



Introduction to Programming

Lesson 4

Outline

- Strings revisited
- Formatting
- File I/O
 - read
 - process
 - write
 - close

Lesson Code

<http://goo.gl/N42N0h>

<http://www.pythonweekly.com>

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

```
>>> excuse = 'I am sick'  
>>> excuse = "I am sick"  
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>>
```


String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>> excuse = "I'm "sick""
SyntaxError: invalid syntax
>>> excuse = 'I'm "sick"'
SyntaxError: invalid syntax
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

Escape sequence \' or \"

=> ընդունիր որպես string-ի մաս

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>> excuse = "I'm "sick""
SyntaxError: invalid syntax
>>> excuse = 'I'm "sick"'
SyntaxError: invalid syntax
>>> excuse = 'I\'m "sick"'
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

Escape sequence \' or \"

=> ընդունիր որպես string-ի մաս

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>> excuse = "I'm "sick""
SyntaxError: invalid syntax
>>> excuse = 'I'm "sick"'
SyntaxError: invalid syntax
>>> excuse = 'I\'m "sick"'
>>> excuse
'I\'m "sick"'
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

Escape sequence \ ' or \ "

=> ընդունիր որպես string-ի մաս

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>> excuse = "I'm "sick""
SyntaxError: invalid syntax
>>> excuse = 'I'm "sick"'
SyntaxError: invalid syntax
>>> excuse = 'I\'m "sick"'
>>> excuse
'I\'m "sick"'
>>> print(excuse)
I'm "sick"
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

Escape sequence \ ' or \ "

=> ընդունիր որպես string-ի մաս

Another example:

- \n : escape sequence
(նշանակում է նոր տող)

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>> excuse = "I'm "sick""
SyntaxError: invalid syntax
>>> excuse = 'I'm "sick"'
SyntaxError: invalid syntax
>>> excuse = 'I\'m "sick"'
>>> excuse
'I\'m "sick"'
>>> print(excuse)
I'm "sick"
>>> excuse = 'I\'m ... \n ...
"sick"'
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' ' ' | " " " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

Escape sequence \ ' or \"

=> ընդունիր որպես string-ի մաս

Another example:

- \n : escape sequence
(նշանակում է նոր տող)

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>> excuse = "I'm "sick""
SyntaxError: invalid syntax
>>> excuse = 'I'm "sick"'
SyntaxError: invalid syntax
>>> excuse = 'I\'m "sick"'
>>> excuse
'I\'m "sick"'
>>> print(excuse)
I'm "sick"
>>> excuse = 'I\'m ...\n...
"sick"'
>>> excuse
'I\'m ...\n... "sick"'
>>>
```

String representations

string ==

տառերի հաջորդականություն

| " " | ' ' | ' ' ' ' | " " " " | մեջ

Quotes : single (') or double (")

Եթե (') or (") string-ի մաս էն?

Escape sequence \ ' or \"

=> ընդունիր որպես string-ի մաս

Another example:

- \n : escape sequence
(նշանակում է նոր տող)

```
>>> excuse = 'I am sick'
>>> excuse = "I am sick"
>>> excuse = 'I'm sick'
SyntaxError: invalid syntax
>>> excuse = "I'm sick"
>>> excuse = "I'm "sick""
SyntaxError: invalid syntax
>>> excuse = 'I'm "sick"'
SyntaxError: invalid syntax
>>> excuse = 'I\'m "sick"'
>>> excuse
'I\'m "sick"'
>>> print(excuse)
I'm "sick"
>>> excuse = 'I\'m ...\n...
"sick"'
>>> excuse
'I\'m ...\n... "sick"'
>>> print(excuse)
I'm ...
... "sick"
```

Exercise 1

- `interp` string
 - `'"What's up guys" said Robert'`

Exercise 1

- `nlǎǔǔ string`
 - `' "What's up guys" said Robert'`
- `υυǔǔδǔǔ myquotes.py` file

Exercise 1

- `ունեք` string
 - `' "What's up guys" said Robert'`
- `ստեղծեք` `myquotes.py` file
- `և print` արեք հետևյալ կոմբինացիաները
 - `' եզրերին, " ներսում, e.g. print(' " Hi " ')`
 - `' ամեն տեղ, e.g. print('\Hi\')`
 - `" ամեն տեղ, e.g. print("\ Hi\")`
 - `" " " եզրերին,`

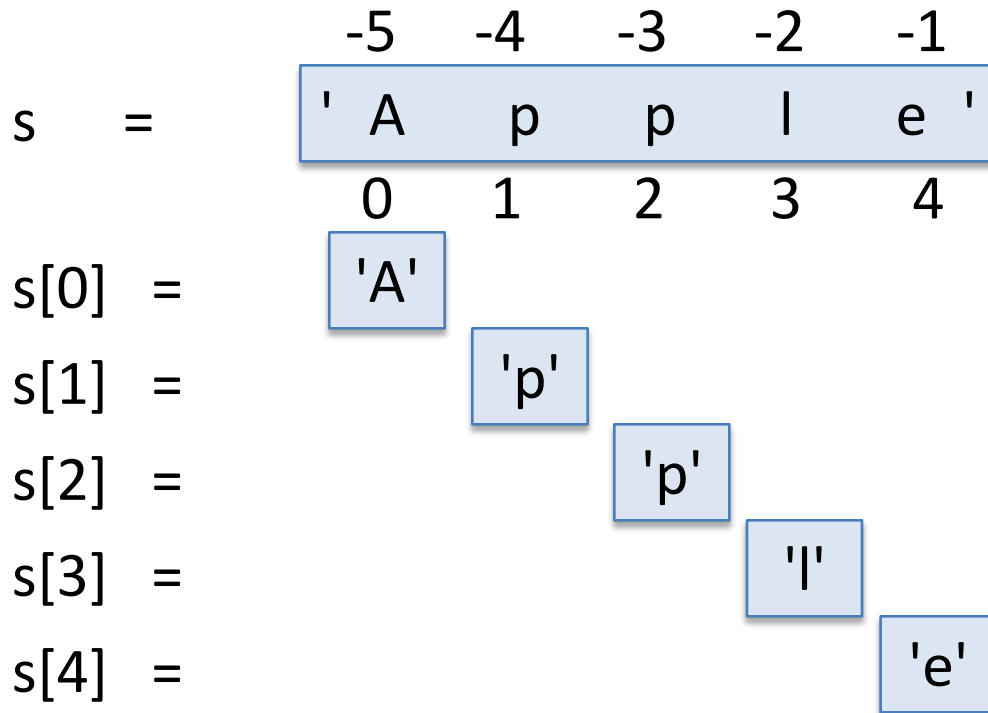
Exercise 1

