



# Introduction to Programming

## Lesson 6

# Outline

- while loop revisited
- Sequence loop pattern
- Infinite loop pattern
- Loop-and-a-half pattern
- break and continue statements
- Dictionaries
- Sets

# Lesson Code

<https://goo.gl/jGUvPI>

<https://goo.gl/n80Ay6>

# Exercise 1 B

Գրեք function **negative()**:

Function input : list of numbers

returns: առաջին բացասական թվի index-ը  
եթե չկա բացասական : returns -1

```
>>> lst = [3, 1, -7, -4, 9, -2]
>>> negative(lst)
2
>>> negative([1, 2, 3])
-1
```

# Exercise 1

```
>>> lst = [3, 1, -7, -4, 9, -2]
>>> negative(lst)
2
>>> negative([1, 2, 3])
-1
```

# Exercise 1

```
>>> lst = [3, 1, -7, -4, 9, -2]
>>> negative(lst)
2
>>> negative([1, 2, 3])
-1
```

```
def negative (lst):

    #use counter loop pattern
    # if
        # return index

    # return -1
```

# Exercise 1

```
>>> lst = [3, 1, -7, -4, 9, -2]
>>> negative(lst)
2
>>> negative([1, 2, 3])
-1
```

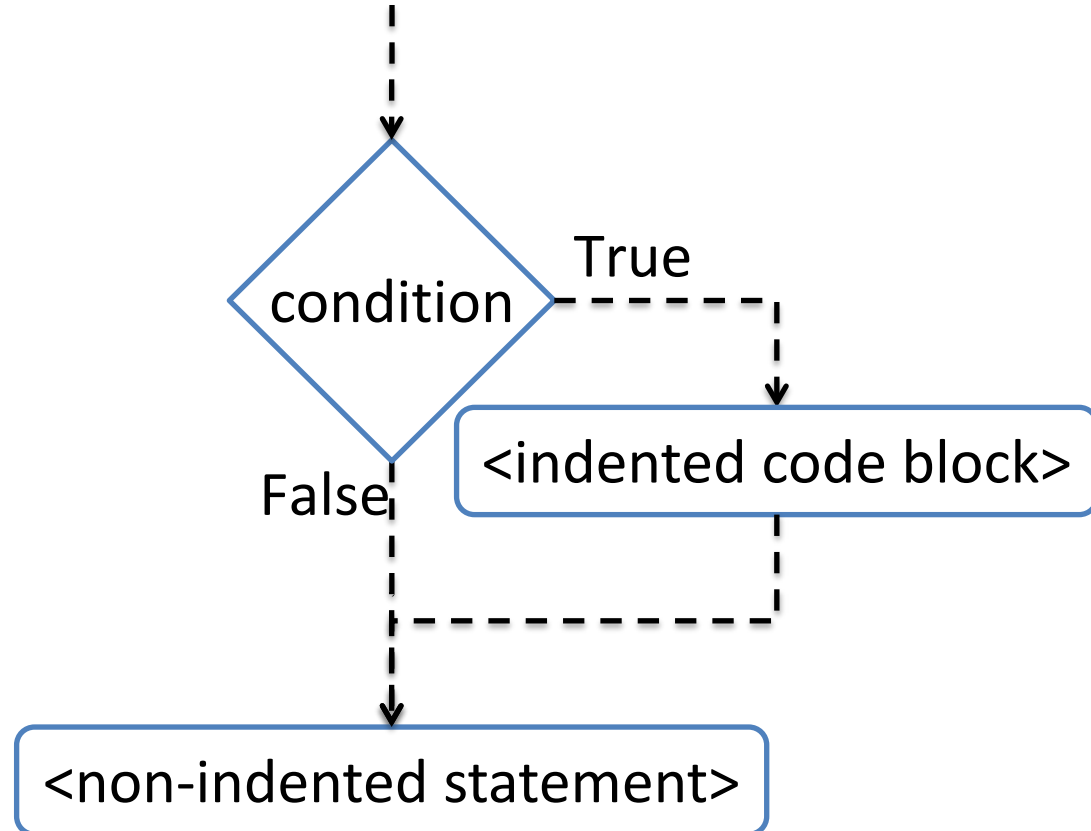
```
def negative (lst):

    for i in range(len(lst)):
        if lst[i] < 0:
            return i

    return -1
```

# while loop

```
if <condition>:  
    <indented code block>  
<non-indented statement>
```

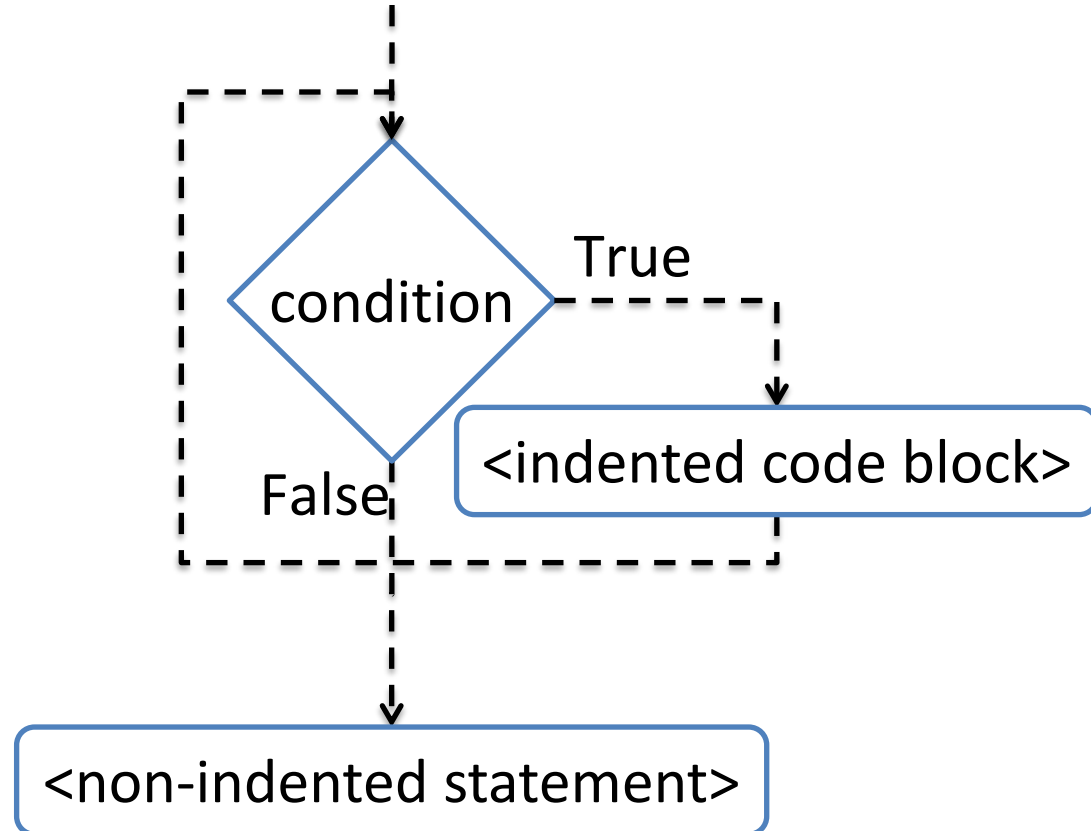




# while loop

```
if <condition>:  
    <indented code block>  
<non-indented statement>
```

```
while <condition>:  
    <indented code block>  
<non-indented statement>
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

↓  
`i = 7`

`i =` 7

```
>>> i = 7  
>>>
```

# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

↓  
`i = 7`

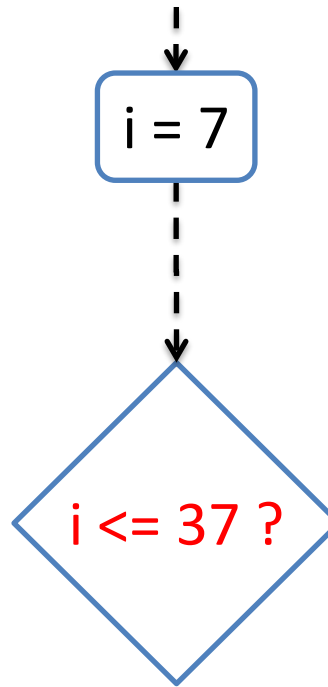
`i =` 7

```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```

# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```

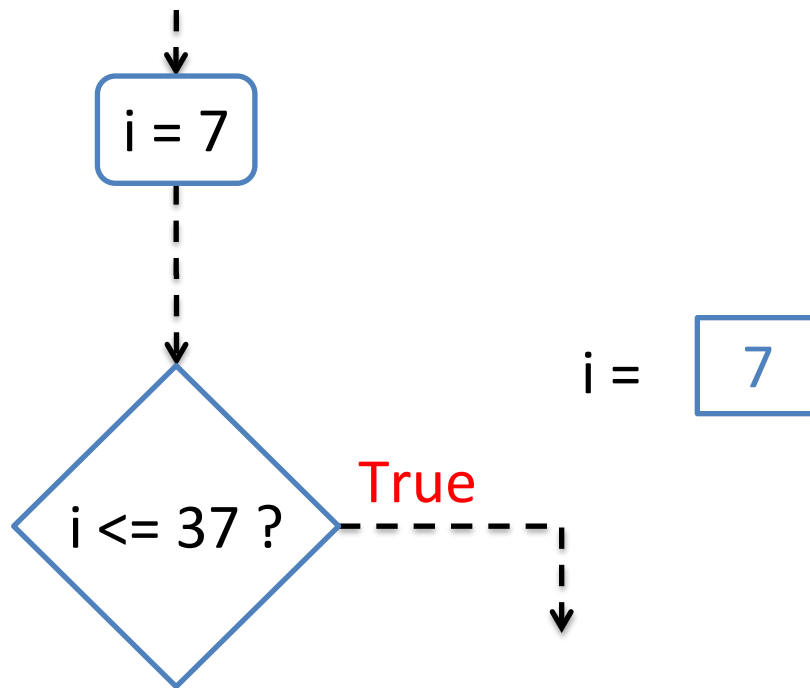


i = 7

# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

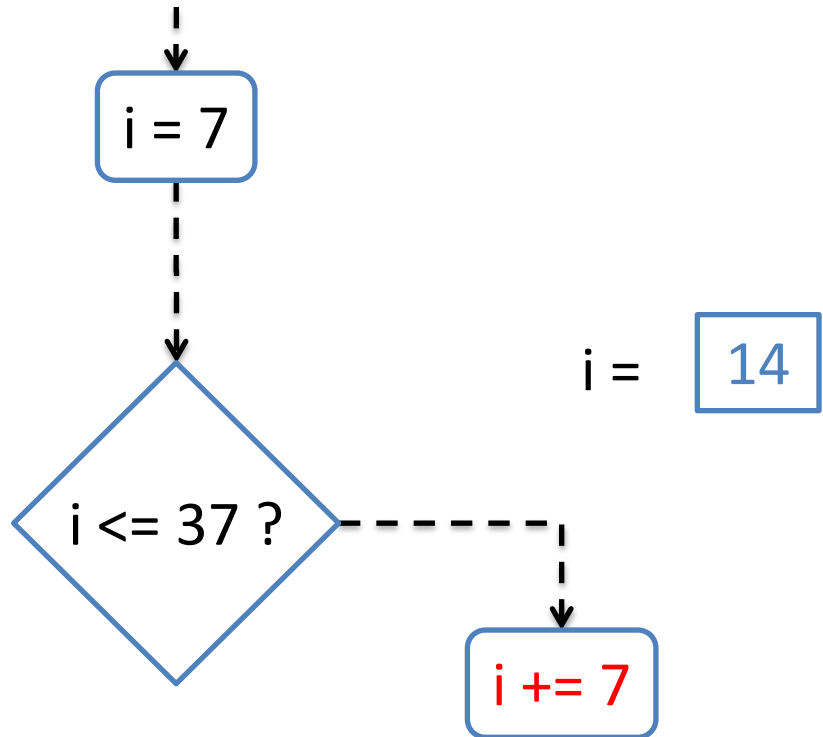
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

