# High-Level Programming Languages 3

## The Python Programming Language

Laszlo SZATHMARY
University of Debrecen
Faculty of Informatics

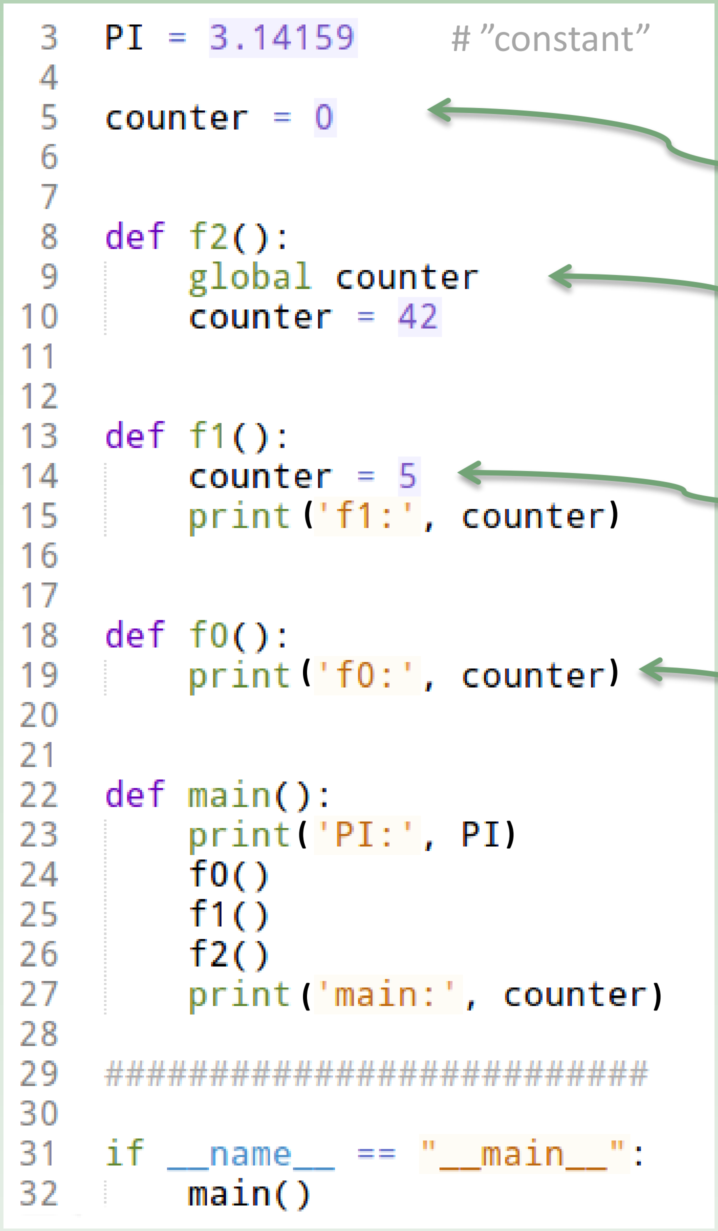
Lab #6

- global variables
- file handling

(last update: 2019-09-17 [yyyy-mm-dd])

2019-2020, 1st semester

## global variables



```python
PI = 3.14159      # ”constant”

counter = 0


def f2():
    global counter
    counter = 42


def f1():
    counter = 5
    print('f1:', counter)


def f0():
    print('f0:', counter)


def main():
    print('PI:', PI)
    f0()
    f1()
    f2()
    print('main:', counter)

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

if __name__ == "__main__":
    main()
```

**Convention:** ”constants” are written with capital letters

global variables

A global variable can be modified, but then we must use the „global” keyword.

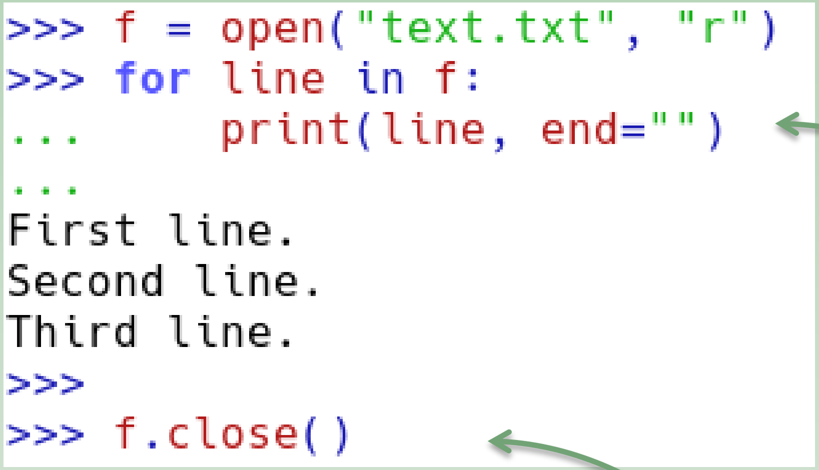
*counter* is a local variable here, which hides the global *counter*