```
s1 = '"What\'s up guys" said Robert'  
s2 = "'What\'s up guys' said Robert"  
s3 = '\''What\'s up guys\' said Robert'  
s4 = """\'What\'s up guys\' said Robert"""  
print('\n',s1, '\n',s2, '\n',s3, '\n',s4)
```

myquotes.py

Strings : Indexing operator

indexing operator - վերադարձնում է տառը
տվյալ index-ում- **s[i]**



Strings : Indexing operator

s =

-5	-4	-3	-2	-1		
'	A	p	p	l	e	'
0	1	2	3	4		

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

	-5	-4	-3	-2	-1		
s	'	A	p	p	l	e	'
=	0	1	2	3	4		

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

	-5	-4	-3	-2	-1	
s	=	' A p p l e '				
		0	1	2	3	4

s[0:2]	=	'A p'	
--------	---	-------	--

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

	-5	-4	-3	-2	-1
s =	' A p p l e '				
	0	1	2	3	4

s[0:2] =	'A p'	
----------	-------	--

s[1:4] =	'p p l'		
----------	---------	--	--

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

$s[i:j]$: s string-ի կտորը index i -ից
մինչև $j-1$ -ը

	-5	-4	-3	-2	-1
s	'	A	p	p	l
					e
					'
	0	1	2	3	4

$s[0:2]$	'	A	p	'
----------	---	---	---	---

$s[1:4]$	'	p	p	l	'
----------	---	---	---	---	---

$s[2:5]$	'	p	l	e	'
----------	---	---	---	---	---

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

$s[i:j]$: s string-ի կտորը index i -ից
մինչև $j-1$ -ը

$s[i:]$: s -ի կտորը index i -ից մինչև
վերջ

	-5	-4	-3	-2	-1
s	'	A	p	p	l
	e	'			
	0	1	2	3	4

$s[0:2]$	'	A	p	'
----------	---	---	---	---

$s[1:4]$	'	p	p	l	'
----------	---	---	---	---	---

$s[2:5]$	'	p	l	e	'
----------	---	---	---	---	---

$s[2:]$	'	p	l	e	'
---------	---	---	---	---	---

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

$s[i:j]$: s string-ի կտորը index i -ից
մինչև $j-1$ -ը

$s[i:]$: s -ի կտորը index i -ից մինչև
վերջ

$s[:j]$: s -ի կտորը մինչև index $j-1$ -ը

	-5	-4	-3	-2	-1		
s	'	A	p	p	l	e	'
	0	1	2	3	4		

$s[0:2]$	'	A	p	'
----------	---	---	---	---

$s[1:4]$	'	p	p	l	'
----------	---	---	---	---	---

$s[2:5]$	'	p	l	e	'
----------	---	---	---	---	---

$s[2:]$	'	p	l	e	'
---------	---	---	---	---	---

$s[:2]$	'	A	p	'
---------	---	---	---	---

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

$s[i:j]$: s string-ի կտորը index i -ից
մինչև $j-1$ -ը

$s[i:]$: s -ի կտորը index i -ից մինչև
վերջ

$s[:j]$: s -ի կտորը մինչև index $j-1$ -ը

	-5	-4	-3	-2	-1		
s	'	A	p	p	l	e	'
	0	1	2	3	4		
$s[0:2]$	'A p'						
$s[1:4]$			'p p l'				
$s[2:5]$						'p l e'	
$s[2:]$						'p l e'	
$s[:2]$	'A p'						
$s[-3:-1]$			'p l'				

Strings : Indexing operator

indexing operator կարելի
օգտագործել տառերի
խումբ անջատելու համար

$s[i:j]$: s string-ի կտորը index i -ից
մինչև $j-1$ -ը

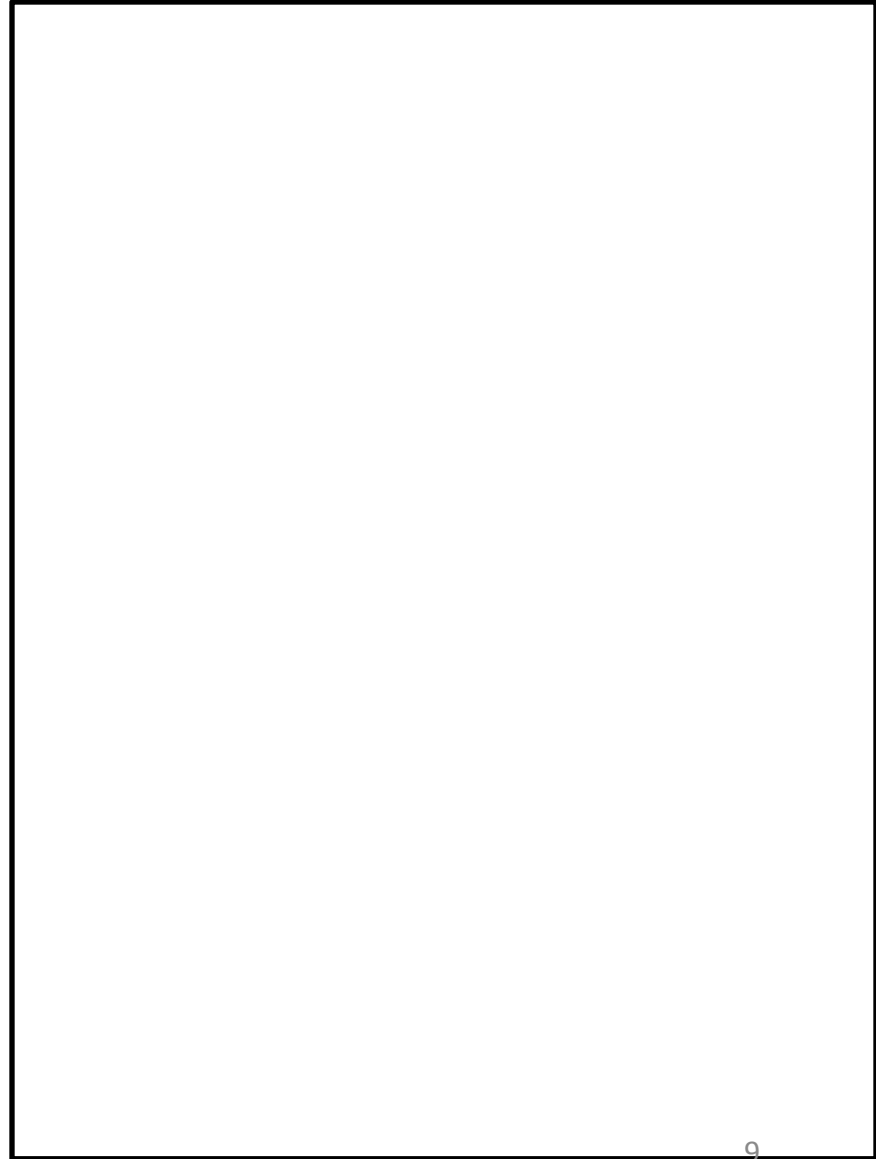
$s[i:]$: s -ի կտորը index i -ից մինչև
վերջ

$s[:j]$: s -ի կտորը մինչև index $j-1$ -ը

	-5	-4	-3	-2	-1		
s	'	A	p	p	l	e	'
	0	1	2	3	4		
$s[0:2]$	'A p'						
$s[1:4]$			'p p l'				
$s[2:5]$						'p l e'	
$s[2:]$						'p l e'	
$s[:2]$	'A p'						
$s[-3:-1]$			'p l'				

```
>>> s = 'Apple'
>>> s[0:2]
'Ap'
>>> s[1:4]
'ppl'
>>> s[2:5]
'ple'
>>> s[2:]
'ple'
>>> s[:2]
'Ap'
>>> s[-3:-1]
'pl'
```

Lists : Indexing operator



Lists : Indexing operator

```
>>> days = ['Sun', 'Mon',  
            'Tue', 'Wed', 'Thur',  
            'Fri', 'Sat']  
>>> middle_days =  
days[2:5]  
>>> middle_days  
['Tue', 'Wed', 'Thur']
```

Lists : Indexing operator

indexing operator
նույնն է list-երի և
string-երի համար

```
>>> days = ['Sun', 'Mon',  
            'Tue', 'Wed', 'Thur',  
            'Fri', 'Sat']  
>>> middle_days =  
days[2:5]  
>>> middle_days  
['Tue', 'Wed', 'Thur']
```


Lists : Indexing operator

indexing operator
նույնն է list-երի և
string-երի համար

```
>>> days = ['Sun', 'Mon',  
            'Tue', 'Wed', 'Thur',  
            'Fri', 'Sat']  
>>> middle_days =  
days[2:5]  
>>> middle_days  
['Tue', 'Wed', 'Thur']  
>>> numbers = [1, 2, 3, 4, 5]  
>>> print(numbers)  
[1, 2, 3, 4, 5]
```

Lists : Indexing operator

indexing operator
նույնն է list-երի և
string-երի համար

```
>>> days = ['Sun', 'Mon',  
            'Tue', 'Wed', 'Thur',  
            'Fri', 'Sat']  
>>> middle_days =  
days[2:5]  
>>> middle_days  
['Tue', 'Wed', 'Thur']  
>>> numbers = [1, 2, 3, 4, 5]  
>>> print(numbers)  
[1, 2, 3, 4, 5]  
>>> print(numbers[1:3])  
[2, 3]  
>>> print(numbers[:3])  
[1, 2, 3]  
>>> print(numbers[3:])  
[4, 5]
```

Lists : Indexing operator



Lists : Indexing operator

```
>>> numbers = [1,2,3,4,5]
>>> numbers_copy =
numbers[:]
>>> numbers_copy
[1, 2, 3, 4, 5]
```

Lists : Indexing operator

list-copy with `[:]`

```
>>> numbers = [1,2,3,4,5]
>>> numbers_copy =
numbers[:]
>>> numbers_copy
[1, 2, 3, 4, 5]
```

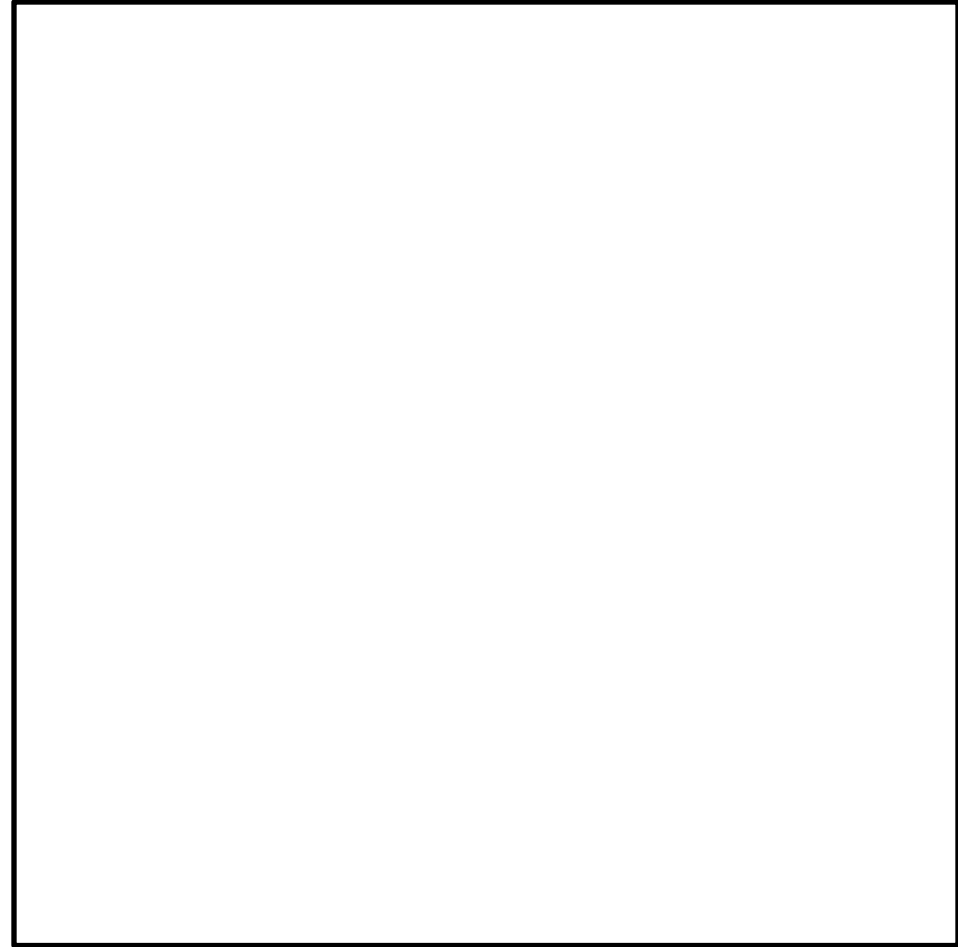
Lists : Indexing operator

list-ը copy անել `[:]`

հիշողության
տարբեր հասցեներ
=> տարբեր լիստեր