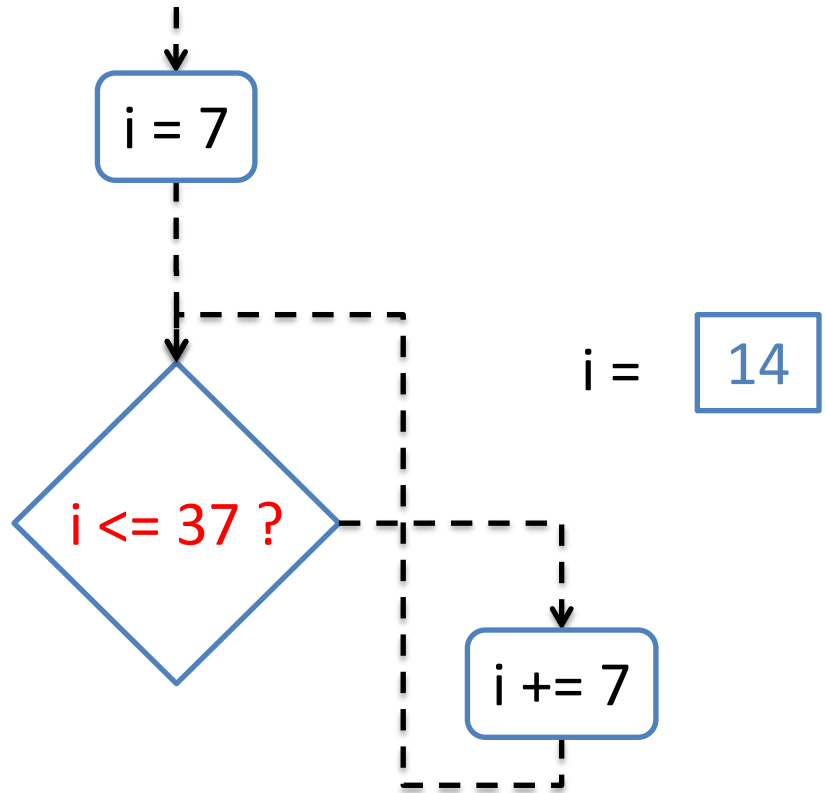
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```

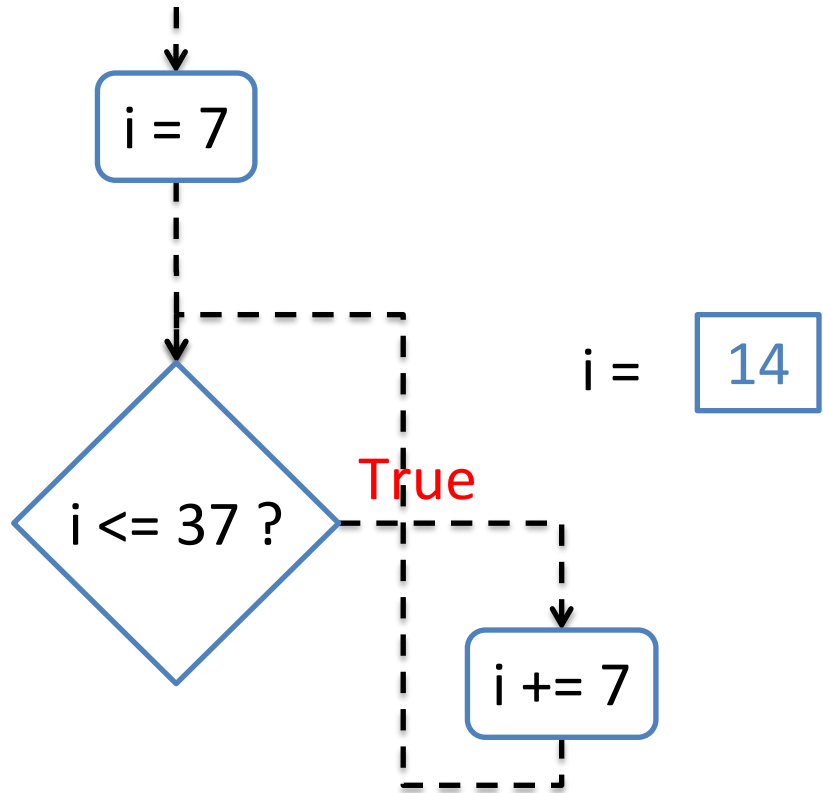




# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

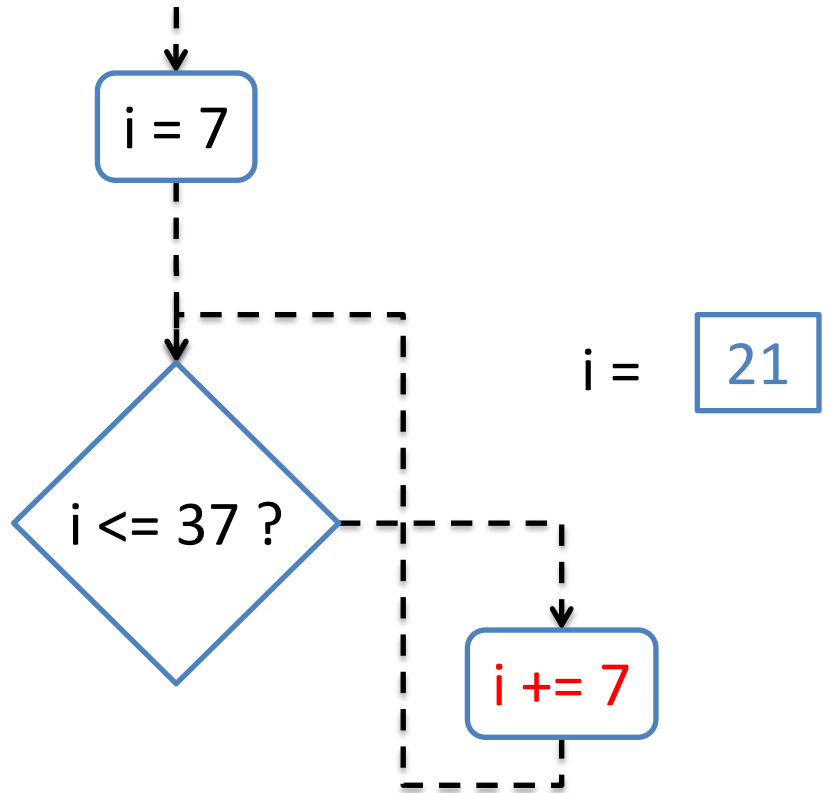
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

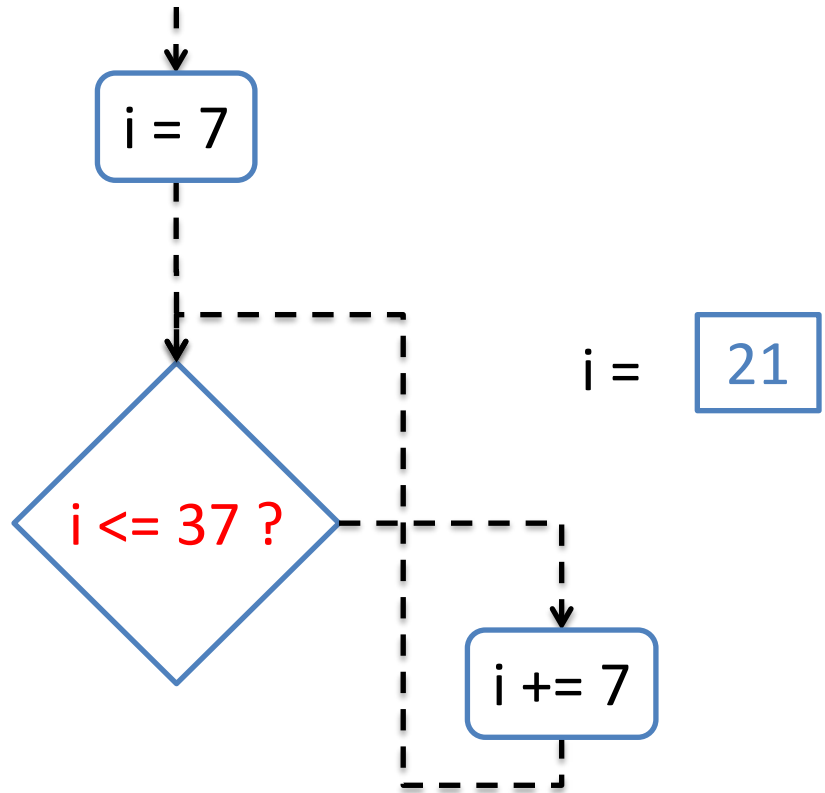
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

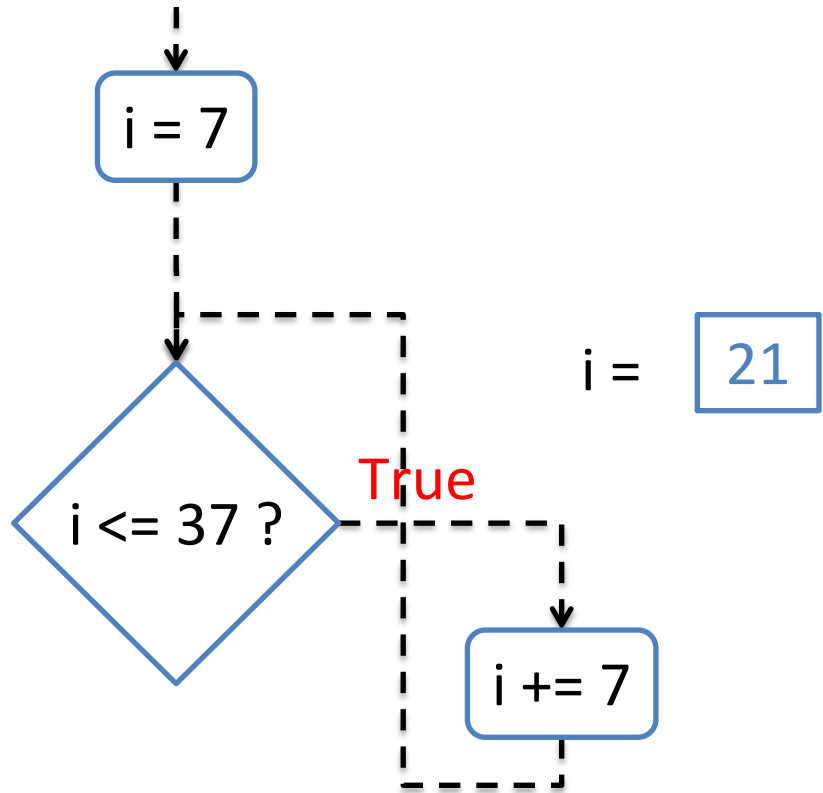
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

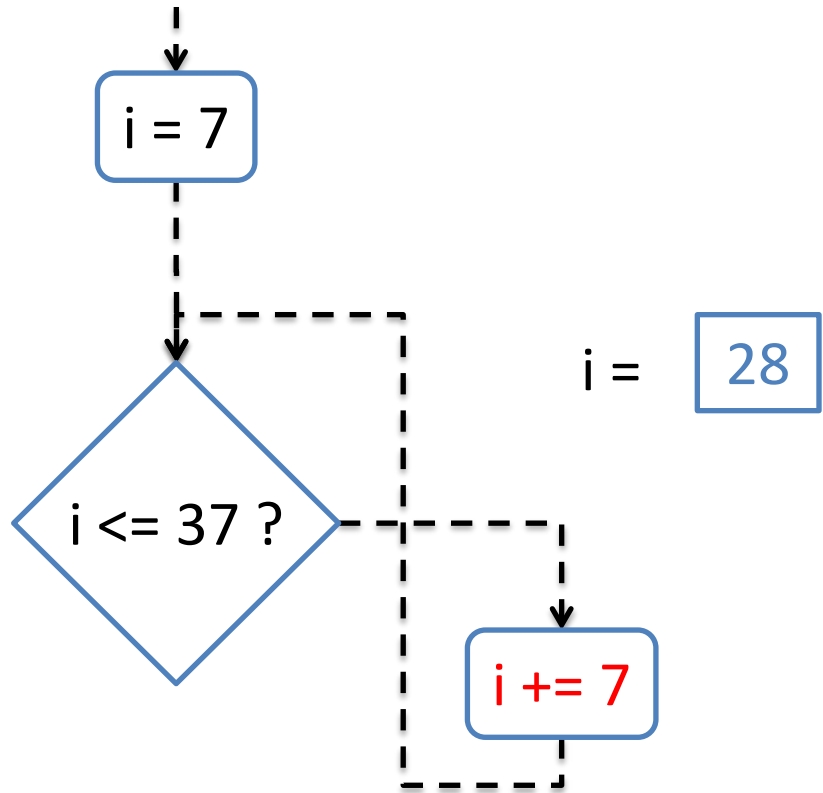
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