Global variables are visible from every function. By default, their values cannot be changed.

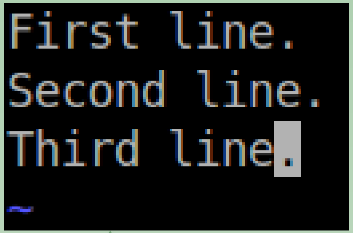
Reduce the number of global variables that you modify.

If something is a constant, then try NOT to modify it.

## reading from a file



```
>>> f = open("text.txt", "r")
>>> for line in f:
...     print(line, end="")
...
First line.
Second line.
Third line.
>>>
>>> f.close()
```

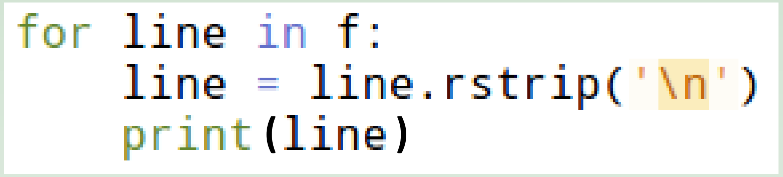


```
First line.
Second line.
Third line.
~
```

`text.txt`

`'\n'` is part of a line

*or:*

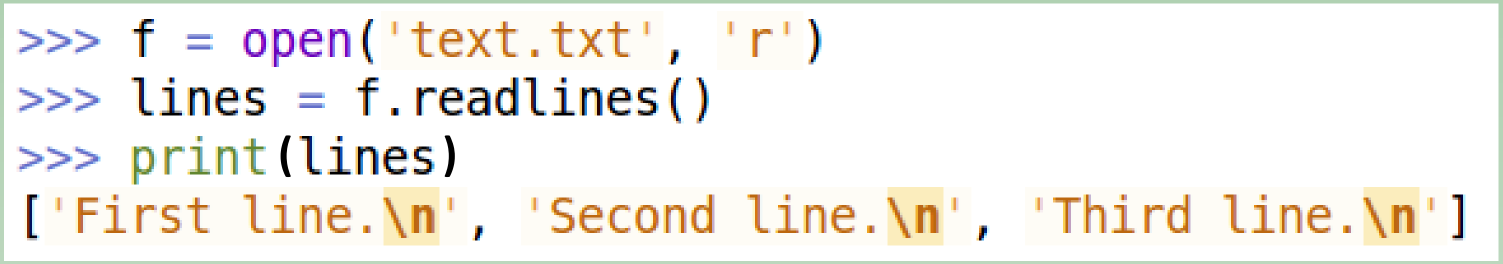


```
for line in f:
    line = line.rstrip('\n')
    print(line)
```

Don't forget to **close** the file!

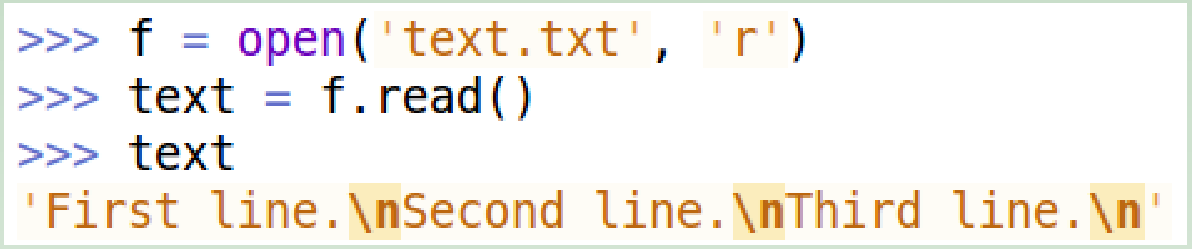
Opening modes:

| | | |
|---|---|---|
| r | -- | read |
| w | -- | write |
| a | -- | append |



```
>>> f = open('text.txt', 'r')
>>> lines = f.readlines()
>>> print(lines)
['First line.\n', 'Second line.\n', 'Third line.\n']
```

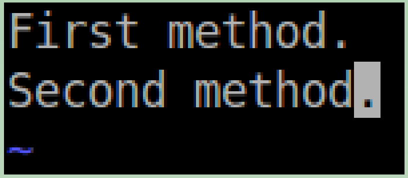
It reads the whole file and the lines are returned in a list.
`'\n'` is still part of the lines.



```
>>> f = open('text.txt', 'r')
>>> text = f.read()
>>> text
'First line.\nSecond line.\nThird line.\n'
```

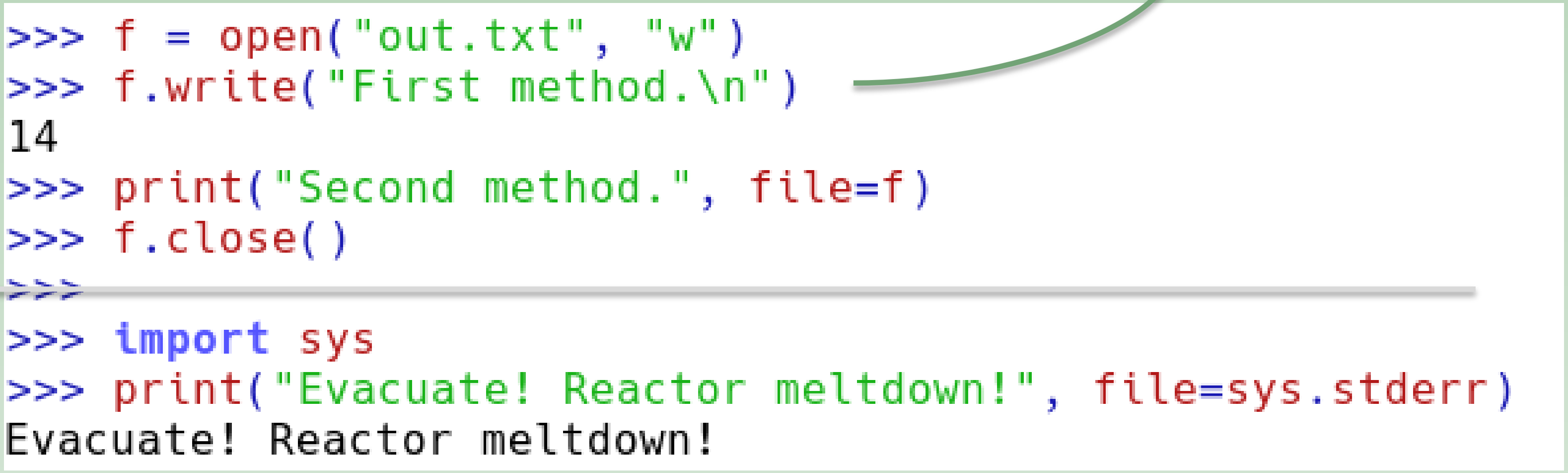
It reads the whole file and the content of the file is returned in a string.

Question: which method(s) to use in the case of large files?



```
First method.
Second method.
~
```

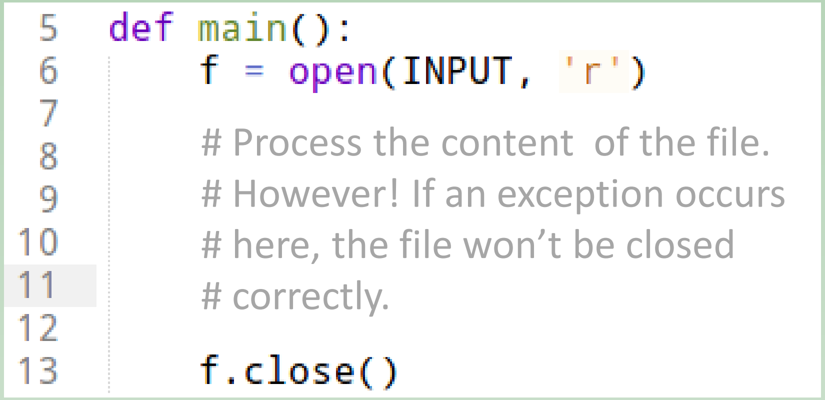
## writing to a file



```
>>> f = open("out.txt", "w")
>>> f.write("First method.\n")
14
>>> print("Second method.", file=f)
>>> f.close()
>>>
>>> import sys
>>> print("Evacuate! Reactor meltdown!", file=sys.stderr)
Evacuate! Reactor meltdown!
```

Use this method if you want to write to the standard error.