```
>>> numbers = [1,2,3,4,5]
>>> numbers_copy =
numbers[:]
>>> numbers_copy
[1, 2, 3, 4, 5]
>>> id(numbers_copy)
36380344
>>> id(numbers)
36375048
>>>
```

Lists : del operator



Lists : del operator

```
>>> numbers =  
[1,2,3,4,5,6,7,8,9,10]  
>>> numbers[-5:]  
[6, 7, 8, 9, 10]
```


Lists : del operator

list-ից անդամ ջնջելու
համար: **del operator**

```
>>> numbers =  
[1,2,3,4,5,6,7,8,9,10]  
>>> numbers[-5:]  
[6, 7, 8, 9, 10]  
>>> del numbers[2]
```

Lists : del operator

list-ից անդամ ջնջելու
համար: **del operator**

```
>>> numbers =  
[1,2,3,4,5,6,7,8,9,10]  
>>> numbers[-5:]  
[6, 7, 8, 9, 10]  
>>> del numbers[2]  
>>> numbers  
[1, 2, 4, 5, 6, 7, 8, 9, 10]
```

Lists : del operator

list-ից անդամ ջնջելու
համար: **del operator**

```
>>> numbers =  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
>>> numbers[-5:]  
[6, 7, 8, 9, 10]  
>>> del numbers[2]  
>>> numbers  
[1, 2, 4, 5, 6, 7, 8, 9, 10]  
>>> del numbers[1:3]  
>>> numbers  
[1, 5, 6, 7, 8, 9, 10]
```

Lists : del operator

list-ից անդամ ջնջելու
համար: **del operator**

դասարկ list:
del listname[:]

```
>>> numbers =  
[1,2,3,4,5,6,7,8,9,10]  
>>> numbers[-5:]  
[6, 7, 8, 9, 10]  
>>> del numbers[2]  
>>> numbers  
[1, 2, 4, 5, 6, 7, 8, 9, 10]  
>>> del numbers[1:3]  
>>> numbers  
[1, 5, 6, 7, 8, 9, 10]  
>>> del numbers[:]  
>>> numbers  
[]
```

Exercise 2

- Ստեղծեք հետևյալ լիստը

`lst = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']`

- օգտագործեք indexing operator-ը ստանալու համար

a) `['a', 'b', 'c', 'd']`

b) `['d', 'e', 'f']`

c) `['d']`

d) `['f', 'g']`

e) `['d', 'e', 'f', 'g', 'h']`

f) `['f', 'g', 'h']`

Exercise 2

- Ստեղծեք հետևյալ լիստը

```
lst = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

- օգտագործեք indexing operator-ը ստանալու համար

- a) ['a', 'b', 'c', 'd']
- b) ['d', 'e', 'f']
- c) ['d']
- d) ['f', 'g']
- e) ['d', 'e', 'f', 'g', 'h']
- f) ['f', 'g', 'h']

```
>>> lst[:4]
['a', 'b', 'c', 'd']
>>> lst[3:6]
['d', 'e', 'f']
>>> lst[3:4]
['d']
>>> lst[-3:-1]
['f', 'g']
>>> lst[3:]
['d', 'e', 'f', 'g', 'h']
>>> lst[-3:]
['f', 'g', 'h']
```

String methods

string-երը immutable են

=> method-ները չեն փոխում string-ը այլ return են անում copy-ն

Usage

Explanation

s.capitalize()

returns a **copy of s** with first character capitalized (առաջի տառը մեծատառ)

s.count(t)

returns the number of occurrences of t in s (քանի անգամ t-ն s-ի մեջ)

s.find(t)

returns the index of the first occurrence of t in s (գտիր t string-ի գտնվելու index-ը string-ի մեջ)

s.lower()

returns lowercase **copy of s** (բոլորը փոքրատառ)

String methods

string-երը immutable են

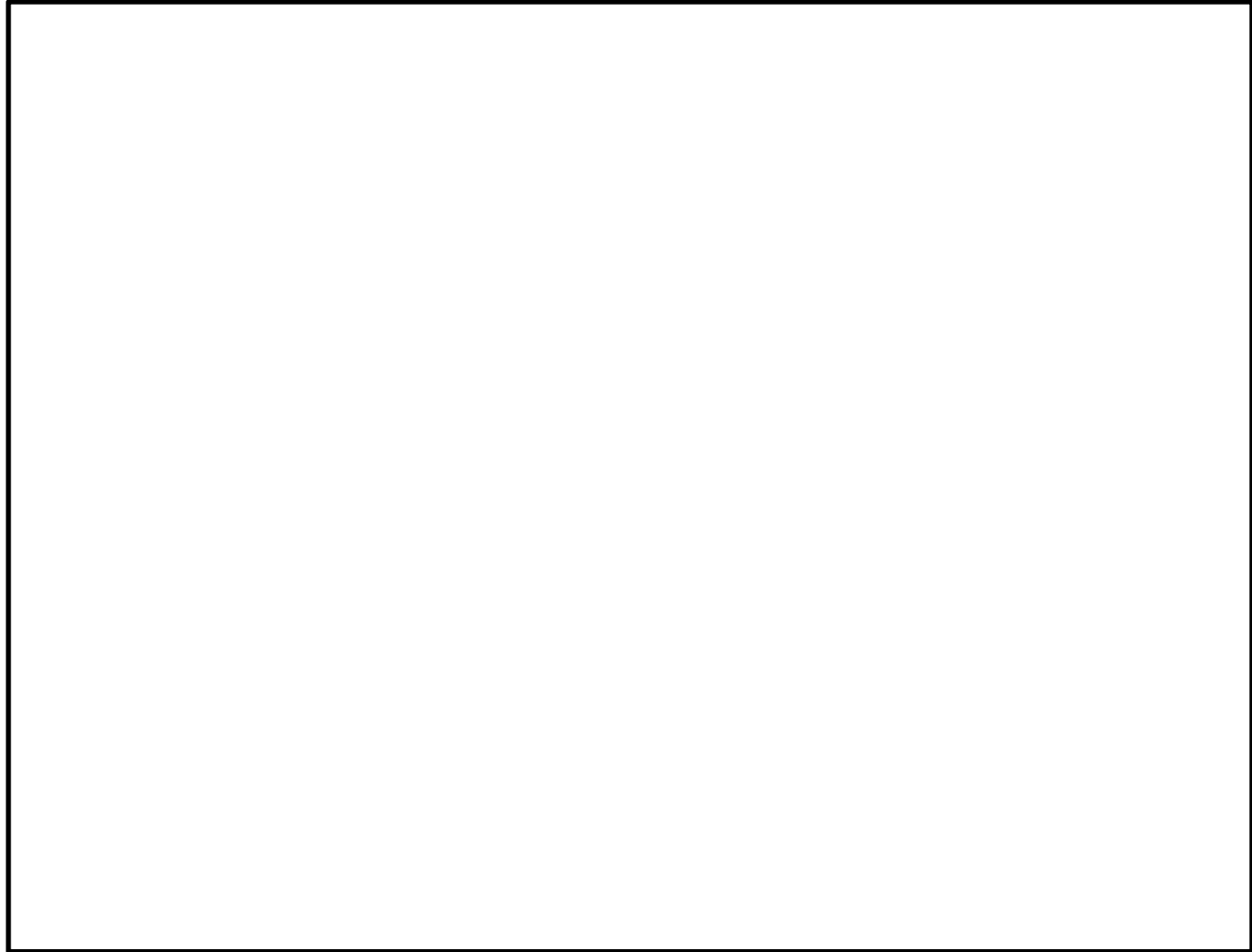
=> method-ները չեն փոխում string-ը այլ return են անում copy-ն