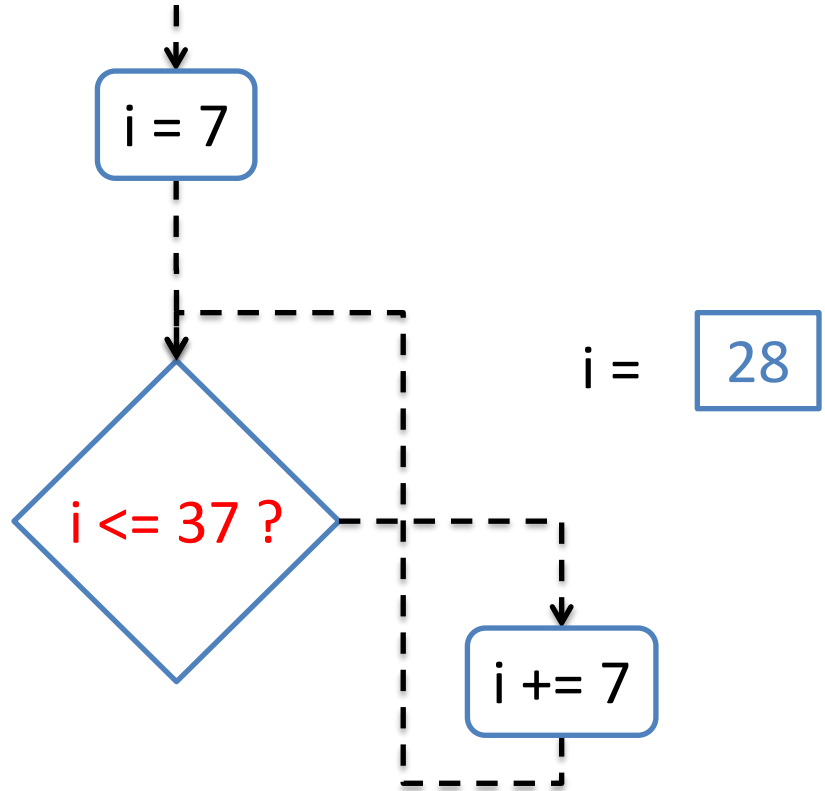
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

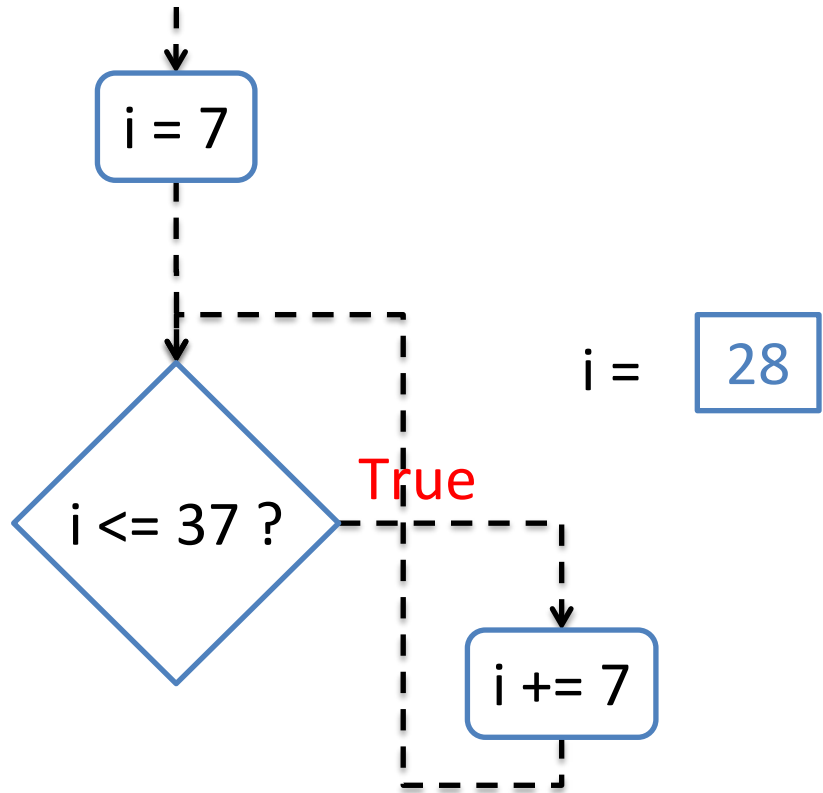
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

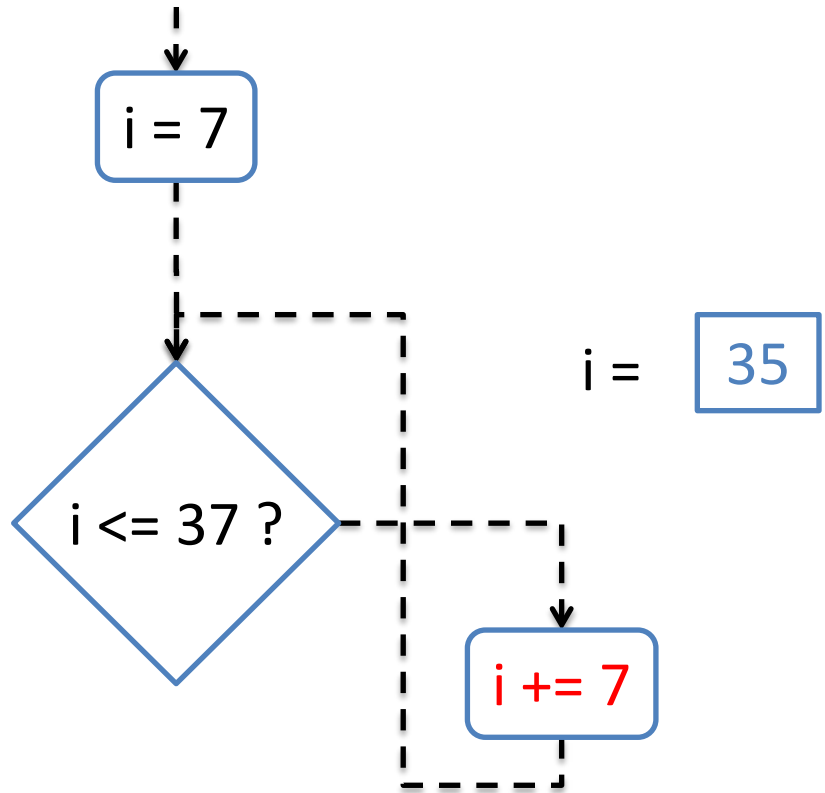
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```

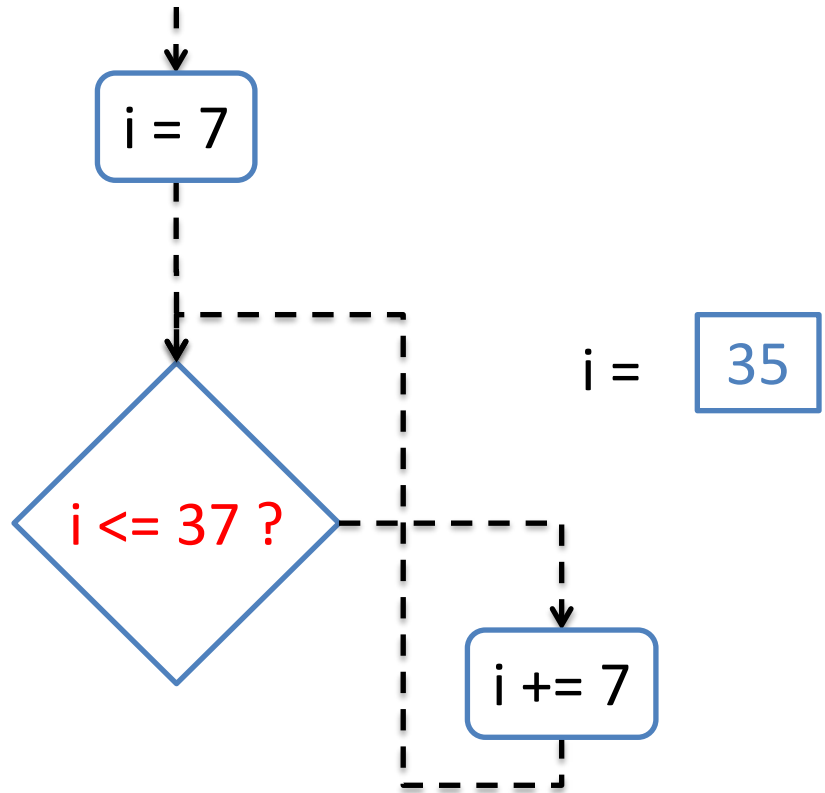




# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

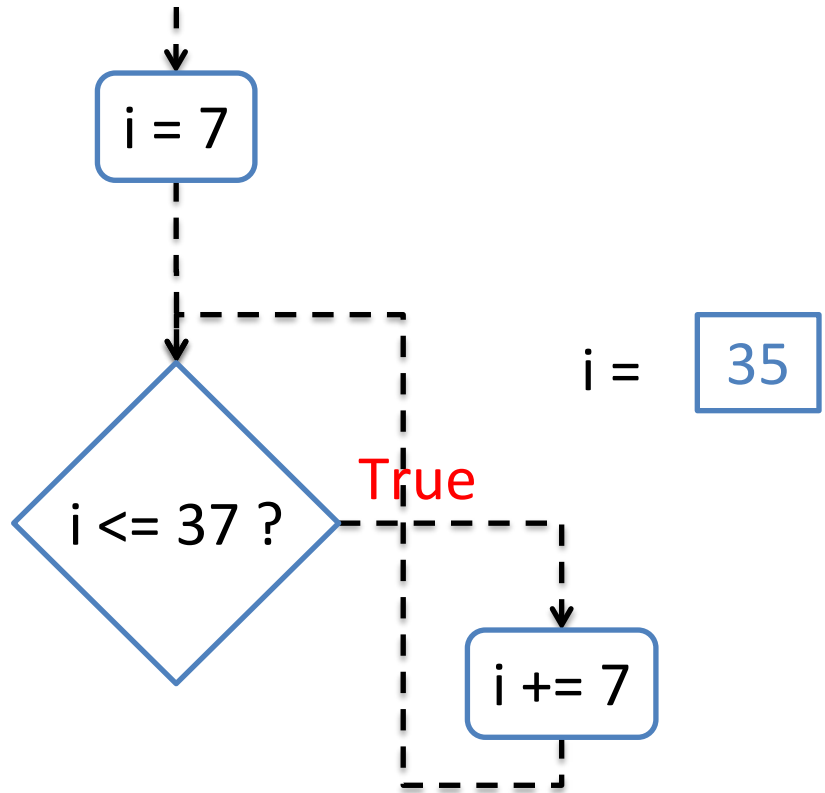
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

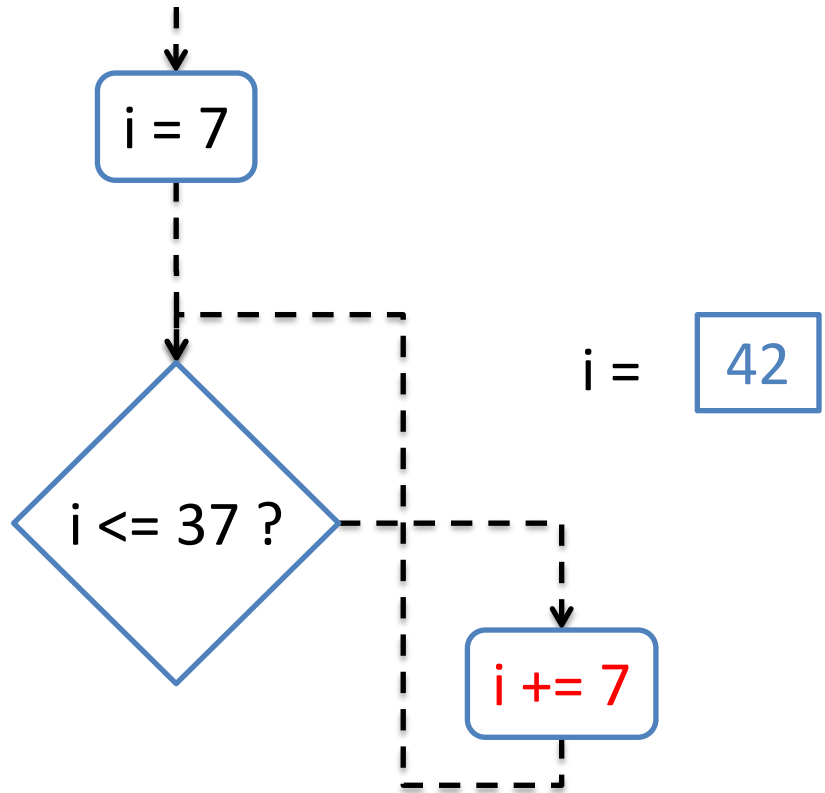
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

