Errors - not just an annoyance

Errors point out where you’re wrong

Sometimes you can expect errors:

Wrong input

Missing data

...

Handling Errors can be crucial

Try - Except

Attempts to execute code under try

If *try* code crashes, executes *except* code

Needs specification of Error type

```
def average(number_list):
    total = 0
    for number in number_list:
        try:
            total += number
        except TypeError:
            print(‘Element is not a number’)
```

Error types

- SyntaxError
- IndexError
- KeyError
- TypeError
- ValueError
- NameError
- ZeroDivisionError
- ...

Exercises - Errors

- Write a function that transcribes DNA into RNA.
- Think about what problems could occur when someone uses the function.
- Write some problematic inputs for the function. Basically try to wreck your own code as much as possible.
- Try to account for your problematic inputs.
- Discuss problems and solutions with your neighbor.
- Have your neighbor design problematic inputs for your code.

Throwing Errors

It can be useful to purposely throw errors

Crashes make mistakes obvious

Don’t let code run to produce wrong results