Usage

Explanation

s.replace(old, new)	returns copy of s with every occurrence of old replaced with new (փոխարինի բոլոր old string-երը new-ով)
s.split(sep)	returns list of substrings of s, delimited by sep (բաժանիք separator-ի միջոցով)
s.strip()	returns copy of s without leading and trailing whitespace (մաքրի s-ի սկզբի և վերջի պռաբելները)
s.upper()	returns all uppercase copy of s (լրիվ մեծատառ)

String methods



String methods

```
>>> link =  
'http://www.main.com/smith/index.html'  
>>> link[:4]  
'http'  
>>> link[:4].upper()  
'HTTP'  
>>> link.find('smith')  
20  
>>> link[20:25]  
'smith'  
>>> link[20:25].capitalize()  
'Smith'
```

String methods

method-ները չեն
փոխում string-ը
այլ return են
անում copy-ն

```
>>> link =  
'http://www.main.com/smith/index.html'  
>>> link[:4]  
'http'  
>>> link[:4].upper()  
'HTTP'  
>>> link.find('smith')  
20  
>>> link[20:25]  
'smith'  
>>> link[20:25].capitalize()  
'Smith'  
>>> link =  
'http://www.main.com/smith/index.html'
```

String methods

```
>>> link =  
'http://www.main.com/smith/index.html'  
>>> link.replace('smith', 'ferreira')  
'http://www.main.com/ferreira/index.html'  
>>> link  
'http://www.main.com/smith/index.html'
```

String methods

return են անույն copy-ն => վերագրի նոր փոփոխականի

```
>>> link =  
'http://www.main.com/smith/index.html'  
>>> link.replace('smith', 'ferreira')  
'http://www.main.com/ferreira/index.html'  
>>> link  
'http://www.main.com/smith/index.html'  
>>> new = link.replace('smith', 'ferreira')  
>>> new  
'http://www.main.com/ferreira/index.html'
```

String methods

return են անույն copy-ն => վերագրի նոր փոփոխականի

```
>>> link =  
'http://www.main.com/smith/index.html'  
>>> link.replace('smith', 'ferreira')  
'http://www.main.com/ferreira/index.html'  
>>> link  
'http://www.main.com/smith/index.html'  
>>> new = link.replace('smith', 'ferreira')  
>>> new  
'http://www.main.com/ferreira/index.html'  
>>> link.count('/')  
4  
>>> link.split('/')  
['http:', '', 'www.main.com', 'smith',  
'index.html']
```

Exercise 3

```
>>> events = '9/13 2:30 PM\n9/14 11:15  
AM\n9/14 1:00 PM\n9/15 9:00 AM'
```

```
>>> print(events)
```

```
9/13 2:30 PM
```

```
9/14 11:15 AM
```

```
9/14 1:00 PM
```

```
9/15 9:00 AM
```

Exercise 3

```
>>> events = '9/13 2:30 PM\n9/14 11:15 AM\n9/14 1:00 PM\n9/15 9:00 AM'
```

```
>>> print(events)
```

```
9/13 2:30 PM
```

```
9/14 11:15 AM
```

```
9/14 1:00 PM
```

```
9/15 9:00 AM
```

- a) Քանի հատ 9/14 կա
- b) 9/14 –ի առաջին index-ը
- c) Քանի պատահար կա 9/14-ին

```
>>> lst = events...
```

```
>>> lst
```

```
['9/14 11:15 AM',  
'9/14 1:00 PM']
```


Exercise 3

```
>>> events = '9/13 2:30 PM\n9/14 11:15 AM\n9/14 1:00 PM\n9/15 9:00 AM'
```

```
>>> print(events)
```

```
9/13 2:30 PM
```

```
9/14 11:15 AM
```

```
9/14 1:00 PM
```

```
9/15 9:00 AM
```

- a) Քանի հատ 9/14 կա
- b) 9/14 –ի առաջին index-ը
- c) Քանի պատահար կա 9/14-ին

```
>>> lst = events...
```

```
>>> lst
```

```
['9/14 11:15 AM',
```

```
'9/14 1:00 PM']
```

```
>>> events.count('9/14')
```

```
2
```

```
>>> events.find('9/14')
```

```
13
```

```
>>> events.find('9/15')
```

```
40
```

```
>>> events[13:40]
```

```
'9/14 11:15 AM\n9/14 1:00 PM\n'
```

```
>>> lst =
```

```
events[13:40].strip().split('\n')
```

```
>>> lst
```

```
['9/14 11:15 AM', '9/14 1:00 PM']
```

String methods

Ուզում են առանձնացնենք

string event-ի date և time կոմպոնենտները

```
>>> event = "Tuesday, Feb 29, 2012 -- 3:35 PM"
```

String methods

Ուզում են առանձնացնենք
string event-ի date և time կոմպոնենտները

```
>>> event = "Tuesday, Feb 29, 2012 -- 3:35 PM"
```

Solution: փոխարինի կետադրությունը պռաբելներով

Usage	Explanation
str.maketrans(old, new)	returns a table mapping characters in string old to characters in string new
s.translate(table)	returns a copy of s in which the original characters are replaced using the mapping described by table

String methods

Ուզում են առանձնացնենք
string event-ի date և time կոմպոնենտները

```
>>> event = "Tuesday, Feb 29, 2012 -- 3:35 PM"  
>>> table = str.maketrans(':', '-', 3*' ')
```

Solution: փոխարինի կետադրությունը պոաբելներով

Usage	Explanation
str.maketrans(old, new)	returns a table mapping characters in string old to characters in string new
s.translate(table)	returns a copy of s in which the original characters are replaced using the mapping described by table

String methods

Ուզում են առանձնացնենք
string event-ի date և time կոմպոնենտները

```
>>> event = "Tuesday, Feb 29, 2012 -- 3:35 PM"
>>> table = str.maketrans(':', '-', 3*' ')
>>> event.translate(table)
'Tuesday Feb 29 2012 3 35 PM'
```

Solution: փոխարինի կետադրությունը պռաբելներով

Usage	Explanation
str.maketrans(old, new)	returns a table mapping characters in string old to characters in string new
s.translate(table)	returns a copy of s in which the original characters are replaced using the mapping described by table

String methods

Ուզում են առանձնացնենք
string event-ի date և time կոմպոնենտները

```
>>> event = "Tuesday, Feb 29, 2012 -- 3:35 PM"
>>> table = str.maketrans(':', '-', 3*' ')
>>> event.translate(table)
'Tuesday Feb 29 2012 3 35 PM'
>>> event.translate(table).split()
['Tuesday', 'Feb', '29', '2012', '3', '35', 'PM']
```

Solution: փոխարինի կետադրությունը պռաբելներով

Usage	Explanation
str.maketrans(old, new)	returns a table mapping characters in string old to characters in string new
s.translate(table)	returns a copy of s in which the original characters are replaced using the mapping described by table

print()

Function `print()` վերցնում է 0 և ավելի արգումենտ և տպում նրանց **string representation**-ը

```
>>> prod = 'bread'
>>> cost = 2
>>> wght = 700
>>> total = cost*wght
>>> print(prod, cost, wght, total)
bread 2 700 1400
```

print()

Function print() վերցնում է 0 և ավելի արգումենտ և տպում նրանց **string representation**-ը

```
>>> prod = 'bread'
>>> cost = 2
>>> wght = 700
>>> total = cost*wght
>>> print(prod, cost, wght, total)
bread 2 700 1400
```

A blank space separator (պռաբել) տպվում է argument-ների միջև

print()

Function `print()` վերցնում է 0 և ավելի արգումենտ և տպում նրանց **string representation**-ը

```
>>> prod = 'bread'
>>> cost = 2
>>> wght = 700
>>> total = cost*wght
>>> print(prod, cost, wght, total)
bread 2 700 1400
>>> print(prod, cost, wght, total, sep=': ')
bread: 2: 700: 1400
>>> print(prod, cost, wght, total, sep=':::')
bread:::2:::700:::1400
```

A blank space separator (պիտանի) սեպարատոր է
argument-ների միջև

sep : allows for customized separators

(թույլ է տալիս հարմարեցված սեպարատորներ)

print()

Function `print()` argument-ներից հետո տպում է նոր տող “\n”

```
>>> pets = ['boa', 'cat', 'dog']  
>>> for pet in pets:  
    print(pet)
```

```
boa  
cat  
dog
```

print()

Function **print()** argument-ներից հետո տպում է նոր տող “\n”

```
>>> pets = ['boa', 'cat', 'dog']
```

```
>>> for pet in pets:  
    print(pet)
```

```
boa
```

```
cat
```

```
dog
```

```
>>> for pet in pets:  
    print(pet, end=', ')
```

```
boa, cat, dog,
```

```
>>> for pet in pets:  
    print(pet, end='!!! ')
```

```
boa!!! cat!!! dog!!!
```

```
>>>
```

end : allows for customized end characters

(թույլ է տալիս հարմարեցնել վերջավորությունը)

formatting output

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
```

formatting output

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
```

formatting output

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print(hour+' ':'+minute+' ':'+second)
Traceback (most recent call last):
...
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

formatting output

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print(hour+' ':'+minute+' ':'+second)
Traceback (most recent call last):
...
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> print(str(hour)+' ':'+str(minute)+' ':'+str(second))
11:45:33
```

formatting output

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print(hour+':'+minute+':'+second)
Traceback (most recent call last):
...
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> print(str(hour)+':'+str(minute)+':'+str(second))
11:45:33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
```


format() method of str (string)