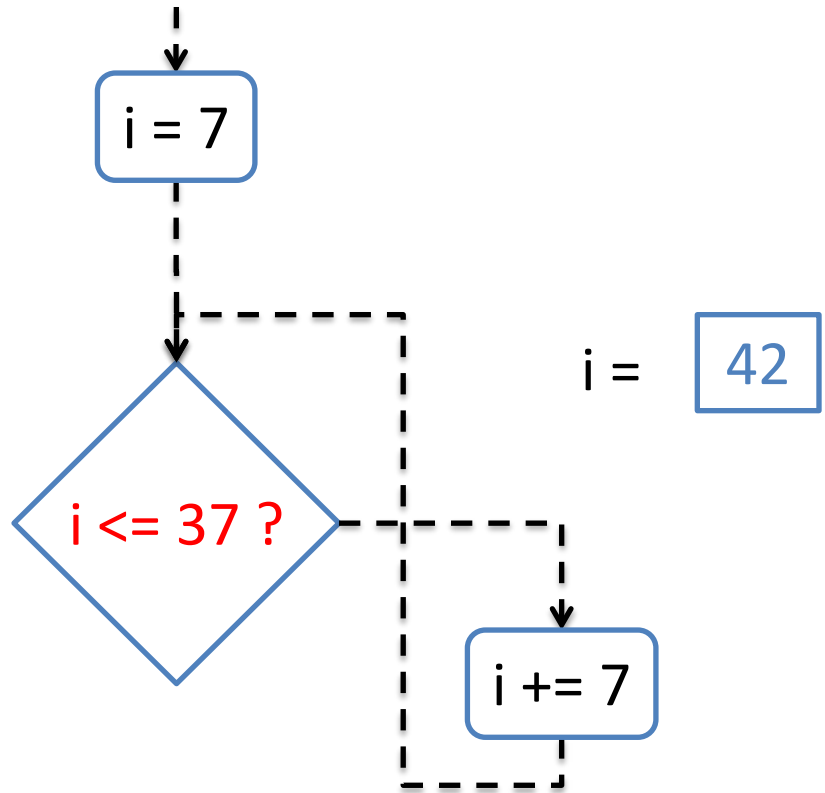
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

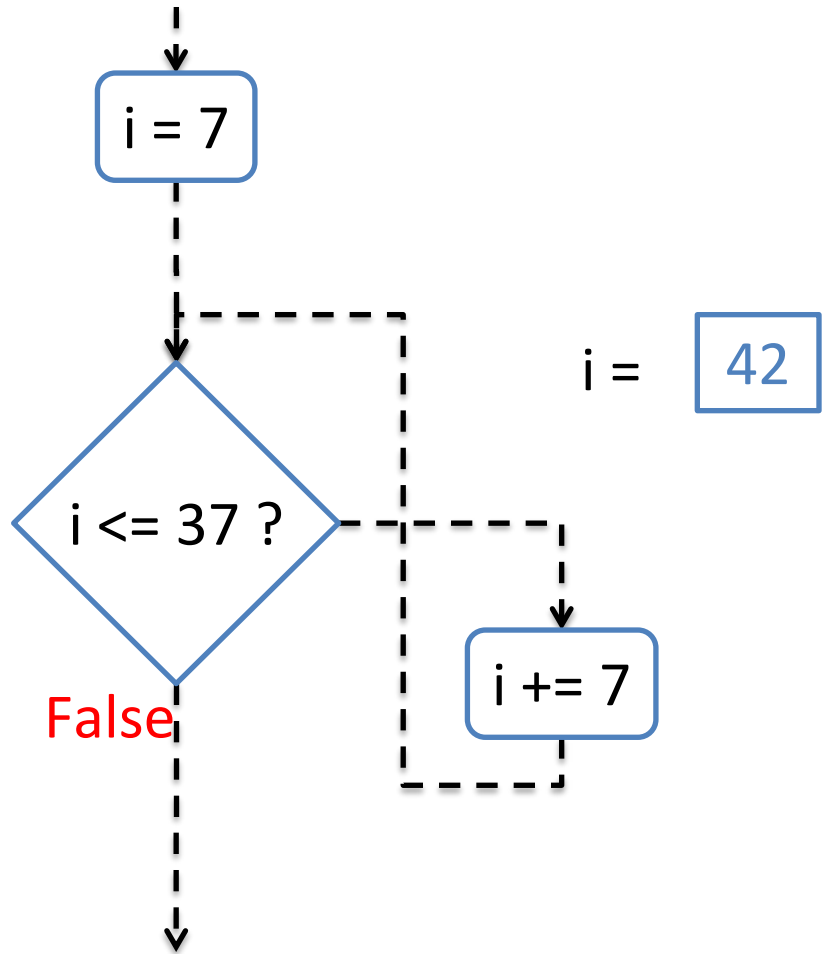
```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```



# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```

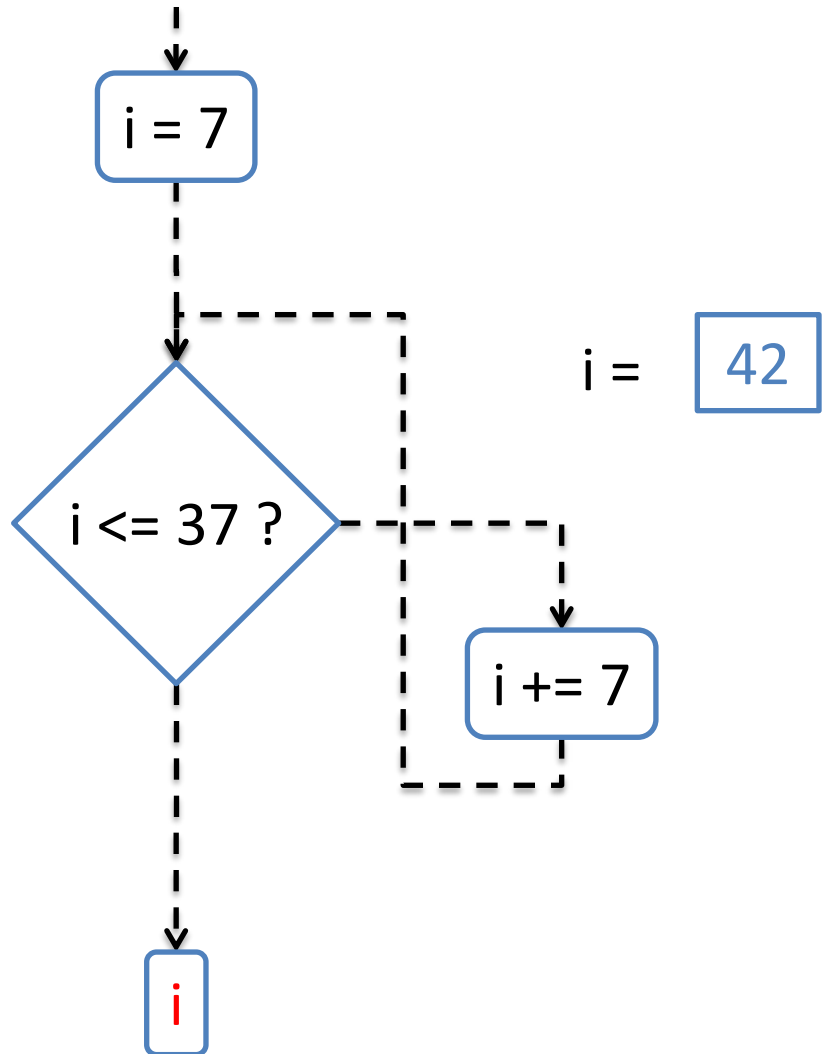


# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

```
>>> i = 7
>>> while i <= 37:
        i += 7
```

```
>>> i
42
```

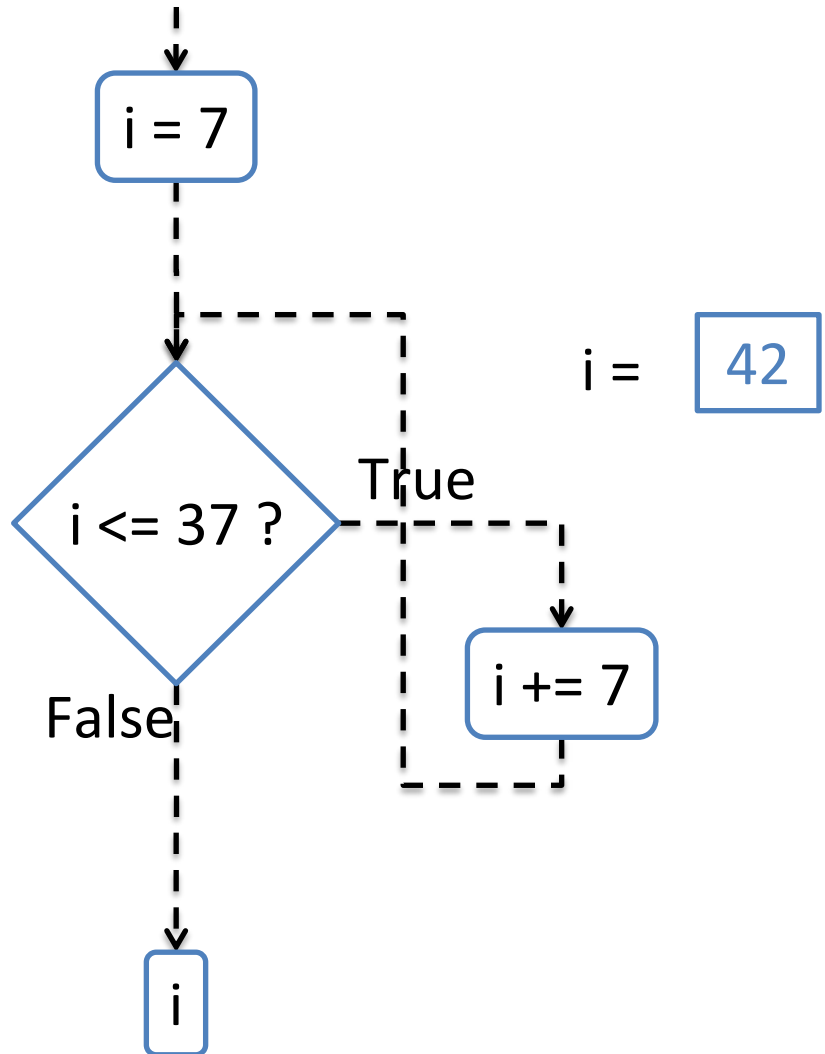


# while loop

Example: հաշվել 7-ի նվազագույն բազմապատիկը որը  $> 37$

