Introduction to Computer Programming

Week 7.1: Reading & Writing Files

a+

Example: When we want to read and/or edit (append only).

The stream position is:

- at the *end* when opened (must be moved to the start to read).
- always moved to the *end* before writing when `write` is called (previous contents never overwritten).
- at the *end* after writing.

In [23]:
```
file = open('sample_data/scores.txt', 'a+')

file.write('Tim 50\nMajid 500\n')          # append


file.seek(0)
for line in file:                          # read
    print(line, end='')


file.close()
```

```
Elena 550
Sajid 480
Tom 380
Farhad 305
Manesha 150
Jen 100
Tim 50
Majid 500
Tim 50
Majid 500
Tim 50
Majid 500
```

r+

Example: When we want to read and/or edit.

The stream position is at the *end* of the file:

- after reading
- before appending
- after appending

In [22]:

```
file = open('sample_data/scores.txt', 'r+')

# stream position at start of file

for line in file:                               # read file content
    print(line, end='')

# stream position at end of file

file.write('Ben 50\n')                          # append some data
file.write('Ola 500\n')


file.seek(0)
for line in file:                               # read file content
    print(line, end='')


file.close()
```

```
Sid 50

Jo 20

Tim 50

Majid 500

Sid 50
Jo 20
Tim 50
Majid 500
Ben 50
Ola 500
```

w+

Example: When we want to overwrite file then read

The stream position is:

- at the *start* when opened (previous contents overwritten).
- at the *end* after writing (subsequent lines added using `write` will appended the file, not overwrite previous contents, until file is closed).

Writing *must* happen before reading.

Unlike the `+a` mode specifier `+r` allows writing from anywhere in the file.

Notice the effect of overwriting.

In [12]:

```
file = open('sample_data/scores.txt', 'w+')


# C) read does not work (would move write position to end)
# for line in file:
#     print(line)



# A) write  (overwrite prev. contents) then read
file.write('Tim 50\nMajid 500\n')

file.seek(0)

for line in file:                                   # read
    print(line, end='')



# B) append then read
file.write('Ola 500\n')


file.seek(0)

print()
for line in file:                                   # read again
    print(line, end='')


file.close()
```

```
Tim 50
Majid 500

Tim 50
Majid 500
Ola 500
```

In-class demos

Try it yourself

Example 1: Write a high score table stored as two **lists** to a new file with the name scores.csv

Hint: use a `for` loop

Solution 1a

In [31]:
```
names = ['Elena', 'Sajid', 'Tom', 'Farhad', 'Manesha']
scores = [550, 480, 380, 305, 150]

file = open('sample_data/scores.csv', 'w')

# loop through two lists
for n, s in zip(names, scores):

    file.write(n + ',' + str(s) + '\n') # a comma seperates the

    #file.write(f'{n}, {s}\n')

file.close()
```

Solution 1b

In [32]:
```
scores = {'Elena': 550,
          'Sajid': 480}

file = open('sample_data/scores.csv', 'w')

# loop through two lists - keys and values of dictionary
for k, v in scores.items():

    file.write(f'{k}, {v}\n')

    #file.write(k + ' ' + str(v) + '\n')

file.close()
```

Solution 1c

In [33]:
```
scores = {'Elena': 550,
          'Sajid': 480}

with open('sample_data/scores.csv', 'w') as file:

    # loop through two lists - keys and values of dictionary
    for k, v in scores.items():

        file.write(f'{k}, {v}\n')

        #file.write(k + ' ' + str(v) + '\n')
```

Try it yourself

Example 2: Read the file you just created and print each line

Solution 2

In [34]:
```
with open('sample_data/scores.csv', 'r') as file:

    for line in file:            # iterable: collect names and
        print(line)
```

```
Elena, 550

Sajid, 480
```

Try it yourself

Example 3: Read the file you just created and print the first row

Solution 3

In [35]:
```
with open('sample_data/scores.csv', 'r') as f:

    file = list(f)

    print(file[0])
```

```
Elena, 550
```

Example 4: Read the file you just created and make a Python list of:

- names
- scores

Solution 4a: loop

In [36]:
```
with open ('sample_data/scores.csv', 'r') as file:

    f = list(file)         # convert to list of strings (lines)

    names, scores = [], []

    for line in f:         # iterable: collect names and scores
        L = line.split()   # split() converts string (line) to
        names.append(L[0].strip(','))
        scores.append(L[1])

    print(names, scores)
```

```
['Elena', 'Sajid'] ['550', '480']
```

Solution 4b: list comprehension

In [37]:
```
with open('sample_data/scores.csv', 'r') as file:

    f = list(file)                # convert to list of strings


    L = [line.split() for line in f]  # list of lists
    print(L)
    names = [i[0].strip(',') for i in L]
    scores = [i[1] for i in L]

    print(names, scores)
```

```
[['Elena,', '550'], ['Sajid,', '480']]
['Elena', 'Sajid'] ['550', '480']
```

Example 5: change the first row to `'Mia, 700'` :

Solution 5

In [38]:

```
with open('sample_data/scores.csv', 'r+') as f:

    file = list(f)                          # convert to list of

    L = [line.split() for line in file]   # list of lists
    print(L)
    names = [i[0].strip(',') for i in L]  # names and scores
    scores = [i[1] for i in L]

    ###################################

    names[0] = 'Mia'
    scores[0] = '700'

    f.seek(0)

    for n, s in zip(names, scores):
        f.write(n + ' ' + s + '\n')

    f.truncate()


    f.seek(0)
    for line in f:
        print(line)
```

```
[['Elena,', '550'], ['Sajid,', '480']]
Mia 700

Sajid 480
```

In [ ]:

In [ ]: