



# Programming 1

Laszlo SZATHMARY  
University of Debrecen  
Faculty of Informatics

## Lecture #6

- global variables
- file handling

(last update: 2024-01-31 [yyyy-mm-dd])

2023-2024, 2nd semester



# global variables

```
3 PI = 3.14159 # "constant"
4
5 counter = 0
6
7
8 def f2():
9     global counter
10    counter = 42
11
12
13 def f1():
14    counter = 5
15    print('f1:', counter)
16
17
18 def f0():
19    print('f0:', counter)
20
21
22 def main():
23    print('PI:', PI)
24    f0()
25    f1()
26    f2()
27    print('main:', counter)
28
29 #####
30
31 if __name__ == "__main__":
32    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)
```

Opening modes:

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

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



```
>>> 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?

```
>>> f = open("text.txt", "r")
>>> lines = f.read().splitlines()
>>> f.close()
>>>
>>> lines
['First line.', 'Second line.', 'Third line.']
```

Split text to individual lines.

```
>>> html
'<html>\n<body>\n...\n</body>\n</html>'
>>> print(html)
<html>
<body>
...
</body>
</html>
>>> html.splitlines()
['<html>', '<body>', '...', '</body>', '</html>']
>>> html.split("\n")
['<html>', '<body>', '...', '</body>', '</html>']
```

```
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:

```
5 def main():
6     f = open(INPUT, 'r')
7
8     # Process the content of the file.
9     # However! If an exception occurs
10    # here, the file won't be closed
11    # correctly.
12
13    f.close()
```

Modern method:

```
6 def main():
7     with open(INPUT, 'r') as f:
8
9         # Process the content of the file.
10        # Even if an exception occurs, the
11        # file will be closed correctly.
12        # The "with" block guarantees that.
13
14        print(f.read())
```

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

Example: creating a copy of a text file.

from Python 2.7

```
7 def main():
8     with open(INPUT, 'r') as f1, open(OUTPUT, 'w') as to:
9         for line in f1:
10            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>





# 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

# Exercises #2



homework



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