```
>>> i = 7
>>> while i <= 37:
>>>     i += 7
```

```
>>> i
42
```



# Sequence loop pattern

Fibonacci sequence



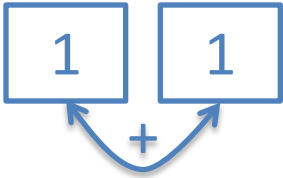
# Sequence loop pattern

Fibonacci sequence



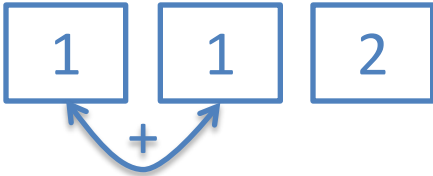
# Sequence loop pattern

Fibonacci sequence



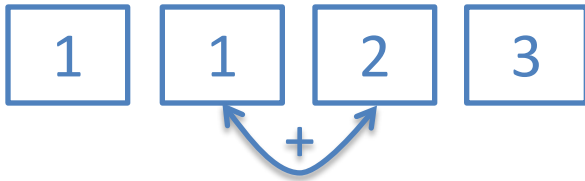
# Sequence loop pattern

Fibonacci sequence



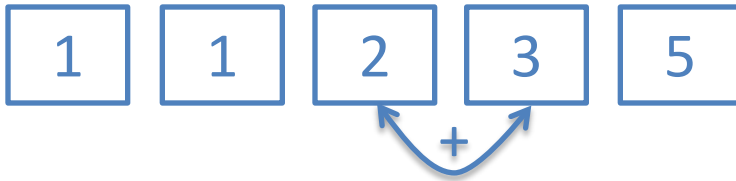
# Sequence loop pattern

Fibonacci sequence



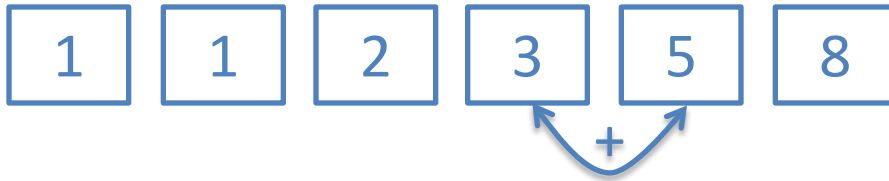
# Sequence loop pattern

Fibonacci sequence



# Sequence loop pattern

Fibonacci sequence



# Sequence loop pattern

Fibonacci sequence



# Sequence loop pattern

Fibonacci sequence





# Sequence loop pattern

Fibonacci sequence



# Sequence loop pattern

Fibonacci sequence



# Sequence loop pattern

Fibonacci sequence



# Sequence loop pattern

Fibonacci sequence



Goal: գենեռացնել Fibonnaci number  
որը մեծ է տրված **bound**-ից

# Sequence loop pattern

Fibonacci sequence



Goal: գենեռացնել Fibonacci number  
որը մեծ է տրված **bound**-ից

```
>>> fibonacci(5)
8
>>> fibonacci(20)
21
```

# Sequence loop pattern

```
def fibonacci(bound):  
  
    p = 1 # previous  
    c = 1 # current  
    while c <= bound:  
        p, c = c, p+c  
  
    return c
```

# Infinite loop pattern

An infinite loop : continuous (շարունակական) service

```
>>> hello()  
What is your name? Gago  
Hello Gago
```

# Infinite loop pattern

An infinite loop : continuous (շարունակական) service

```
>>> hello()  
What is your name? Gago  
Hello Gago  
What is your name?  
Armen  
Hello Armen
```



# Infinite loop pattern

An infinite loop : continuous (շարունակական) service

```
>>> hello()  
What is your name? Gago  
Hello Gago  
What is your name?  
Armen  
Hello Armen  
What is your name?  
Valod  
Hello Valod
```

# Infinite loop pattern

An infinite loop : continuous (շարունակական) service

```
>>> hello()  
What is your name? Gago  
Hello Gago  
What is your name?  
Armen  
Hello Armen  
What is your name?  
Valod  
Hello Valod  
What is your name?
```

# Infinite loop pattern

An infinite loop : continuous (շարունակական) service

```
>>> hello()  
What is your name? Gago  
Hello Gago  
What is your name?  
Armen  
Hello Armen  
What is your name?  
Valod  
Hello Valod  
What is your name?
```

```
def hello():  
    '''a greeting service  
    repeatedly requests user's name  
    and greets'''  
    while True:  
        name = input(  
            'What is your name? '  
        )  
        print('Hello ' + name)
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

```
>>> cities()  
Enter city:
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

```
>>> cities()  
Enter city: Lisbon  
Enter city:
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

```
>>> cities()  
Enter city: Lisbon  
Enter city: San Francisco  
Enter city:
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

```
>>> cities()  
Enter city: Lisbon  
Enter city: San Francisco  
Enter city: Hong Kong  
Enter city:
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

empty string "" =>  
stop and return

```
>>> cities()  
Enter city: Lisbon  
Enter city: San Francisco  
Enter city: Hong Kong  
Enter city:  
['Lisbon', 'San Francisco',  
'Hong Kong']
```



# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

```
def cities():  
    lst = []  
  
    city = input('Enter city: ')  
  
    while city != '':  
        lst.append(city)  
        city = input('Enter city: ')  
  
    return lst
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

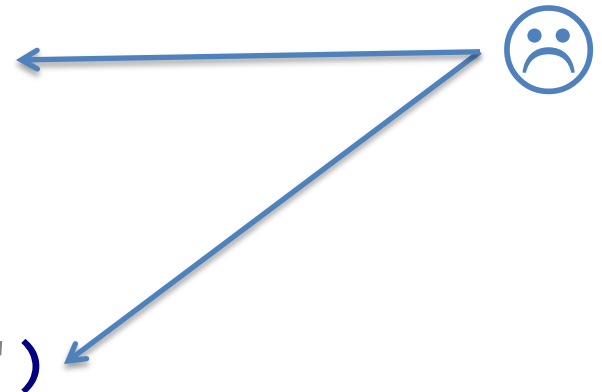
```
def cities():  
    lst = []  
  
    city = input('Enter city: ')  
  
    while city != '':  
        lst.append(city)  
        city = input('Enter city: ')  
  
    return lst
```

accumulator pattern

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

```
def cities():  
    lst = []  
  
    city = input('Enter city: ')  
  
    while city != '':  
        lst.append(city)  
        city = input('Enter city: ')  
  
    return lst
```



# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

```
def cities2():  
    lst = []  
  
    # repeat:  
        # ask user to enter city  
  
        # if ""  
            # return lst  
  
        # lst.append
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

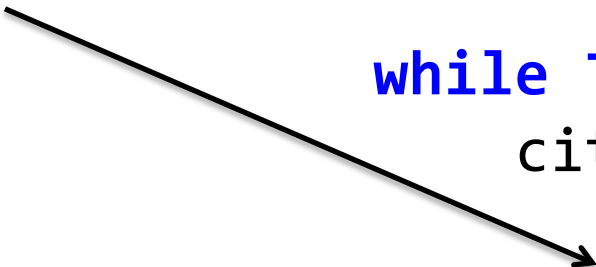
```
def cities2():  
    lst = []  
  
    while True:  
        city = input('Enter city: ')  
  
        if city == '':  
            return lst  
  
        lst.append(city)
```

# Loop-and-a-half pattern

Example: function որը ստեղծում է **user**-ի ներմուծած քաղաքների **list** և վերադարձնում այն

last loop iteration  
stops here

```
def cities2():  
    lst = []  
  
    while True:  
        city = input('Enter city: ')  
  
        if city == '':  
            return lst  
  
        lst.append(city)
```



# break statement

break

- used inside loop (օգտագործվում է ցիկլի մեջ)
- interrupts the loop (ընդհատում է ցիկլը)
- execution continues after the loop body  
(հաշվարկը շարունակվում է ցիկլից դուրս )

# break statement

break

- used inside loop (օգտագործվում է ցիկլի մեջ)
- interrupts the loop (ընդհատում է ցիկլը)
- execution continues after the loop body  
(հաշվարկը շարունակվում է ցիկլից դուրս )

```
def cities2():  
    lst = []  
    while True:  
        city = input(  
            'Enter city: ' )  
  
        if city == '':  
            return lst  
  
    lst.append(city)
```



# break statement

break

- used inside loop (օգտագործվում է ցիկլի մեջ)
- interrupts the loop (ընդհատում է ցիկլը)
- execution continues after the loop body  
(հաշվարկը շարունակվում է ցիկլից դուրս )

```
def cities2():  
    lst = []  
    while True:  
        city = input(  
            'Enter city: ' )  
  
        if city == '':  
            return lst  
  
    lst.append(city)
```

```
def cities3():  
    lst = []  
    while True:  
        city = input(  
            'Enter city: ' )  
  
        if city == '':  
            break  
  
    lst.append(city)  
  
    return lst
```

# continue statement

## continue

- used inside loop (օգտագործվում է ցիկլի մեջ)
- interrupts the loop (ընդհատում է ցիկլը)
- execution continues with next iteration of the loop  
(հաշվարկը շարունակվում է ցիկլի հաջորդ իտերացիայից )

```
>>> table = [  
    [2, 3, 0, 6],  
    [0, 3, 4, 5],  
    [4, 5, 6, 0]]
```

# continue statement

## continue

- used inside loop (օգտագործվում է ցիկլի մեջ)
- interrupts the loop (ընդհատում է ցիկլը)
- execution continues with next iteration of the loop (հաշվարկը շարունակվում է ցիկլի հաջորդ իտերացիայից )

```
>>> table = [  
    [2, 3, 0, 6],  
    [0, 3, 4, 5],  
    [4, 5, 6, 0]]  
>>> ignore(table)  
2 3 6  
3 4 5  
4 5 6
```

# continue statement

## continue

- used inside loop (օգտագործվում է ցիկլի մեջ)
- interrupts the loop (ընդհատում է ցիկլը)
- execution continues with next iteration of the loop (հաջվարկը շարունակվում է ցիկլի հաջորդ իտերացիայից )

```
>>> table = [  
    [2, 3, 0, 6],  
    [0, 3, 4, 5],  
    [4, 5, 6, 0]]  
>>> ignore(table)  
2 3 6  
3 4 5  
4 5 6
```

```
def ignore(table):  
    for row in table:  
        for num in row:  
            if num == 0:  
                continue  
            print(num, end=' ' )  
        print()
```

# break and continue statements

In both cases, only the innermost loop is affected

Երկու դեպքում էլ ներքին ցիկլն է ընդհատվում

# break and continue statements

In both cases, only the innermost loop is affected

Երկու դեպքում էլ ներքին ցիկլն է ընդհատվում

```
>>> table = [  
    [2, 3, 0, 6],  
    [0, 3, 4, 5],  
    [4, 5, 6, 0]]
```

# break and continue statements

In both cases, only the innermost loop is affected

Երկու դեպքում էլ ներքին ցիկլն է ընդհատվում

```
>>> table = [  
    [2, 3, 0, 6],  
    [0, 3, 4, 5],  
    [4, 5, 6, 0]]
```

```
>>> ignore(table)  
2 3 6  
3 4 5  
4 5 6
```

```
def ignore(table):  
    for row in table:  
        for num in row:  
            if num == 0:  
                continue  
            print(num, end=' ')  
        print()
```

# break and continue statements

In both cases, only the innermost loop is affected

Երկու դեպքում էլ ներքին ցիկլն է ընդհատվում

```
>>> before(table)
```

```
2 3
```

```
4 5 6
```

```
>>> table = [  
    [2, 3, 0, 6],  
    [0, 3, 4, 5],  
    [4, 5, 6, 0]]
```

```
>>> ignore(table)
```

```
2 3 6
```

```
3 4 5
```

```
4 5 6
```

```
def before(table):  
    for row in table:  
        for num in row:  
            if num == 0:  
                break  
            print(num, end= ' ' )  
        print()
```

```
def ignore(table):  
    for row in table:  
        for num in row:  
            if num == 0:  
                continue  
            print(num, end= ' ' )  
        print()
```



# Dictionaries

Goal: աշխատողների գրառումների կոնտեյներ  
ինդեքսավորված ըստ աշխատակցի passport-ի համարի P#

```
>>> employee['864-20']  
['Arm', 'Spartyan']  
>>> employee['100-01']  
['Kiazh', 'Damyan']  
>>> employee['987-65']  
['Vlad', 'Tundyan']  
>>>
```

# Dictionaries

Goal: աշխատողների գրառումների կոնստեյնտներ  
ինդեքսավորված ըստ աշխատակցի passport-ի համարի P#

## Problems:

- P#-ի range-ը շատ մեծ է
- P#-ները integer-ներ չեն

**Solution:** dictionary class dict

```
>>> employee['864-20']  
['Arm', 'Spartyan']  
>>> employee['100-01']  
['Kiazh', 'Damyan']  
>>> employee['987-65']  
['Vlad', 'Tundyan']  
>>>
```

key	value
'864-20'	['Arm', 'Spartyan']
'987-65'	['Vlad', 'Tundyan']
'100-01'	['Kiazh', 'Damyan']

