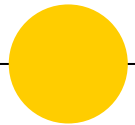
Error handling



Python error types

- **SyntaxError:** When code has been typed incorrectly.
- **AttributeError:** When you try to access an attribute on an object that does not exist.
- **KeyError:** When you try to access a key in a dictionary that does not exist.
- **TypeError:** When an argument to a function is not of the right type (e.g. a str instead of int).
- **ValueError:** When an argument to a function is of the right type but is not in the right domain (e.g. an empty string)
- **ImportError:** When an import fails.
- **IOError:** When Python cannot access a file correctly on disk.

Python error types

```
from pandas import does_not_exist

---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-105-e9d71fb092be> in <module>()
----> 1 from pandas import does_not_exist

ImportError: cannot import name 'does_not_exist'
```

Python error types

```
def func(string):
    print(does_not_exist)
func('this is a string')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-116-5b48da61d415> in <module>()
      2     print(does_not_exist)
      3
----> 4 func('this is a string')
<ipython-input-116-5b48da61d415> in func(string)
      1 def func(string):
----> 2     print(does_not_exist)
      3
      4 func('this is a string')

NameError: name 'does_not_exist' is not defined
```

Writing exceptions

```
def even_number(number):
    if number % 2 != 0:
        raise ValueError("The number entered is not even!")
    else:
        print("Number accepted.")

even_number(3)
```

It is a bad practice to code looking for exceptions, you have to decide what to do with them. You want a program that works fine!

Catching exceptions

```
try:
    even_number(3)
except:
    print("The even_number function errored out.")

print("This line of code still executes.")
```

Run this

If an error is encountered, stop and run this:

Afterwards, no matter what, run this

Catching exceptions

```
def int_check(integer):
    if type(integer) != int:
        raise ValueError("The number entered is not an integer!")
    else:
        pass
```

Catching exceptions

```
for number in numbers:
    print("Analyzing the number:", str(number))

    try:
        int_check(number)
    except:
        print("The int_check function errored out.")
    else:
        print("The int_check function ran successfully.")
        try:
            even_number(number)
        except:
            print("The even_number function errored out.")
        else:
            evens.append(number)
    print("\n")
```