computer science

Handling problems

- syntax errors: raised while the interpreter while executing the program reaches an instruction that does not match syntax rules (the first time the instruction is executed)

- unexpected behavior (exceptions): raised while the interpreter executes the program, due to a situation (data) not taken into account while writing the program

syntax errors

- To be solved by executing the program and identifying where the written code violates syntax rules …

exceptions

- in our context: handling simple “mistakes”, such as specifying the erroneous name of a file

try … except

```
try:
    instruction that might cause an exception
except _error_to_catch_:
    what to do if the exception occurs
else:
    what to do if the exception does not occur …
finally:
    to be done, after the except or the else part … anyway
```

try … except 2

```
try:
        open a file
except IOError:
        message to the user … nothing more that can be done
else:
        proceed with the typical main processing
finally:
        just in case …
```

try … except 3

- we use it when there is no easy way to prevent the excution of an instruction that might raise an exception

```
fileref = open(_filename_, _mode_)
```

- we could eventually check whether the file exists but not if we have the right permissions

try … except 4

```
try:
    fin = open(name, "r")
except IOError:
    print("error opening file " + name)
else:
    for line in fin:
        …
    fin.close()
```

we handle these situations …

```
val = …
div = …
if div != 0:
    ris = val / div
    …
else:
    …
```