# Dictionaries

key	value
'864-20'	['Arm', 'Spartyan']
'987-65'	['Vlad', 'Tundyan']
'100-01'	['Kiazh', 'Damyan']

dictionary-ն  
պարունակում է  
(key, value) pairs/զույգեր

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyan']  
}
```

# Dictionaries

key	value
'864-20'	['Arm', 'Spartyan']
'987-65'	['Vlad', 'Tundyan']
'100-01'	['Kiazh', 'Damyan']

dictionary-ն  
պարունակում է  
(key, value) pairs/զույգեր

key-ն օգտագործվում է  
որպես index

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyan']  
}
```

# Dictionaries

key	value
'864-20'	['Arm', 'Spartyan']
'987-65'	['Vlad', 'Tundyan']
'100-01'	['Kiazh', 'Damyan']

dictionary-ն  
պարունակում է  
(key, value) pairs/զույգեր

key-ն օգտագործվում է  
նրպես index

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyan']}
```

```
>>> employee['864-20']  
['Arm', 'Spartyan']
```

# Dictionaries

key	value
'864-20'	['Arm', 'Spartyan']
'987-65'	['Vlad', 'Tundyan']
'100-01'	['Kiazh', 'Damyan']

dictionary-ն  
պարունակում է  
(key, value) pairs/զույգեր

key-ն օգտագործվում է  
որպես index

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyan']  
}  
  
>>> employee['864-20']  
['Arm', 'Spartyan']  
>>> employee['100-01']  
['Kiazh', 'Damyan']
```

# Properties of dictionaries

Dictionaries:

not ordered

(դասավորված չեն)

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyant']}
```

# Properties of dictionaries

Dictionaries:

not ordered

(դասավորված չեն)

Dictionaries : mutable

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyant']}  
>>> employee['864-20'] = ['Inga',  
    'Northstrand']
```



# Properties of dictionaries

Dictionaries:

not ordered

(դասավորված չեն)

Dictionaries : mutable

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyan']}  
>>> employee['864-20'] = ['Inga',  
    'Northstrand']  
>>> employee  
{'864-20': ['Inga', 'Northstrand'],  
 '100-01': ['Kiazh', 'Damyan'], '987-  
65': ['Vlad', 'Tundyan']}
```

# Properties of dictionaries

Dictionaries:

not ordered

(դասավորված չեն)

Dictionaries : mutable

- new (key,value) pairs  
կարելի է ավելացնել

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyant']}  
>>> employee['864-20'] = ['Inga',  
    'Northstrand']  
>>> employee  
{'864-20': ['Inga', 'Northstrand'],  
 '100-01': ['Kiazh', 'Damyant'], '987-  
65': ['Vlad', 'Tundyan']}
```

# Properties of dictionaries

Dictionaries:

not ordered

(դասավորված չեն)

Dictionaries : mutable

- new (key,value) pairs  
կարելի է ավելացնել
- համապատասխան  
key-ի value-ն կարելի է  
փոխարինել

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyant']}  
>>> employee['864-20'] = ['Inga',  
    'Northstrand']  
>>> employee  
{'864-20': ['Inga', 'Northstrand'],  
 '100-01': ['Kiazh', 'Damyant'], '987-  
65': ['Vlad', 'Tundyan']}
```

# Properties of dictionaries

Dictionaries:

not ordered

(դասավորված չեն)

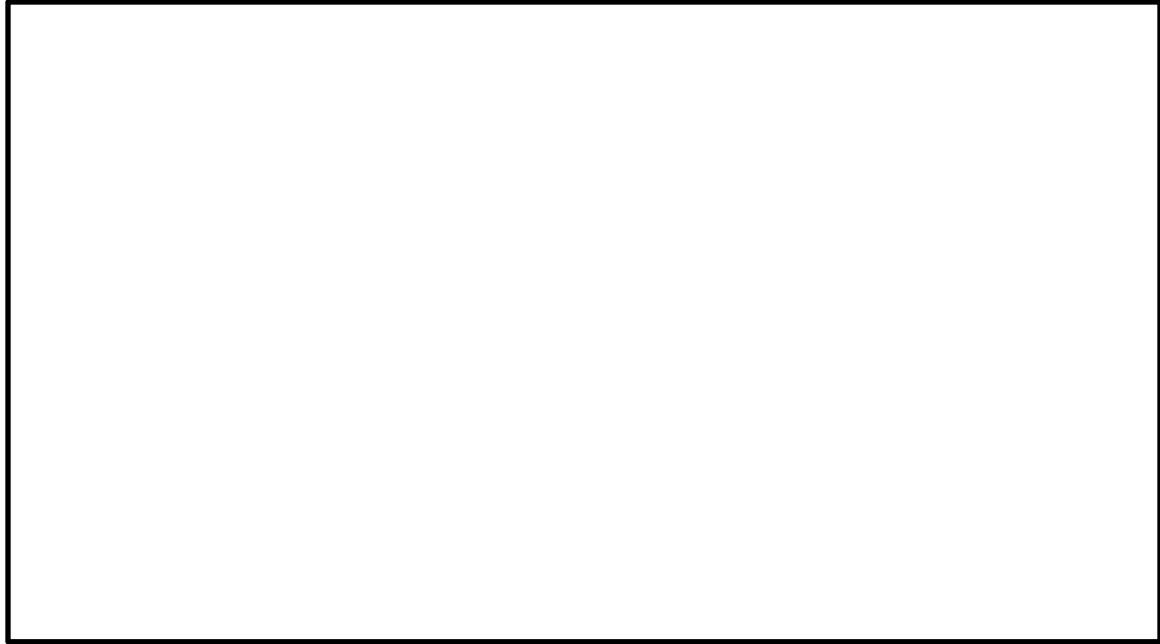
Dictionaries : mutable

- new (key,value) pairs կարելի է ավելացնել
- համապատասխան key-ի value-ն կարելի է փոխարինել

```
>>> employee = {  
    '864-20': ['Arm', 'Spartyan'],  
    '987-65': ['Vlad', 'Tundyan'],  
    '100-01': ['Kiazh', 'Damyant']}  
>>> employee['864-20'] = ['Inga',  
    'Northstrand']  
>>> employee  
{'864-20': ['Inga', 'Northstrand'],  
 '100-01': ['Kiazh', 'Damyant'], '987-  
65': ['Vlad', 'Tundyan']}  
>>> employee['864-20'] = ['Inga',  
    'Northstrand']  
>>> employee  
{'864-20': ['Inga', 'Northstrand'],  
 '100-01': ['Kiazh', 'Damyant'], '987-  
65': ['Vlad', 'Tundyan']}
```

# Properties of dictionaries

`empty(դատարկ)`  
`dictionary {} կամ dict()`



# Properties of dictionaries

empty(դատարկ)  
dictionary {} կամ dict()

```
>>> emptydict = {}  
>>> emptydict  
{}
```

# Properties of dictionaries

empty(դատարկ)  
dictionary {} կամ dict()

```
>>> emptydict = {}  
>>> emptydict  
{}  
>>> type(emptydict)  
<class 'dict'>
```

# Properties of dictionaries

empty(դատարկ)  
dictionary {} կամ dict()

```
>>> emptydict = {}  
>>> emptydict  
{}  
>>> type(emptydict)  
<class 'dict'>  
>>> emptydict = dict()  
>>> emptydict  
{}
```



# Properties of dictionaries

empty(դատարկ)  
dictionary {} կամ dict()

```
>>> emptydict = {}  
>>> emptydict  
{}  
>>> type(emptydict)  
<class 'dict'>  
>>> emptydict = dict()  
>>> emptydict  
{}
```

Dictionary-ի key-երը  
immutable են

# Properties of dictionaries

empty(դատարկ)  
dictionary {} կամ dict()

```
>>> emptydict = {}  
>>> emptydict  
{}  
>>> type(emptydict)  
<class 'dict'>  
>>> emptydict = dict()  
>>> emptydict  
{}
```

Dictionary-ի key-երը  
immutable են

```
>>> employee =  
{[1, 2]: 1, [2, 3]: 3}
```

# Properties of dictionaries

empty(դատարկ)  
dictionary {} կամ dict()

```
>>> emptydict = {}  
>>> emptydict  
{}  
>>> type(emptydict)  
<class 'dict'>  
>>> emptydict = dict()  
>>> emptydict  
{}
```

Dictionary-ի key-երը  
immutable են

```
>>> employee =  
{[1,2]:1,[2,3]:3}  
Traceback
```

# Properties of dictionaries

empty(դատարկ)  
dictionary {} կամ dict()

```
>>> emptydict = {}  
>>> emptydict  
{}  
>>> type(emptydict)  
<class 'dict'>  
>>> emptydict = dict()  
>>> emptydict  
{}
```

Dictionary-ի key-երը  
immutable են

```
>>> employee =  
{[1,2]:1,[2,3]:3}  
Traceback  
TypeError: unhashable type:  
'list'
```

# Dictionary Operators

class **dict** ունի մի քանի օպերատոր, որոնք նույնն են  
class list-ի համար

# Dictionary Operators

class **dict** ունի մի քանի օպերատոր, որոնք նույնն են  
class list-ի համար

```
>>> days = {'Mo':1, 'Tu':2, 'W':3}
>>> days['Mo']
1
>>> days['Th'] = 5
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 5, 'W': 3}
>>> days['Th'] = 4
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4, 'W': 3}
>>> 'Fr' in days
False
>>> len(days)
4
```

# Dictionary methods

Operation	Explanation
d.items()	Returns a view of the (key, value) pairs in d Վերադարձնում է (key, value) զույգերի view
d.keys()	Returns a view of the keys of d Վերադարձնում է key-երի view
d.pop( <b>key</b> )	Removes the (key, value) pair with key <b>key</b> from d and returns the value Ջնջում և վերադարձնում է key d dict-ից
d.update(d2)	Adds the (key, value) pairs of dictionary d2 to d Ավելացնում է d-ին d2-ի (key, value) զույգերը
d.values()	Returns a view of the values of d Վերադարձնում է դ-ի value-ների տեսքը

# Dictionary methods

```
>>> days  
{ 'Mo' : 1, 'Tu' : 2, 'Th' : 4,  
  'W' : 3 }
```



# Dictionary methods

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4,
 'W': 3}
>>> days.pop('Tu')
2
```

# Dictionary methods

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4,
 'W': 3}
>>> days.pop('Tu')
2
>>> days
{'Mo': 1, 'Th': 4, 'W': 3}
```

# Dictionary methods

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4,
 'W': 3}
>>> days.pop('Tu')
2
>>> days
{'Mo': 1, 'Th': 4, 'W': 3}
>>> days2 = {'Tu': 2, 'Fr': 5}
>>> days.update(days2)
>>> days
{'Fr': 5, 'W': 3, 'Th': 4,
 'Mo': 1, 'Tu': 2}
```

# Dictionary methods

```
>>> days.items()  
dict_items([('Fr', 5), ('W',  
3), ('Th', 4), ('Mo', 1),  
('Tu', 2)])
```

```
>>> days  
{'Mo': 1, 'Tu': 2, 'Th': 4,  
'W': 3}  
>>> days.pop('Tu')  
2  
>>> days  
{'Mo': 1, 'Th': 4, 'W': 3}  
>>> days2 = {'Tu': 2, 'Fr': 5}  
>>> days.update(days2)  
>>> days  
{'Fr': 5, 'W': 3, 'Th': 4,  
'Mo': 1, 'Tu': 2}
```

# Dictionary methods

```
>>> days.items()
dict_items([('Fr', 5), ('W', 3), ('Th', 4), ('Mo', 1), ('Tu', 2)])
>>> days.keys()
dict_keys(['Fr', 'W', 'Th', 'Mo', 'Tu'])
```

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4, 'W': 3}
>>> days.pop('Tu')
2
>>> days
{'Mo': 1, 'Th': 4, 'W': 3}
>>> days2 = {'Tu': 2, 'Fr': 5}
>>> days.update(days2)
>>> days
{'Fr': 5, 'W': 3, 'Th': 4, 'Mo': 1, 'Tu': 2}
```

# Dictionary methods