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
```

format() method of str (string)

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
```

```
print('{}: {}: {}'.format(hour, minute, second))
```

format() method of str (string)

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
```

```
print('{}: {}: {}'.format(hour, minute, second))
```



placeholders

format() method of str (string)

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
```

```
print('{}: {}: {}'.format(hour, minute, second))
```

placeholders



format() method of str (string)

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
```

```
print('{}: {}: {}'.format(hour, minute, second))
```

placeholders

format() method of str (string)

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
```

```
print('{}: {}: {}'.format(hour, minute, second))
```

placeholders

format() method of str (string)

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}: {}: {}'.format(hour, minute, second))
11:45:33
>>> print('{} , {} {}, {} at {}: {}: {}'.format(weekday, month,
day, year, hour, minute, second))
Wednesday, March 10, 2012 at 11:45:33
```

placeholders



formatting field width

```
>>> for i in range(1,8):  
        print(i, i**2, 2**i)
```

```
1 1 2  
2 4 4  
3 9 8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```


formatting field width

`format()` method data-ի
սյուները
դասավորելու համար

```
>>> for i in range(1,8):  
        print(i, i**2, 2**i)
```

```
1 1 2  
2 4 4  
3 9 8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

formatting field width

`format()` method data-ի
սյուները
դասավորելու համար

```
>>> for i in range(1,8):  
        print(i, i**2, 2**i)
```

```
1 1 2  
2 4 4  
3 9 8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

```
>>> for i in range(1, 8):  
        print('{ } {:2} {:3}'.  
              format(i, i**2, 2**i))
```

```
1  1  2  
2  4  4  
3  9  8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

formatting field width

`format()` method data-ի
սյուները
դասավորելու համար

```
>>> for i in range(1,8):  
        print(i, i**2, 2**i)
```

```
1 1 2  
2 4 4  
3 9 8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

```
>>> for i in range(1, 8):  
        print('{ } {:2} {:3}'.  
              format(i, i**2, 2**i))
```

```
1  1  2  
2  4  4  
3  9  8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

reserve 2* ' '

formatting field width

`format()` method data-ի
սյուները
դասավորելու համար

```
>>> for i in range(1,8):  
        print(i, i**2, 2**i)
```

```
1 1 2  
2 4 4  
3 9 8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

```
>>> for i in range(1, 8):  
        print('{ } {:2} {:3}'.  
              format(i, i**2, 2**i))
```

```
1  1  2  
2  4  4  
3  9  8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

reserve 3* ' '

reserve 2* ' '

formatting field width

`format()` method data-ի
սյուները
դասավորելու համար

թվերը դասավորված
են դեպի աջ

```
>>> for i in range(1,8):  
        print(i, i**2, 2**i)
```

```
1 1 2  
2 4 4  
3 9 8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

```
>>> for i in range(1, 8):  
        print('{ } {:2} {:3}'.  
              format(i, i**2, 2**i))
```

```
1  1  2  
2  4  4  
3  9  8  
4 16 16  
5 25 32  
6 36 64  
7 49 128
```

reserve 3* ' '

reserve 2* ' '

formatting field width

`format()` method data-ի
սյուները
դասավորելու համար

```
>>> lst = ['Jeff Bezos', 'Tim Cook',  
           'Larry Ellison']  
>>> for name in lst:  
        fl = name.split()  
        print(fl[0], fl[1])
```

```
Jeff Bezos  
Tim Cook  
Larry Ellison
```

formatting field width

format() method data-ի
սյուները
դասավորելու համար

string-երը
դասավորված են
դեպի ձախ

```
>>> lst = ['Jeff Bezos', 'Tim Cook',  
           'Larry Ellison']  
>>> for name in lst:  
        fl = name.split()  
        print(fl[0], fl[1])  
  
Jeff Bezos  
Tim Cook  
Larry Ellison  
>>> for name in lst:  
        fl = name.split()  
        print('{:5} {:10}'.  
              format(fl[0], fl[1]))  
  
Jeff  Bezos  
Tim   Cook  
Larry Ellison  
>>>
```

output format type

`{ }`-ի մեջ կարող ենք նշել

1. field width (դաշտի լայնությունը)
2. output type (binary, decimal, hexadecimal)
3. decimal precision (տասնորդականի ճշտությունը)

Type	Explanation
b	binary
c	character
d	decimal
X	hexadecimal
e	scientific
f	fixed-point

```
>>> n = 10
>>> '{:b}'.format(n)
'1010'
>>> '{:c}'.format(n)
'\n'
>>> '{:d}'.format(n)
'10'
>>> '{:X}'.format(n)
'A'
>>> '{:e}'.format(n)
'1.000000e+01'
>>> '{:7.2f}'.format(n)
'  10.00'
>>>
```


output format type

{ }-ի մեջ կարող ենք նշել

1. field width (դաշտի լայնությունը)
2. output type (binary, decimal, hexadecimal)
3. decimal precision (տասնորդականի ճշտությունը)

Type	Explanation
b	binary
c	character
d	decimal
X	hexadecimal
e	scientific
f	fixed-point

```
>>> n = 10
>>> '{:b}'.format(n)
'1010'
>>> '{:c}'.format(n)
'\n'
>>> '{:d}'.format(n)
'10'
>>> '{:X}'.format(n)
'A'
>>> '{:e}'.format(n)
'1.000000e+01'
>>> '{:7.2f}'.format(n)
'  10.00'
>>>
```

'{:7.2f}'

output format type

{ }-ի մեջ կարող ենք նշել

1. field width (դաշտի լայնությունը)
2. output type (binary, decimal, hexadecimal)
3. decimal precision (տասնորդականի ճշտությունը)

Type	Explanation
b	binary
c	character
d	decimal
X	hexadecimal
e	scientific
f	fixed-point

```
>>> n = 10
>>> '{:b}'.format(n)
'1010'
>>> '{:c}'.format(n)
'\n'
>>> '{:d}'.format(n)
'10'
>>> '{:X}'.format(n)
'A'
>>> '{:e}'.format(n)
'1.000000e+01'
>>> '{:7.2f}'.format(n)
'  10.00'
>>>
```

'{:7.2f}'

field width

output format type

`{ }`-ի մեջ կարող ենք նշել

1. field width (դաշտի լայնությունը)
2. output type (binary, decimal, hexadecimal)
3. decimal precision (տասնորդականի ճշտությունը)

Type	Explanation
b	binary
c	character
d	decimal
X	hexadecimal
e	scientific
f	fixed-point

```
>>> n = 10
>>> '{:b}'.format(n)
'1010'
>>> '{:c}'.format(n)
'\n'
>>> '{:d}'.format(n)
'10'
>>> '{:X}'.format(n)
'A'
>>> '{:e}'.format(n)
'1.000000e+01'
>>> '{:7.2f}'.format(n)
'  10.00'
>>>
```

output type

'{:7.2f}'

field width

output format type

{ }-ի մեջ կարող ենք նշել

1. field width (դաշտի լայնությունը)
2. output type (binary, decimal, hexadecimal)
3. decimal precision (տասնորդականի ճշտությունը)

Type	Explanation
b	binary
c	character
d	decimal
X	hexadecimal
e	scientific
f	fixed-point

```
>>> n = 10
>>> '{:b}'.format(n)
'1010'
>>> '{:c}'.format(n)
'\n'
>>> '{:d}'.format(n)
'10'
>>> '{:X}'.format(n)
'A'
>>> '{:e}'.format(n)
'1.000000e+01'
>>> '{:7.2f}'.format(n)
'  10.00'
>>>
```

output type

' { : 7 . 2 f } '

field width

decimal precision

Absolute and Relative paths

Absolute and Relative paths

- Absolute path to a folder (Բացարձակ ճանապարհը դեպի python folder)

WIN- `C:\Users\Training\Desktop\python`

LIN, MAC- `/home/Training/Desktop/python`

Absolute and Relative paths

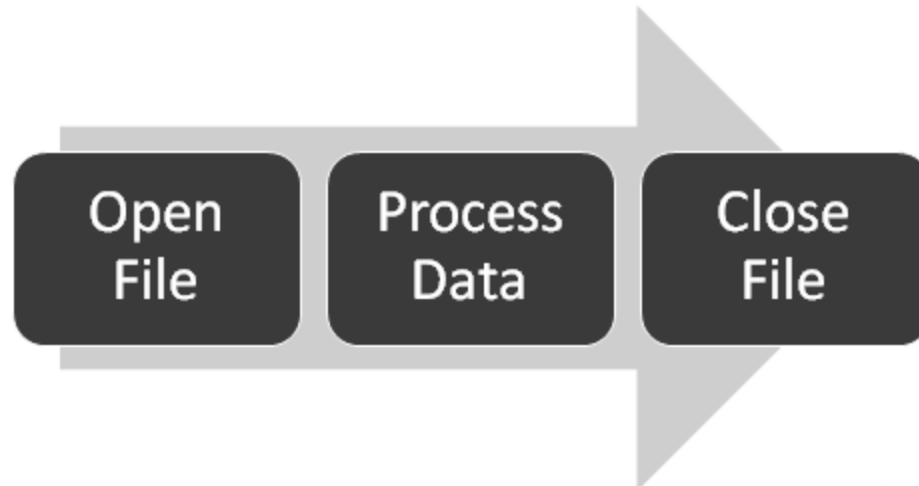
- Absolute path to a folder (Բացարձակ ճանապարհը դեպի python folder)
WIN- `C:\Users\Training\Desktop\python`
LIN, MAC- `/home/Training/Desktop/python`
- Absolute path to a file (Բացարձակ ճանապարհը դեպի file)
WIN- `C:\Users\Training\Desktop\python\data.txt`
LIN, MAC- `/home/Training/Desktop/python/data.txt`

