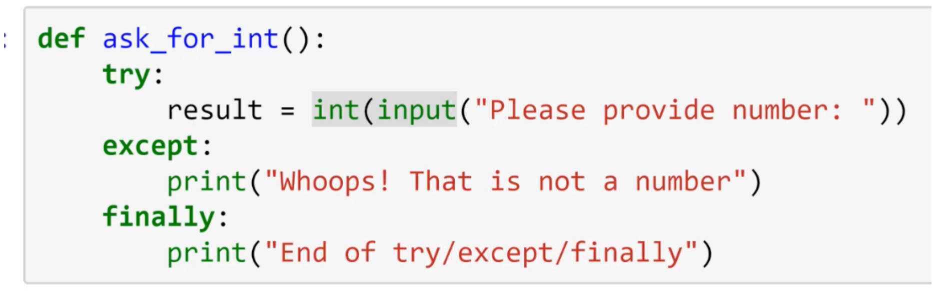
- **Write file example**
    - -> we are still working with try, except and final
    - -> .write
    - -> there are different errors we can except for
    - -> e.g specific type errors
    - -> there are different types of errors which you can look up <- e.g recursion errors
    - -> we are typing different errors into the code in this case
    - -> he is also adding in a finally block in this case <- this code executes no matter what was previously entered
    - -> so we have try, except and finally
        - -> try <- try and attempt this code
        - -> except <- in case there is an error
        - -> finally <- code which runs regardless of whether we have an error



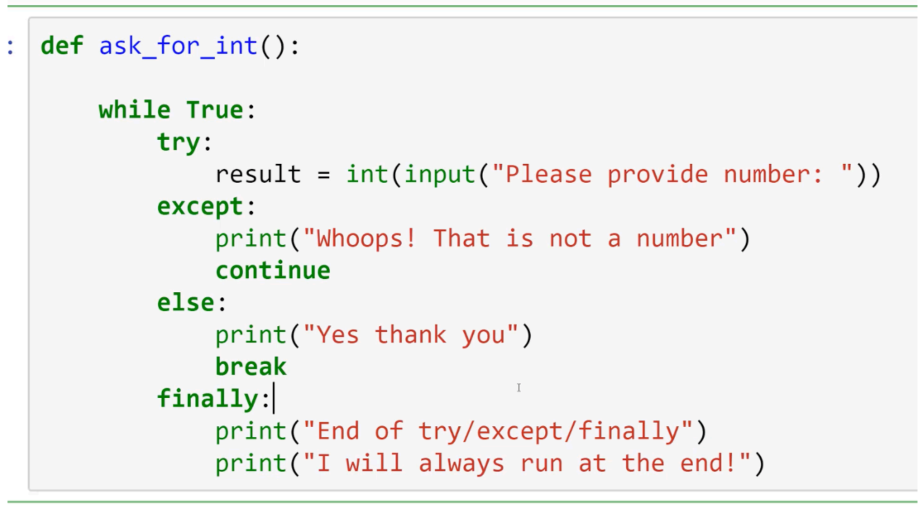
```
: try:
      f = open('testfile','r')
      f.write("Write a test line")
  except:
      print('All other exceptions!')
  finally:
      print("I always run")

  All other exceptions!
  I always run
```

- **Ask for integer example**
    - -> this is a function which asks for a number

- -> then we have error handling depending on the syntax of the input
- -> he is then placing the code inside a while loop
- -> we carry on running it until we have code which does not return an error
- -> for while loops, we must use a break statement somewhere <- to avoid getting stuck inside an infinite loop
- -> we then run the code, to test the function
- -> when we test it, we deliberately put different arguments into the function which we know will return error messages
- -> this carries on asking the user for an input, until the condition that the input is True
- -> and we have to use a break statement



```
def ask_for_int():
    try:
        result = int(input("Please provide number: "))
    except:
        print("Whoops! That is not a number")
    finally:
        print("End of try/except/finally")
```



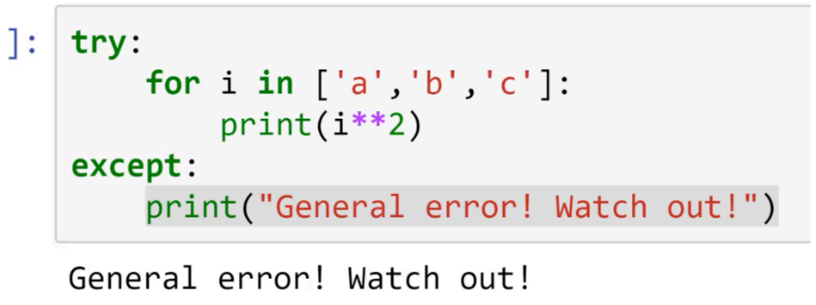
```
def ask_for_int():

    while True:
        try:
            result = int(input("Please provide number: "))
        except:
            print("Whoops! That is not a number")
            continue
        else:
            print("Yes thank you")
            break
        finally:
            print("End of try/except/finally")
            print("I will always run at the end!")
```

**2/6 Errors and Exceptions Homework**

- **Homework (three problems)**
  - -> this is in the second notebook
  - -> the first problem is using the try and except blocks
  - -> the second question is to catch a ZeroDivisionError
  - -> the third is a function which prints x**2
  - -> then we are accounting for incorrect inputs

**3/6 Errors and Exception Homework - Solutions**

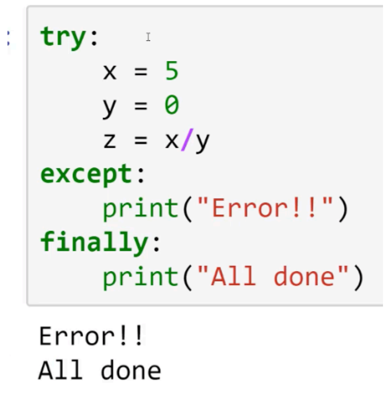
- **Error handling for squaring strings**
  - -> we are handling the errors and exceptions
  - -> running the code alone returns an error message <- if we try and square a string
  - -> run this code, if it does not work, then return this error message



```
try:
    for i in ['a','b','c']:
        print(i**2)
except:
    print("General error! Watch out!")
```

```
General error! Watch out!
```

- **Using a "finally" block to print "all done"**
  - -> in the second example, we have a block of code which we have now indented inside an except statement



```
try:
    x = 5
    y = 0
    z = x/y
except:
    print("Error!!")
finally:
    print("All done")
```

```
Error!!
All done
```