```
>>> days.items()
dict_items([('Fr', 5), ('W', 3), ('Th', 4), ('Mo', 1), ('Tu', 2)])
>>> days.keys()
dict_keys(['Fr', 'W', 'Th', 'Mo', 'Tu'])
>>> vals = days.values()
>>> vals
dict_values([5, 3, 4, 1, 2])
```

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4, 'W': 3}
>>> days.pop('Tu')
2
>>> days
{'Mo': 1, 'Th': 4, 'W': 3}
>>> days2 = {'Tu': 2, 'Fr': 5}
>>> days.update(days2)
>>> days
{'Fr': 5, 'W': 3, 'Th': 4, 'Mo': 1, 'Tu': 2}
```

# Dictionary methods

```
>>> days.items()
dict_items([('Fr', 5), ('W', 3), ('Th', 4), ('Mo', 1), ('Tu', 2)])
>>> days.keys()
dict_keys(['Fr', 'W', 'Th', 'Mo', 'Tu'])
>>> vals = days.values()
>>> vals
dict_values([5, 3, 4, 1, 2])
```

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4, 'W': 3}
>>> days.pop('Tu')
2
>>> days
{'Mo': 1, 'Th': 4, 'W': 3}
>>> days2 = {'Tu': 2, 'Fr': 5}
>>> days.update(days2)
>>> days
{'Fr': 5, 'W': 3, 'Th': 4, 'Mo': 1, 'Tu': 2}
```

`d.items()`, `d.keys()` և `d.values()`

մեթոդները (կոչվում են **views**) վերադարձնում են իստեռացվող container-ներ

# Dictionary methods

```
>>> days.items()
dict_items([('Fr', 5), ('W', 3), ('Th', 4), ('Mo', 1), ('Tu', 2)])
>>> days.keys()
dict_keys(['Fr', 'W', 'Th', 'Mo', 'Tu'])
>>> vals = days.values()
>>> vals
dict_values([5, 3, 4, 1, 2])
>>> for val in vals:
    print(val, end=' ')
```

5 3 4 1 2

```
>>>
```

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4, 'W': 3}
>>> days.pop('Tu')
2
>>> days
{'Mo': 1, 'Th': 4, 'W': 3}
>>> days2 = {'Tu': 2, 'Fr': 5}
>>> days.update(days2)
>>> days
{'Fr': 5, 'W': 3, 'Th': 4, 'Mo': 1, 'Tu': 2}
```

`d.items()`, `d.keys()` և `d.values()`

մեթոդները (կոչվում են **views**) վերադարձնում են իստեացվող container-ներ



# Dictionary methods

```
>>> days.items()
dict_items([('Fr', 5), ('W', 3), ('Th', 4), ('Mo', 1), ('Tu', 2)])
>>> days.keys()
dict_keys(['Fr', 'W', 'Th', 'Mo', 'Tu'])
>>> vals = days.values()
>>> vals
dict_values([5, 3, 4, 1, 2])
>>> for val in vals:
    print(val, end=' ')
```

5 3 4 1 2

```
>>>
```

```
>>> days
{'Mo': 1, 'Tu': 2, 'Th': 4, 'W': 3}
>>> days.pop('Tu')
2
>>> days
{'Mo': 1, 'Th': 4, 'W': 3}
>>> days2 = {'Tu': 2, 'Fr': 5}
>>> days.update(days2)
>>> days
{'Fr': 5, 'W': 3, 'Th': 4, 'Mo': 1, 'Tu': 2}
```

`d.items()`, `d.keys()` և `d.values()`

մեթոդները (կոչվում են **views**) վերադարձնում են իստեացվող container-ներ

# `{}/dict()` **vs** `if/elif/else`

Uses of a dictionary: alternative for multi-way if statement

# `{} / dict()` vs `if/elif/else`

Uses of a dictionary: alternative for multi-way if statement

```
def complete(abbreviation):  
    '''returns day of the week  
    corresponding to  
    abbreviation  
    '''  
  
    if abbreviation == 'Mo':  
        return 'Monday'  
    elif abbreviation == 'Tu':  
        return 'Tuesday'  
    elif  
        .....  
    else: #abbreviation Su  
        return 'Sunday'
```

# `{}/dict()` vs `if/elif/else`

Uses of a dictionary: alternative for multi-way if statement

```
def complete(abbreviation):  
    '''returns day of the week  
    corresponding to  
    abbreviation  
    '''  
  
    if abbreviation == 'Mo':  
        return 'Monday'  
    elif abbreviation == 'Tu':  
        return 'Tuesday'  
    elif  
        .....  
    else: #abbreviation Su  
        return 'Sunday'
```

```
def complete(abbreviation):  
    '''returns day of the week  
    corresponding to abbreviation  
    '''  
  
    days = {'Mo': 'Monday',  
            'Tu': 'Tuesday',  
            'We': 'Wednesday',  
            'Th': 'Thursday',  
            'Fr': 'Friday',  
            'Sa': 'Saturday',  
            'Su': 'Sunday'}  
    return days[abbreviation]
```

# dict() as a container of counters

Uses of a dictionary: container of counters

# dict() as a container of counters

Uses of a dictionary: container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
>>> frequency(grades)
{96: 1, 90: 1, 100: 3, 85: 1, 95: 3}
>>>
```

# dict() as a container of counters

Uses of a dictionary: container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
>>> frequency(grades)
{96: 1, 90: 1, 100: 3, 85: 1, 95: 3}
>>>
```

Solution:

- list-ը iterate արեք
- ամեն grade-ի համար, մեծացրեք counter-ը
- օգտագործեք dictionary, որի
  - key is grade
  - value is counter

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```



# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
               ^
```

counters

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

95

1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

95

1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

95

1

96

1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
               ^
```

counters

95	96
1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

95	96	100
1	1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

^

counters

95

1

96

1

100

1



# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

95	96	100	85
1	1	1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

95	96	100	85
1	1	1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

	95	96	100	85
counters	2	1	1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

	95	96	100	85
counters	2	1	1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

counters

95	96	100	85	90
2	1	1	1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

	95	96	100	85	90
counters	2	1	1	1	1

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

	95	96	100	85	90
counters	3	1	1	1	1

```
counters = {}
```

# dict() as a container of counters

Problem: քանի անգամ է item-ը կրկնվում list-ում

```
>>> grades = [95, 96, 100, 85, 95, 90, 95, 100, 100]
```

	95	96	100	85	90
counters	3	1	1	1	1

```
counters = {}  
for item in itemList:  
    if item in counters:  
        counters[item] += 1  
    else:  
        counters[item] = 1  
return counters
```



# Exercise 2A

Գրեք ծրագիր որը

- վերցնում է տվյալ string-ը 'to be or not to be'
- և տպում է ամեն բառի հաճախությունը
- կետադրություն չկա.

# Exercise 2A

Գրեք ծրագիր որը

- վերցնում է տվյալ string-ը 'to be or not to be'
- և տպում է ամեն բառի հաճախությունը
- կետադրություն չկա.

to	appears 2 times
be	appears 2 times
or	appears 1 times
not	appears 1 times

# Solution 2A

```
text = 'to be or not to be'
word_list = text.split()
counters = {}
for word in word_list:
    if word in counters:
        counters[word] += 1
    else:
        counters[word] = 1

for word, count in counters.items():
    print('{:8} appears {} times'.format(word, count))
```

# Exercise 2B

Գրեք function `word_count(text)` որը տպում է ամեն բառի հաճախությունը

# Exercise 2B

Գրեք function `word_count(text)` որը տպում է ամեն բառի հաճախությունը

```
>>> text = 'to be or not to be'
>>>
>>> word_count(text)
to          appears 2 times
be          appears 2 times
or          appears 1 times
not         appears 1 times ...
```

# Exercise 2B

Գրեք function `word_count(text)` որը տպում է ամեն  
բառի հաճախությունը

# Exercise 2B

Գրեք function `word_count(text)` որը տպում է ամեն բառի հաճախությունը

```
def word_count(text):  
    'prints frequency of each word in text'  
    word_list = text.split()  
    counters = {}  
    for word in wordList:  
        if word in counters:  
            counters[word] += 1  
        else:  
            counters[word] = 1  
  
    for word in counters:  
        print('{:8} {}'.format(word, counters[word]))
```

# Tuples (immutable lists)

class **tuple** նույնն է ինչ class **list**-ը բայց **immutable** տեսակի

```
>>> lst = ['one', 'two', 3]
>>> lst[2]
3
>>> lst[2] = 'three'
>>> lst
['one', 'two', 'three']
>>> tpl = ('one', 'two', 3)
>>> tpl
('one', 'two', 3)
>>> tpl[2]
3
```



# Tuples (immutable lists)

class **tuple** նույնն է ինչ class **list**-ը բայց **immutable** տեսակի

```
>>> lst = ['one', 'two', 3]
>>> lst[2]
3
>>> lst[2] = 'three'
>>> lst
['one', 'two', 'three']
>>> tpl = ('one', 'two', 3)
>>> tpl
('one', 'two', 3)
>>> tpl[2]
3
>>> tpl[2] = 'three'
```

# Tuples (immutable lists)

class **tuple** նույնն է ինչ class **list**-ը բայց **immutable** տեսակի

```
>>> lst = ['one', 'two', 3]
>>> lst[2]
3
>>> lst[2] = 'three'
>>> lst
['one', 'two', 'three']
>>> tpl = ('one', 'two', 3)
>>> tpl
('one', 'two', 3)
>>> tpl[2]
3
>>> tpl[2] = 'three'
Traceback (most recent call last):
...TypeError: 'tuple' object does
not support item assignment
```

# Tuples (immutable lists)

class **tuple** նույնն է ինչ class **list**-ը բաց **immutable** տեսակի

**tuples** (“immutable lists”)  
կարելի է օգտագործել  
dict()-ում, որի **key**-երը **list**  
են

```
>>> lst = ['one', 'two', 3]
>>> lst[2]
3
>>> lst[2] = 'three'
>>> lst
['one', 'two', 'three']
>>> tpl = ('one', 'two', 3)
>>> tpl
('one', 'two', 3)
>>> tpl[2]
3
>>> tpl[2] = 'three'
Traceback (most recent call last):
...TypeError: 'tuple' object does
not support item assignment
```

# Tuples (immutable lists)

class **tuple** նույնն է ինչ class **list**-ը բայց **immutable** տեսակի

**tuples** (“immutable lists”) կարելի է օգտագործել dict()-ում, որի **key**-երը **list** են

```
>>> employee = {[1,2]:1,
[2,3]:3}
Traceback (most recent
call ...TypeError:
unhashable type: 'list'
>>> employee = {(1,2):1,
(2,3):3}
>>> employee
{(1, 2): 1, (2, 3): 3}
```

```
>>> lst = ['one', 'two', 3]
>>> lst[2]
3
```

```
>>> lst[2] = 'three'
>>> lst
```

```
['one', 'two', 'three']
>>> tpl = ('one', 'two', 3)
>>> tpl
('one', 'two', 3)
>>> tpl[2]
3
```

```
>>> tpl[2] = 'three'
Traceback (most recent call last):
...TypeError: 'tuple' object does
not support item assignment
```

# Exercise 3A

Գրեք ծրագիր որը տպում է հեռախոսի համարը տրված անունի համար եթե անունը **phonebook**-ի մեջ է

```
phonebook = {  
    'Arm' : '56-78-90',  
    'Vlad' : '34-56-78',  
    'Kiazh' : '48-76-54'  
}
```

first name: Arm

56-78-90

first name:

# Solution 3A

```
phonebook = {  
    'Arm': '56-78-90',  
    'Vlad': '34-56-78',  
    'Kiazh': '48-76-54'  
}  
  
while True:  
    person = input('first name: ')  
  
    if person in phonebook:  
        print(phonebook[person])  
    else:  
        print('unknown.')
```

# Exercise 3B

Գրեք function **lookup(phonebook)** որը տպում է հեռախոսի համարը տրված անունի համար

```
>>> phonebook = {  
    'Arm' : '56-78-90',  
    'Vlad' : '34-56-78',  
    'Kiazh' : '48-76-54'  
}  
>>> lookup(phonebook)  
first name: Arm  
56-78-90  
first name:
```

# Exercise 3B

Գրեք function `lookup(phonebook)` որը տպում է հեռախոսի համարը տրված անունի համար

```
def lookup(phonebook):  
    while True:  
        print('first name: ', end='')  
        person = input()  
        if person in phonebook:  
            print(phonebook[person])  
        else:  
            print('unknown.')
```