Absolute and Relative paths

- Absolute path to a folder (Բացարձակ ճանապարհը դեպի python folder)
WIN- `C:\Users\Training\Desktop\python`
LIN, MAC- `/home/Training/Desktop/python`
- Absolute path to a file (Բացարձակ ճանապարհը դեպի file)
WIN- `C:\Users\Training\Desktop\python\data.txt`
LIN, MAC- `/home/Training/Desktop/python/data.txt`
- Relative to current working directory (Desktop) (համեմատած ընթացիկ աշխատանքային դիրեկտորիայի հետ)
WIN- `\python\data.txt`
LIN, MAC- `./python/data.txt`

open/close a file

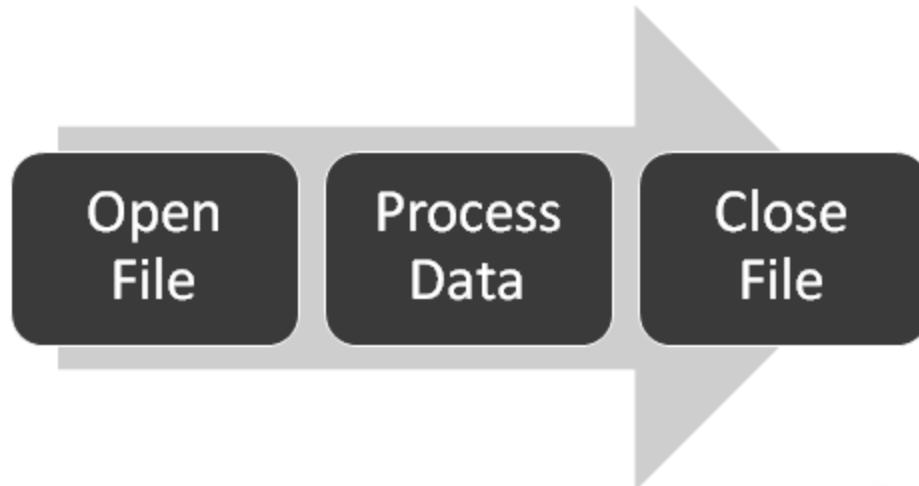
file-ի հետ աշխատելու համար պետք է



open/close a file

file-ի հետ աշխատելու համար պետք է

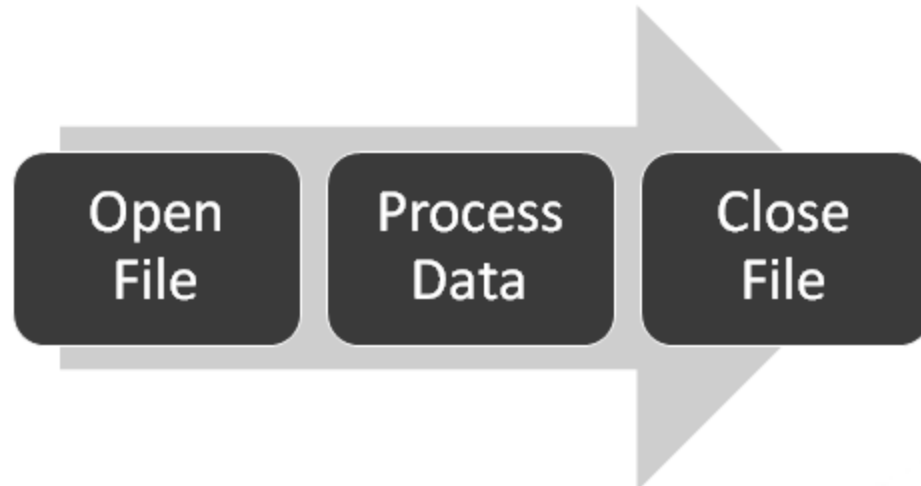
1. Open the file (բացել file-ը)



open/close a file

file-ի հետ աշխատելու համար պետք է

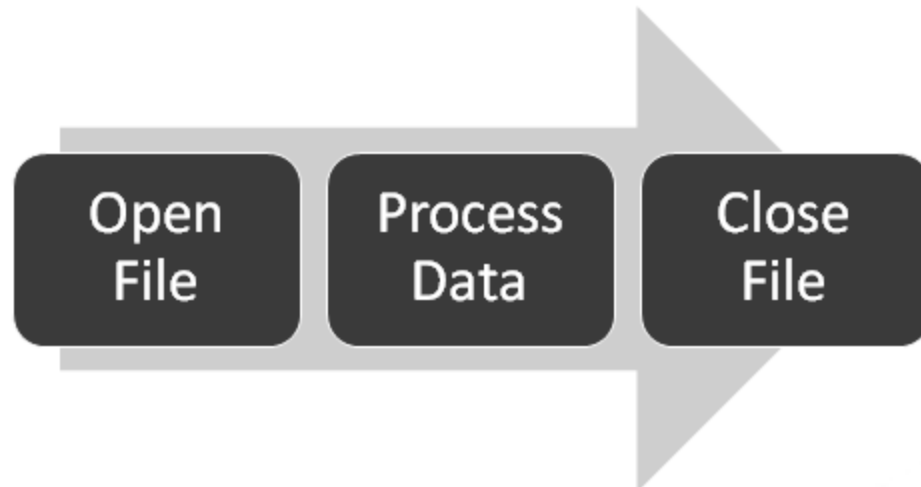
1. Open the file (բացել file-ը)
2. Read/Write file (կարդալ/գրել file-ը)



open/close a file

file-ի հետ աշխատելու համար պետք է

1. Open the file (բացել file-ը)
2. Read/Write file (կարդալ/գրել file-ը)
3. Close the file (փակել file-ը)



open/close a file

open/close a file

```
infile = open("filename")
```

file-ը բացելու համար օգտագործեք open() function-ը

open/close a file

```
infile = open("filename")
```

file-ը բացելու համար օգտագործեք open() function-ը
open() function-ի առաջին argument-ը file-ի անունն է,
որը բացարձակ կամ համեմատատական
ճանապարհն է դեպի file

open/close a file

```
infile = open("filename")
```

```
infile = open("filename", 'r')
```

file-ը բացելու համար օգտագործեք open() function-ը

open() function-ի առաջին argument-ը file-ի անունն է, որը բացարձակ կամ համեմատատական ճանապարհն է դեպի file

open/close a file

File mode **'r'** : read file

```
infile = open("filename")  
infile = open("filename", 'r')
```

The second (**optional**) argument is the **file mode**
(երկրորդ argument-ը file-ի ռեժիմն է)

file-ը բացելու համար օգտագործեք open() function-ը
open() function-ի առաջին argument-ը file-ի անունն է,
որը բացարձակ կամ համեմատատական
ճանապարհն է դեպի file