Old method:

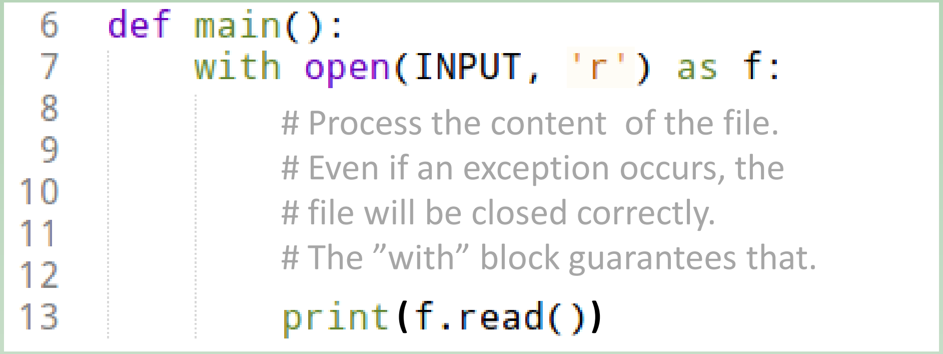


```python
def main():
    f = open(INPUT, 'r')

    # Process the content  of the file.
    # However! If an exception occurs
    # here, the file won’t be closed
    # correctly.

    f.close()
```

Modern method:

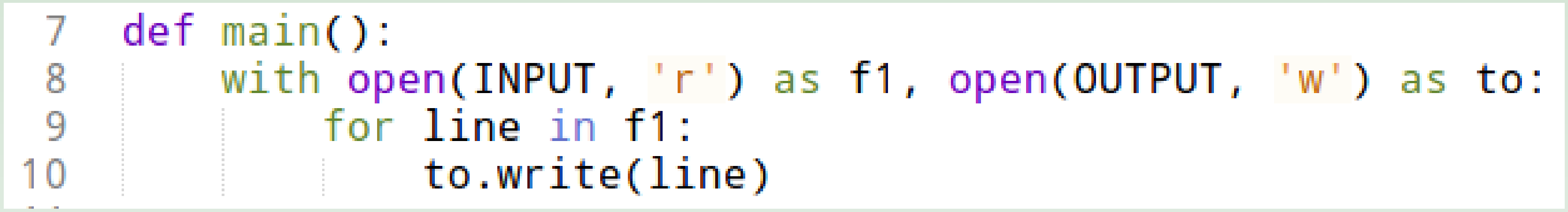


```python
def main():
    with open(INPUT, 'r') as f:
        # Process the content  of the file.
        # Even if an exception occurs, the
        # file will be closed correctly.
        # The ”with” block guarantees that.
        print(f.read())
```

There is no need to explicitly call f.close() .

Example: creating a copy of a text file.

from Python 2.7



```python
def main():
    with open(INPUT, 'r') as f1, open(OUTPUT, 'w') as to:
        for line in f1:
            to.write(line)
```

**Exercise:** rewrite this example using the old method.

## Exercise

Remove the comments from the file `string1.py` .
For the sake of simplicity, just remove the lines that start with a '`#`' symbol.
Write the output to a file called `string1_clean.py` .

Link: https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=EnPy3.20121006d homework

# Exercises #1

1. [20121006d] file handling (removing comments)
2. [20120818h] one hundred 50-digit long numbers (PE #13) [version **B**]
3. [20130218c] character count
4. [20120818g] largest product of five adjacent digits (PE #8)
5. [20130211a] anagram
6. [20130919b] a-z; reversed: z-a (Ouch! Requires thinking!)
7. [20130902e] parentheses
8. [20130902b] Hamming distance

☐ homework

# Exercises #2

1. [20120815g] PI verse (*list comprehension*-nel)
2. [20120818i] sum of digits (PE #16)
3. [20130902c] sentence without extra spaces