# Sorting List of Tuples

list of tuples-ը դասավորելու համար

- `sort` method-ը `list`-ի `sort` method-ը կամ
- `sorted` function-ը, որը վերցնում է `sequence` և վերադարձնում է `sorted sequence`

```
>>> t
[('a', 10), ('c', 22), ('b', 1)]
>>> t.sort()
>>> t
[('a', 10), ('b', 1), ('c', 22)]
>>> sorted(t)
[('a', 10), ('b', 1), ('c', 2)]
```

# Sorting dictionaries by keys

dict-երը դասավորելու  
ձևերից մեկը  
օգտագործել `sorted()`  
function-ը

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
dict_items([('a', 10), ('c', 22),
('b', 1)])
>>> type(d.items())
<class 'dict_items'>
```

# Sorting dictionaries by keys

dict-երը դասավորելու  
ձևերից մեկը  
օգտագործել `sorted()`  
function-ը

`sorted()`: դասավորում  
է ըստ `key`-երի

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
dict_items([('a', 10), ('c', 22),
('b', 1)])
>>> type(d.items())
<class 'dict_items'>
>>> t = sorted(d.items())
>>> t
[('a', 10), ('b', 1), ('c', 22)]
>>> type(t)
<class 'list'>
```

# Sorting dictionaries by keys

dict-երը դասավորելու  
ձևերից մեկը  
օգտագործել `sorted()`  
function-ը

`sorted()`: դասավորում  
է ըստ `key`-երի

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
dict_items([('a', 10), ('c', 22),
('b', 1)])
>>> type(d.items())
<class 'dict_items'>
>>> t = sorted(d.items())
>>> t
[('a', 10), ('b', 1), ('c', 22)]
>>> type(t)
<class 'list'>
>>> for k,v in sorted(d.items()):
        print(k,v)
```

```
a 10
b 1
c 22
>>>
```

# Sorting dictionaries by values

Չնեքից մեկը

**dict**-ից կառուցել **list** of **tuples**,  
որտեղ tuple is (value, key)

```
>>> d = {'a':10, 'b':1, 'c':22}  
>>> lst = list()
```

# Sorting dictionaries by values

Չնեքից մեկը

**dict**-ից կառուցել **list of tuples**,  
որտեղ tuple is (value,key)

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> lst = list()
>>> for k,v in d.items():
>>>     lst.append((v,k))

>>> lst
[(10, 'a'), (22, 'c'), (1, 'b')]
```

# Sorting dictionaries by values

Չևերից մեկը

**dict**-ից կառուցել **list of tuples**,  
որտեղ tuple is (value,key)

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> lst = list()
>>> for k,v in d.items():
>>>     lst.append((v,k))

>>> lst
[(10, 'a'), (22, 'c'), (1, 'b')]
>>> lst.sort(reverse=True)
>>> lst
[(22, 'c'), (10, 'a'), (1, 'b')]
>>>
```

# Exercise 4A

Գրեք ծրագիր որը

- բացում է romeo.txt file-ը
- ցիկլի մեջ կարդում է ամեն տողը
- և բառերը դասավորում է dict-ի մեջ
- վերջում տպում է top 10 բառերի լիստը

```
>>>
```

```
I          60
```

```
to         31
```

```
the        30
```

```
thou       29
```

```
JULIET     28
```

```
ROMEO      27
```

```
that       23
```

```
my         22
```

```
and        22
```

```
a          22
```

```
>>>
```



# Exercise 4A

Գրեք ծրագիր որը

- բացում է romeo.txt file-ը
- ցիկլի մեջ կարդում է ամեն տողը
- և բառերը դասավորում է dict-ի մեջ
- վերջում տպում է top 10 բառերի լիստը

HINT :

- դասավորել dict-ը list-ում
- որտեղ անդամները tuple-ներ են
- օգտագործել list-ի sort method-ը

```
>>>
```

I	60
to	31
the	30
thou	29
JULIET	28
ROMEO	27
that	23
my	22
and	22
a	22

```
>>>
```

# Solution 4A

```
filename = 'romeo.txt'
infile = open(filename)
counts = dict()
for line in infile:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0 ) + 1
infile.close()

lst = []
for key, val in counts.items():
    lst.append( (val, key) )
lst.sort(reverse=True)
for val, key in lst[:10] :
    print(key, '\t', val)
```

# Exercise 4B

Գրեք function topten() որը:

- բացում է romeo.txt file-ը
- ցիկլի մեջ կարդում է ամեն տողը
- և բառերը դասավորում է dict-ի մեջ
- վերջում տպում է top 10 բառերի լիստը

```
>>>
topten("romeo.txt")
I            60
to           31
the          30
thou         29
JULIET       28
ROMEO        27
that         23
my           22
and          22
a            22
>>>
```

# Exercise 4B

Գրեք function `topten()` որը:

- բացում է `romeo.txt` file-ը
- ցիկլի մեջ կարդում է ամեն տողը
- և բառերը դասավորում է `dict`-ի մեջ
- վերջում տպում է top 10 բառերի լիստը

HINT :

- դասավորել `dict`-ը `list`-ում
- որտեղ անդամները `tuple`-ներ են
- օգտագործել `list`-ի `sort method`-ը

```
>>>
topten("romeo.txt")
I            60
to           31
the          30
thou         29
JULIET       28
ROMEO        27
that         23
my           22
and          22
a            22
>>>
```

# Exercise 4B

```
def topten(filename):  
    #infile = open(filename)  
    #empty dict  
    #for each line  
        #split into words  
        #for each word  
            #add to dict  
    #infile.close()  
  
    #empty list  
    #for key, val in dict  
        #append to list (key,val)  
    #sort the list  
    #for (val,key) in first ten  
        #print  
  
filename = 'romeo.txt'  
topten(filename)
```

# Exercise 4B

```
def topten(filename):
    infile = open(filename)
    counts = dict()
    for line in infile:
        words = line.split()
        for word in words:
            counts[word] = counts.get(word, 0 ) + 1
    infile.close()

    lst = []
    for key, val in counts.items():
        lst.append( (val, key) )
    lst.sort(reverse=True)
    for val, key in lst[:10] :
        print(key, '\t', val)

filename = 'romeo.txt'
topten(filename)
```

# Sets

set : mathematical set : **mutable**

- an unordered collection of non-identical items
- տարբեր անդամների չդասավորված հավաքածու

# Sets

set : mathematical set : **mutable**

- an unordered collection of non-identical items
- տարբեր անդամների չդասավորված հավաքածու

```
>>> ages = {28, 25, 22}
```

curly braces  
(ձևավոր պակագծեր)





# Sets

set : mathematical set : **mutable**

- an unordered collection of non-identical items
- տարբեր անդամների չդասավորված հավաքածու

```
>>> ages = {28, 25, 22}
```

```
>>> ages
```

```
{25, 28, 22}
```

curly braces

(ձևավոր պակագծեր)

# Sets

set : mathematical set : **mutable**

- an unordered collection of non-identical items
- տարբեր անդամների չդասավորված հավաքածու

```
>>> ages = {28, 25, 22}
```

```
>>> ages
```

```
{25, 28, 22}
```

```
>>> type(ages)
```

```
<class 'set'>
```

curly braces  
(ձևավոր պակագծեր)



# Sets

set : mathematical set : **mutable**

- an unordered collection of non-identical items
- տարբեր անդամների չդասավորված հավաքածու

```
>>> ages = {28, 25, 22}
```

```
>>> ages
```

```
{25, 28, 22}
```

```
>>> type(ages)
```

```
<class 'set'>
```

```
>>> ages2 = {22, 23, 22, 23, 25}
```

```
>>> ages2
```

```
{25, 22, 23}
```

curly braces  
(ձևավոր պակագծեր)

duplicates removed

# Sets

set : mathematical set : **mutable**

- an unordered collection of non-identical items
- տարբեր անդամների չդասավորված հավաքածու

```
>>> ages = {28, 25, 22}
```

```
>>> ages
```

```
{25, 28, 22}
```

```
>>> type(ages)
```

```
<class 'set'>
```

```
>>> ages2 = {22, 23, 22, 23, 25}
```

```
>>> ages2
```

```
{25, 22, 23}
```

```
>>> lst = [22, 23, 22, 23, 25]
```

```
>>> list(set(lst))
```

```
[25, 22, 23]
```

curly braces  
(ձևավոր պակագծեր)

duplicates removed

order changes

# Sets

set : mathematical set : **mutable**

- an unordered collection of non-identical items
- տարբեր անդամների չդասավորված հավաքածու

```
>>> ages = {28, 25, 22}
```

```
>>> ages
```

```
{25, 28, 22}
```

```
>>> type(ages)
```

```
<class 'set'>
```

```
>>> ages2 = {22, 23, 22, 23, 25}
```

```
>>> ages2
```

```
{25, 22, 23}
```

```
>>> lst = [22, 23, 22, 23, 25]
```

```
>>> list(set(lst))
```

```
[25, 22, 23]
```

curly braces  
(ձևավոր պակագծեր)

duplicates removed

order changes

```
>>> s = set()
>>> type(s)
<class 'set'>
```

Empty set

# Set operators

Operation	Explanation
$s == t$	True if $s$ -ը և $t$ -ն ունեն նույն անդամները
$s != t$	True if $s$ -ը և $t$ -ն չունեն նույն անդամները
$s <= t$	True if $s$ -ը $t$ -ի ենթաբազմություն է
$s < t$	True if $s <= t$ and $s != t$
$s   t$	Returns the union of sets $s$ and $t$ (միավորումը $s$ -ի և $t$ -ի)
$s \& t$	Returns the intersection of sets $s$ and $t$ (հատումը $s$ -ի և $t$ -ի)
$s - t$	Returns the difference between sets $s$ and $t$ (տարբերությունը $s$ -ի և $t$ -ի)
$s \wedge t$	Returns the symmetric difference of sets $s$ and $t$ (սիմետրիկ տարբերությունը)

# Set operators

Operation	Explanation
<code>s == t</code>	True if s-ը և t-ն ունեն նույն անդամները
<code>s != t</code>	True if s-ը և t-ն չունեն նույն անդամները
<code>s &lt;= t</code>	True if s-ը t-ի ենթաբազմություն է
<code>s &lt; t</code>	True if <code>s &lt;= t</code> and <code>s != t</code>
<code>s   t</code>	Returns the union of sets s and t (միավորումը s-ի և t-ի)
<code>s &amp; t</code>	Returns the intersection of sets s and t (հատումը s-ի և t-ի)
<code>s - t</code>	Returns the difference between sets s and t (տարբերությունը s-ի և t-ի)
<code>s ^ t</code>	Returns the symmetric difference of sets s and t (սիմետրիկ տարբերությունը)

```
>>> ages
{28, 25, 22}
>>> ages2
{25, 22, 23}
>>> 28 in ages
True
>>> len(ages2)
3
>>> ages == ages2
False
>>> {22, 25} <
ages2
True
>>> ages <= ages2
False
>>> ages | ages2
{22, 23, 25, 28}
>>> ages & ages2
{25, 22}
>>> ages - ages2
{28}
>>> ages ^ ages2
{28, 23}
```

# Set methods

Operation	Explanation
<code>s.add(item)</code>	add item to set s (ավելացրա անդամ)
<code>s.remove(item)</code>	remove item from set s (ջնջի անդամ)
<code>s.clear()</code>	removes all elements from s (ջնջի բոլոր անդամները)



# Set methods

```
>>> ages
{28, 25, 22}
>>> ages2
{25, 22, 23}
>> ages.add(30)
>>> ages
{25, 28, 30, 22}
>>> ages.remove(25)
>>> ages
{28, 30, 22}
>>> ages.clear()
>>> ages
set()
```

Operation	Explanation
s.add(item)	add item to set s (ավելացրա անդամ)
s.remove(item)	remove item from set s (ջնջի անդամ)
s.clear()	removes all elements from s (ջնջի բոլոր անդամները)

# Set methods

```
>>> ages
{28, 25, 22}
>>> ages2
{25, 22, 23}
>> ages.add(30)
>>> ages
{25, 28, 30, 22}
>>> ages.remove(25)
>>> ages
{28, 30, 22}
>>> ages.clear()
>>> ages
set()
```

Operation	Explanation
s.add(item)	add item to set s (ավելացրա անդամ)
s.remove(item)	remove item from set s (ջնջի անդամ)
s.clear()	removes all elements from s (ջնջի բոլոր անդամները)

sets are **mutable!**

# 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.