open/close a file

File mode **'r'** : read file

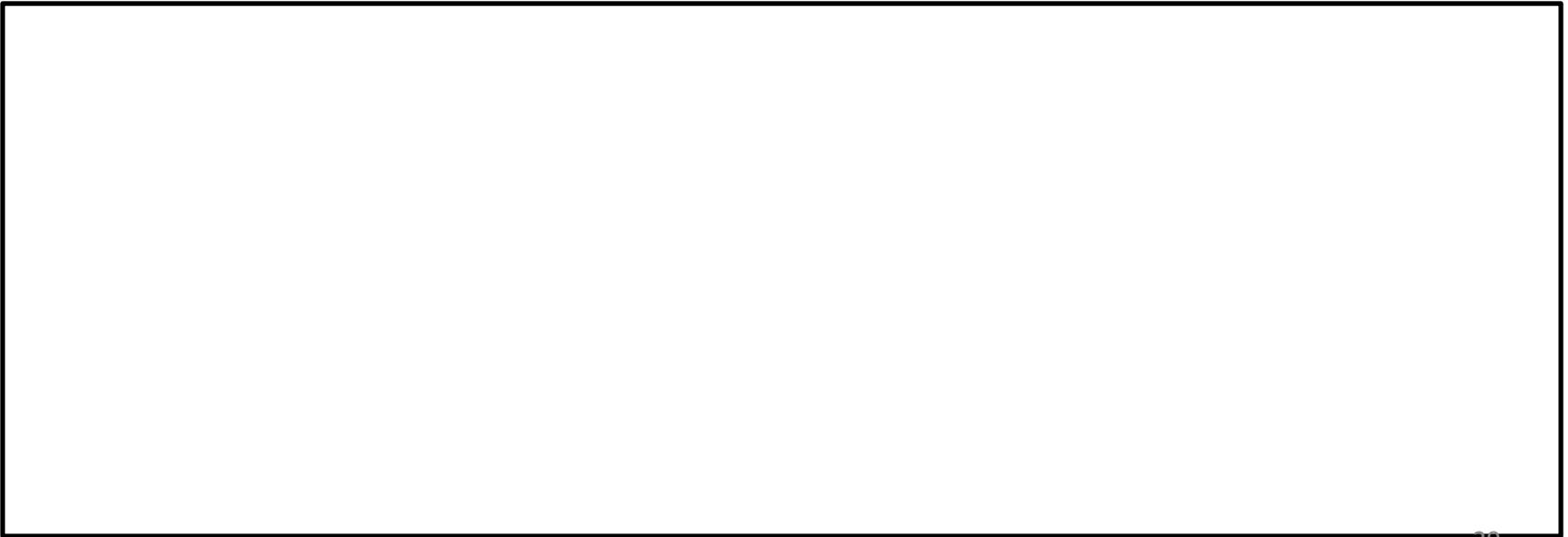
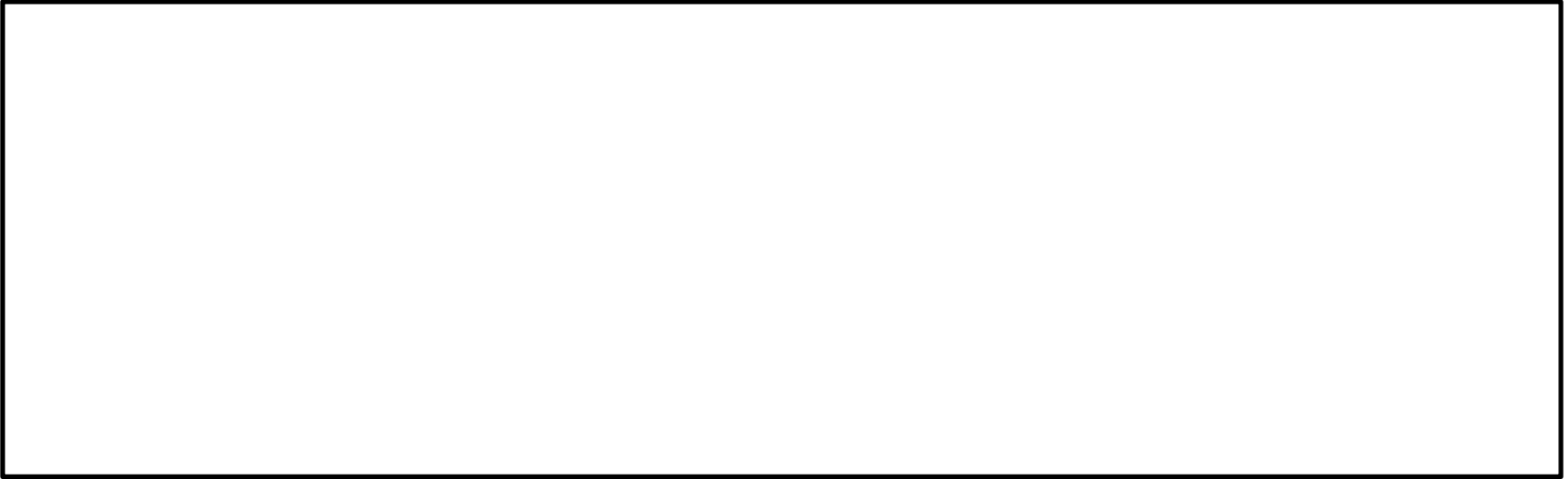
```
infile = open("filename")  
infile = open("filename", 'r')
```

open() returns a **"file"** object argument is the **file mode**
վերադարձնում է **"file"** object և-ի տեղիքն է)

file-ը բացելու համար օգտագործեք open() function-ը

open() function-ի առաջին argument-ը file-ի անունն է,
որը բացարձակ կամ համեմատատական
ճանապարհն է դեպի file

open/close a file



open/close a file

```
>>> infile = open('data.txt')
```

open/close a file

```
>>> infile = open('data.txt')
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    infile = open('data.txt')
IOError: [Errno 2] No such file or directory: 'data.txt'
```

open/close a file

```
>>> infile = open('data.txt')
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    infile = open('data.txt')
IOError: [Errno 2] No such file or directory: 'data.txt'
>>> infile = open(
r"C:\Users\Training\Desktop\python\data.txt", 'r')
```

open/close a file

```
>>> infile = open('data.txt')
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    infile = open('data.txt')
IOError: [Errno 2] No such file or directory: 'data.txt'
>>> infile = open(
r"C:\Users\Training\Desktop\python\data.txt", 'r')
>>> infile.close()
```

open/close a file

```
>>> infile = open('data.txt')
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    infile = open('data.txt')
IOError: [Errno 2] No such file or directory: 'data.txt'
>>> infile = open(
r"C:\Users\Training\Desktop\python\data.txt", 'r')
>>> infile.close()
```

```
>>> directory = r"C:\Users\Training\Desktop\python"
```


open/close a file

```
>>> infile = open('data.txt')
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    infile = open('data.txt')
IOError: [Errno 2] No such file or directory: 'data.txt'
>>> infile = open(
r"C:\Users\Training\Desktop\python\data.txt", 'r')
>>> infile.close()
```

```
>>> directory = r"C:\Users\Training\Desktop\python"
>>> import os
>>> os.getcwd()
'C:\\Python34'
```

open/close a file

```
>>> infile = open('data.txt')
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    infile = open('data.txt')
IOError: [Errno 2] No such file or directory: 'data.txt'
>>> infile = open(
r"C:\Users\Training\Desktop\python\data.txt", 'r')
>>> infile.close()
```

```
>>> directory = r"C:\Users\Training\Desktop\python"
>>> import os
>>> os.getcwd()
'C:\\Python34'
>>> os.chdir(directory)
>>> os.getcwd()
'C:\\Users\\Training\\Desktop\\python'
```

open/close a file

```
>>> infile = open('data.txt')
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    infile = open('data.txt')
IOError: [Errno 2] No such file or directory: 'data.txt'
>>> infile = open(
r"C:\Users\Training\Desktop\python\data.txt", 'r')
>>> infile.close()
```

```
>>> directory = r"C:\Users\Training\Desktop\python"
>>> import os
>>> os.getcwd()
'C:\\Python34'
>>> os.chdir(directory)
>>> os.getcwd()
'C:\\Users\\Training\\Desktop\\python'
>>> infile = open('data.txt')
>>> infile.close()
```

open file mode

file mode: սահմանում է file-ից օգտվելու ձևը

open file mode

file mode: սահմանում է file-ից օգտվելու ձևը

Mode	Description
r	Reading (default), կարդալ
w	Writing (if file exists, content is wiped), գրել
a	Append (if file exists, writes are appended) Ավելացնել file-ին եթե գոյություն ունի
r+	Reading and Writing, կարդալ և գրել
t	Text (default)
b	Binary

open file mode

file mode: սահմանում է file-ից օգտվելու ձևը

Mode	Description
r	Reading (default), կարդալ
w	Writing (if file exists, content is wiped), գրել
a	Append (if file exists, writes are appended) Ավելացնել file-ին եթե գոյություն ունի
r+	Reading and Writing, կարդալ և գրել
t	Text (default)
b	Binary

```
>>> infile = open('data.txt', 'r')  
>>> infile = open('data.txt', 'w')  
>>> infile = open('data.txt', 'r+')
```

file methods

file methods

```
>>> infile = open('data.txt', 'r+')
```


file methods

```
>>> infile = open('data.txt', 'r+')
```

'infile' file object-ի method-ներն են
`read()` and `readline()` : վերադարձնում է string

file methods

```
>>> infile = open('data.txt', 'r+')
```

'infile' file object-ի method-ներն են

read() and **readline()** : վերադարձնում է string

readlines() : վերադարձնում է տողերի list
(տողերը string են)

file methods

```
>>> infile = open('data.txt', 'r+')
```

'infile' file object-ի method-ներն են

read() and **readline()** : վերադարձնում է string

readlines() : վերադարձնում է տողերի list
(տողերը string են)

write() : վերադարձնում է գրված տառերի
քանակը

file methods

```
>>> infile = open('data.txt', 'r')
```

Usage	Description
<code>infile.read()</code>	Read starting from cursor up to the end of the file (կարդա cursor-ից մինչև վերջ)
<code>infile.readline()</code>	Read starting from cursor up to, and including, the end of line character (կարդա ամբողջ տողը + '\n')
<code>infile.readlines()</code>	Read starting from cursor up to the end of the file and return list of lines. (կարդա և վերադարձրա տողերի լիստ)
<code>outfile.write(s)</code>	Write string <code>s</code> to file <code>outfile</code> starting from cursor (գրի <code>s</code> string-ը <code>outfile</code> -ի մեջ)
<code>infile.close()</code>	Close file <code>infile</code> (փակիր file-ը)

reading a file

```
1 The 3 lines in this file end with the new line character.\n2 \n3 There is a blank line above this line.\n
```

example.txt

reading a file

```
1 The 3 lines in this file end with the new line character.\n
2 ^\n
3 There is a blank line above this line.\n
```

example.txt

Երբ file-ը բացած է, cursor-ը կապված է file-ի հետ

cursor-ի

սկզբնական դիրքը:

