Date I/O Exceptions

Problem

- Handling all possibilities that can result in an error can require huge and complex if/elif/else sequences!
- Sometimes it is ok that the code ends up in an error

```
1   user_input = input('write an string, int and float separated by ","')
2
3   user_list = user_input.split(', ')
4
5   user_string = user_list[0]
6   user_int = int(user_list[1]) # fail if string is an integer
7   user_float = float(user_list[2]) # fails if string is not a number
```

Handle some errors

```
1   user_input = input('write an string, int and float separated by ", "')
2
3   if user_input.count(',') != 2:
4       print('Wrong input format')
5   user_list = user_input.split(', ')
6   if len(user_list)!=2:
7       print('Wrong input format')
8   if not user_list[1].isdigit():
9       print('Wrong input format')
10  if not user_list[2].replace('.','').isdigit():
11      # will fail if there are multiple "." in the number
12      print('Wrong input format')
```

•

•

•

```
19  user_string = user_list[0]
20  user_int = int(user_list[1])
21  user_float = float(user_list[2])
```

Try Except

- Sometimes it is better to ask for forgiveness than permission

  -> `try, except`
- Try anything indented with `try`
- If anything fails, the code jumps to the `except`.
- `else` (optional): if nothing in try fails, the else part will run.
- `finally` (optional): no matter if it failed or not, this will be run as the last part of the try, except, else

```python
user_input = input('write an string, int and float separated by ", "')

try:
    user_list = user_input.split(', ')
    user_string = user_list[0]
    user_int = int(user_list[1])
    user_float = float(user_list[2])
except:
    print('Wrong input format')
```

```python
try:
    print('ok, i do this')
    do_somthing()
except Exception as e:
    print('i failed :(')
    print(f'this went wrong{e}')
else:
    print('Nice it work! then i can do this too')
finally:
    print('I dont care, I will do this anyway!')
```

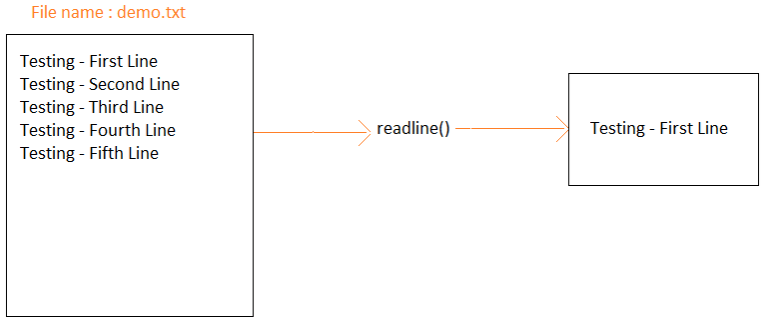
Problem

- What if you
  - Want to save some data from one session to next Example the employee data
  - Want to access files saved on the computer
  - Work on data larger than the computer can hold in RAM

```
employees = {
    10001: {
        'name': 'Ola Nordmann',
        'possition': 'CEO',
        'email': 'ola.nordmann@norge.no',
        'salary': 1_000_000
    },
    10002: {
        'name': 'Kari Nordmann',
        'possition': 'CTO',
        'email': 'kari.nordmann@norge.no',
        'salary': 1_200_000
    }
}
```

I/O

- The “open” function get access to the file
- Different open modes or read, write, etc..
- Methods:
  - `my_file.readline()`
    -> get a single line from the file
  - `my_file.write('something text/n')`
    -> write single string to file
- After we have gathered the desired information, ALWAYS CLOSE THE FILE!!
- To automatically handle closing, use “with” statement



Modes:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

"r+" - Both read and write to file

```
path = r'C:\Users\rodve\Desktop\Project\my_file.txt'
my_file = open(file=path, mode='r')
content = []
for line in my_file:
    content.append(line)
my_file.close()
```

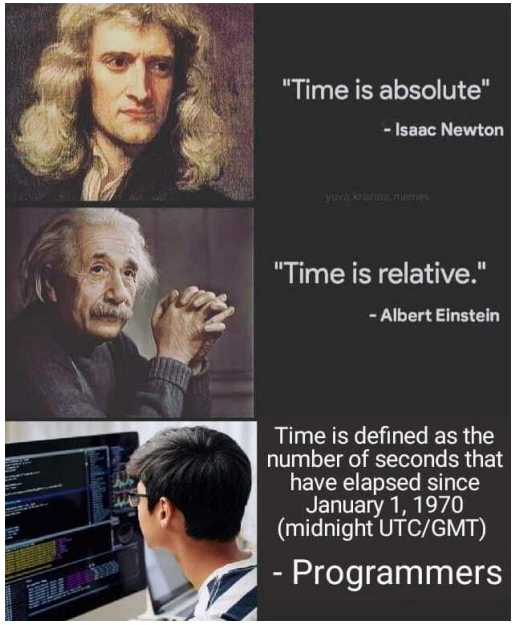
Better

```
path = r'C:\Users\rodve\Desktop\Project\my_file.txt'
content = []
try:
    my_file = open(file=path, mode='r')
    for line in my_file:
        content.append(line)
except:
    print('some error happend')
finally:
    my_file.close()
```

Best!

```
path = r'C:\Users\rodve\Desktop\Project\my_file.txt'
content = []
with open(file=path, mode='r') as my_file
    for line in my_file:
        content.append(line)
```

Datetime



- `import time`
  - `time.time()` -> 1623516457.9715495 (time in seconds)
  - `time.sleep(3)` -> makes console stop for 3 seconds
- `import datetime`
  - `my_dt = datetime.datetime.now()` -> datetime.datetime(2021,6,12,17,0,0,206203)
  - `my_dt-datetime.timedelta(days=7)` -> datetime.datetime(2021,6,5,17,0,0,206203)
  - `my_dt2 - my_dt` -> datetime.timedelta(seconds=92,microseconds=450480)
  - `my_dt.weekday()` -> 5 (Saturday, 0 is Monday)
  - `datetime.datetime.strptime('2021-06-12 17:00:01', '%Y-%m-%d %H:%M:%S')` -> string to datetime by format https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes
- `import pytz`
  - Handling time zone for your dates
- Can use `time.time()` as a stopwatch to time slow part of your code
- Motivation: https://www.youtube.com/watch?v=-5wpm-gesOY

What is the date in time.time() based on?

```
my_dt = datetime.datetime.now()
time_start_date = my_dt-datetime.timedelta(seconds=time.time())
print(time_start_date)
#>>> 1970-01-01 01:59:59.988999
```

Stopwatch to time code:

```
start_time = time.time()
# code to time here ->
result = slow_function()

end_time = time.time()
print(end_time-start_time)
# 14.59151005744934
```