- file-ի սկզբում
 - if file mode is 'r'
- file-ի վերջում
 - if file mode is 'a'('w')

```
>>> infile = open('example.txt')
>>>
```

reading a file

```
1 The 3 lines in this file end with the new line character.\n
2 ^\n
3 There is a blank line above this line.\n
```

example.txt

Երբ file-ը բացած է, cursor-ը կապված է file-ի հետ

```
>>> infile = open('example.txt')
>>> infile.read(1)
'T'
>>>
```

reading a file

1 The 3[^] lines in this file end with the new line character.\n
2 \n
3 There is a blank line above this line.\n

example.txt

Երբ file-ը բացած է, cursor-ը կապված է file-ի հետ

```
>>> infile = open('example.txt')
>>> infile.read(1)
'T'
>>> infile.read(5)
'he 3 '
>>>
```


reading a file

- 1 The 3 lines in this file end with the new line character.\n^
- 2 \n
- 3 There is a blank line above this line.\n

example.txt

Երբ file-ը բացած է, cursor-ը կապված է file-ի հետ

```
>>> infile = open('example.txt')
>>> infile.read(1)
'T'
>>> infile.read(5)
'he 3 '
>>> infile.readline()
'lines in this file end with the new line
character.\n'
>>>
```

reading a file

1 The 3 lines in this file end with the new line character.\n

2 \n

3 There is a blank line above this line.\n^

example.txt

Երբ file-ը բացած է, cursor-ը կապված է file-ի հետ

```
>>> infile = open('example.txt')
>>> infile.read(1)
'T'
>>> infile.read(5)
'he 3 '
>>> infile.readline()
'lines in this file end with the new line
character.\n'
>>> infile.read()
'\nThere is a blank line above this line.\n'
>>>
```

reading a file

- 1 The 3 lines in this file end with the new line character.\n
- 2 \n
- 3 There is a blank line above this line.\n^

example.txt

Երբ file-ը բացած է, cursor-ը կապված է file-ի հետ

```
>>> infile = open('example.txt')
>>> infile.read(1)
'T'
>>> infile.read(5)
'he 3 '
>>> infile.readline()
'lines in this file end with the new line
character.\n'
>>> infile.read()
'\nThere is a blank line above this line.\n'
>>> infile.close()
```

file կարդալու մոդելներ(patterns)

file կարդալու ընդունված մոդելներն են:

file կարդալու մոդելներ(patterns)

file կարդալու ընդունված մոդելներն են:

1. Read the file content into a string
(կարդա ամբողջը string-ի մեջ)

file կարդալու մոդելներ(patterns)

file կարդալու ընդունված մոդելներն են:

1. Read the file content into a string
(կարդա ամբողջը string-ի մեջ)

```
def numChars(filename):  
    '''returns the number of characters in file  
    filename'''  
  
    infile = open(filename, 'r')  
    content = infile.read()  
    infile.close()  
  
    return len(content)
```

file կարդալու մոդելներ(patterns)

2. Read the file content into a list of words
(կարդա ամբողջը և լցրա բառերի լիստի մեջ)

```
def numWords(filename):  
    '''returns the number of words in file  
    filename'''  
  
    infile = open(filename)  
    content = infile.read()  
    infile.close()  
    wordList = content.split()  
  
    return len(wordList)
```

file կարդալու մոդելներ(patterns)

3. Read the file content into a list of lines
(կարդա և լցրա ամեն տողը լիստի մեջ)

```
def numLines(filename):  
    '''returns the number of lines in file  
    filename'''  
  
    infile = open(filename, 'r')  
    lineList = infile.readlines()  
    infile.close()  
  
    return len(lineList)
```


read file

```
input = open("data.txt")  
for line in input:  
    print(line.strip()) # strip() '\n'
```

read_data.py

read file

```
input = open("data.txt")
for line in input:
    print(line.strip()) # strip() '\n'
```

read_data.py

```
126 Lilo 12.5 8.2 7.1 3.2
456 Steve 4.0 10.4 6.5 2.7 12
789 Janne 6.0 8.0 8.0 8.5 7.5
```

Exercise 4

Գրեք `file_stats` function որը վերցնում է `data file`-ի անունը և տպում է ամենաերկար տողը և տառերի թիվը

Exercise 4

Գրեք `file_stats` function որը վերցնում է `data` file-ի անունը և տպում է ամենաերկար տողը և տառերի թիվը

– input file, `data.txt`:

```
126 Lilo 12.5 8.2 7.1 3.2
456 Steve 4.0 10.4 6.5 2.7 12
789 Janne 6.0 8.0 8.0 8.5 7.5
```

– output:

```
>>> file_stats('data.txt')
```

```
# = 30
```

```
456 Steve 4.0 10.4 6.5 2.7 12
```

Exercise 4

```
def file_stats(filename):
```

```
    ...
```

```
    long = ""
```

```
    ...
```

```
    print("# =", len(long))
```

```
    print(long)
```

read_data.py

Exercise 4

```
def file_stats(filename):
```

```
    ...
```

```
    long = ""
```

```
    ...
```

```
    print("# =", len(long))
```

```
    print(long)
```

read_data.py

```
def file_stats(filename):
```

```
    input = open(filename)
```

```
    long = ""
```

```
    for line in input:
```

```
        if len(line) > len(long):
```

```
            long = line
```

```
    print("# =", len(long))
```

```
    print(long)
```

read_data.py

write text file

write text file

```
1  
2  
3  
4
```

test.txt

```
>>> outfile = open('test.txt', 'w')  
>>>
```


write text file

```
1 T  
2  
3  
4
```

test.txt

```
>>> outfile = open('test.txt', 'w')  
>>> outfile.write('T')  
1  
>>>
```

write text file

```
1 This is the first line. ^
2
3
4
```

test.txt

```
>>> outfile = open('test.txt', 'w')
>>> outfile.write('T')
1
>>> outfile.write('his is the first line.')
22
>>>
```

write text file

```
1 This is the first line. Still the first line...\n
2 ^
3
4
```

test.txt

```
>>> outfile = open('test.txt', 'w')
>>> outfile.write('T')
1
>>> outfile.write('his is the first line.')
22
>>> outfile.write(' Still the first line...\n')
25
>>>
```

write text file

```
1 This is the first line. Still the first line...\n
2 Now we are in the second line.\n
3 ^
4
```

test.txt

```
>>> outfile = open('test.txt', 'w')
>>> outfile.write('T')
1
>>> outfile.write('his is the first line.')
22
>>> outfile.write(' Still the first line...\n')
25
>>> outfile.write('Now we are in the second line.\n')
31
>>>
```

write text file

```
1 This is the first line. Still the first line...\n
2 Now we are in the second line.\n
3 Non string value like 5 must be converted first.\n
4
```

test.txt

```
>>> outfile = open('test.txt', 'w')
>>> outfile.write('T')
1
>>> outfile.write('his is the first line.')
22
>>> outfile.write(' Still the first line...\n')
25
>>> outfile.write('Now we are in the second line.\n')
31
>>> outfile.write('Non string value like '+str(5)+' must be converted first.\n')
49
>>>
```

write text file

```
1 This is the first line. Still the first line...\n
2 Now we are in the second line.\n
3 Non string value like 5 must be converted first.\n
4 Non string value like 5 must be converted first.\n
```

test.txt

```
>>> outfile = open('test.txt', 'w')
>>> outfile.write('T')
1
>>> outfile.write('his is the first line.')
22
>>> outfile.write(' Still the first line...\n')
25
>>> outfile.write('Now we are in the second line.\n')
31
>>> outfile.write('Non string value like '+str(5)+' must be converted first.\n')
49
>>> outfile.write('Non string value like {} must be converted first.\n'.format(5))
49
>>> outfile.close()
```

Exercise 5

1. Ստեղծեք `“write_read.py”` file-ը
2. Բացեք `“tobe.txt”` file-ը գրելու `‘w’` մոդում
3. Գրեք նրանում `“to be or not to be”`
4. Փակեք file-ը
5. Բացեք նույն `“tobe.txt”` file-ը `‘r’` մոդում
6. `print` արեք պարունակությունը

Exercise 5

```
def write_read():  
    filename = "tobe.txt"  
    outfile = open(filename, 'w')  
    outfile.write("to be or not to be")  
    outfile.close()  
  
    infile = open(filename)  
    string = infile.read()  
    print(string)
```

```
write_read() #call function
```

write_read.py

References

1. Franek. “CS 1MD3 Introduction to Programming.” Accessed July 8, 2014.
2. Downey, Allen B. *Think Python*. 1 edition. Sebastopol, CA: O’Reilly Media, 2012.
3. Guo, Philip. “Online Python Tutor - Visualize Program Execution.” *Pythontutor*. Accessed July 16, 2014. <http://pythontutor.com>.