Introduction to Programming
Lesson 5

Outline

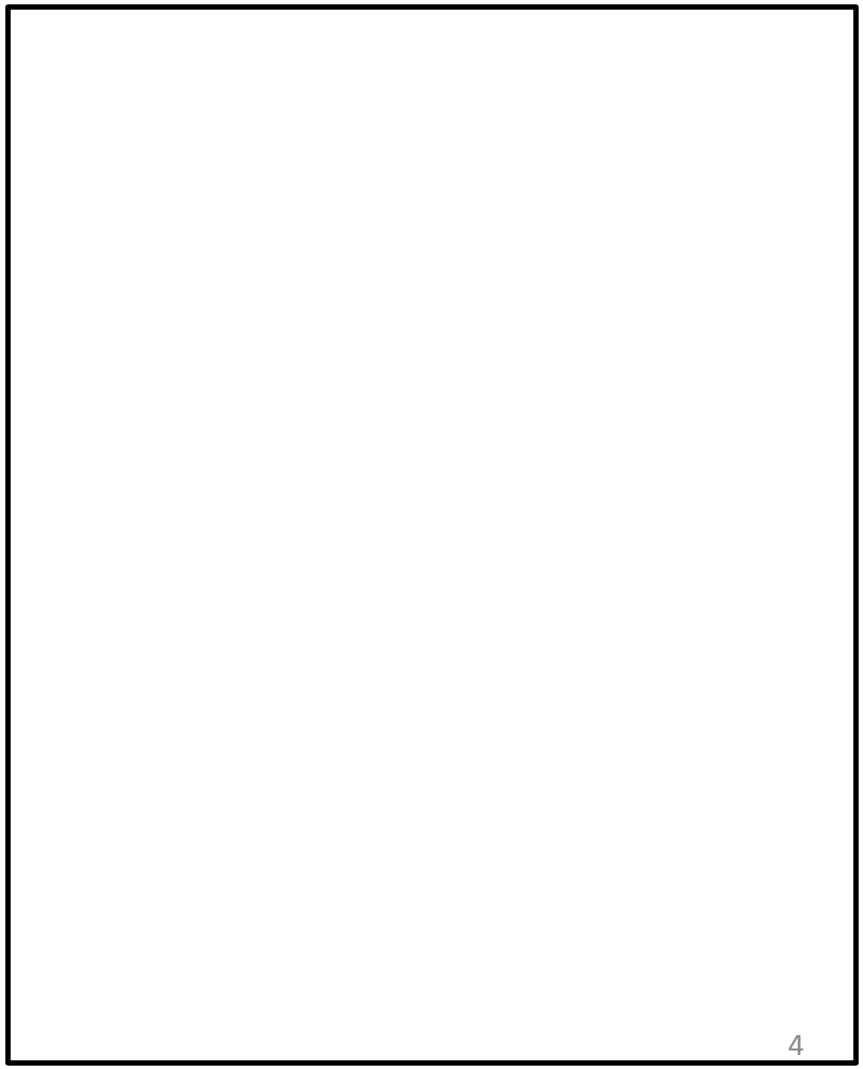
- Errors/Exceptions
- Multi way if
- Patterns
  - Iteration loop pattern
  - Counter loop pattern
  - Accumulator loop pattern
  - Nested loop pattern

Lesson Code

goo.gl/K5CGnb

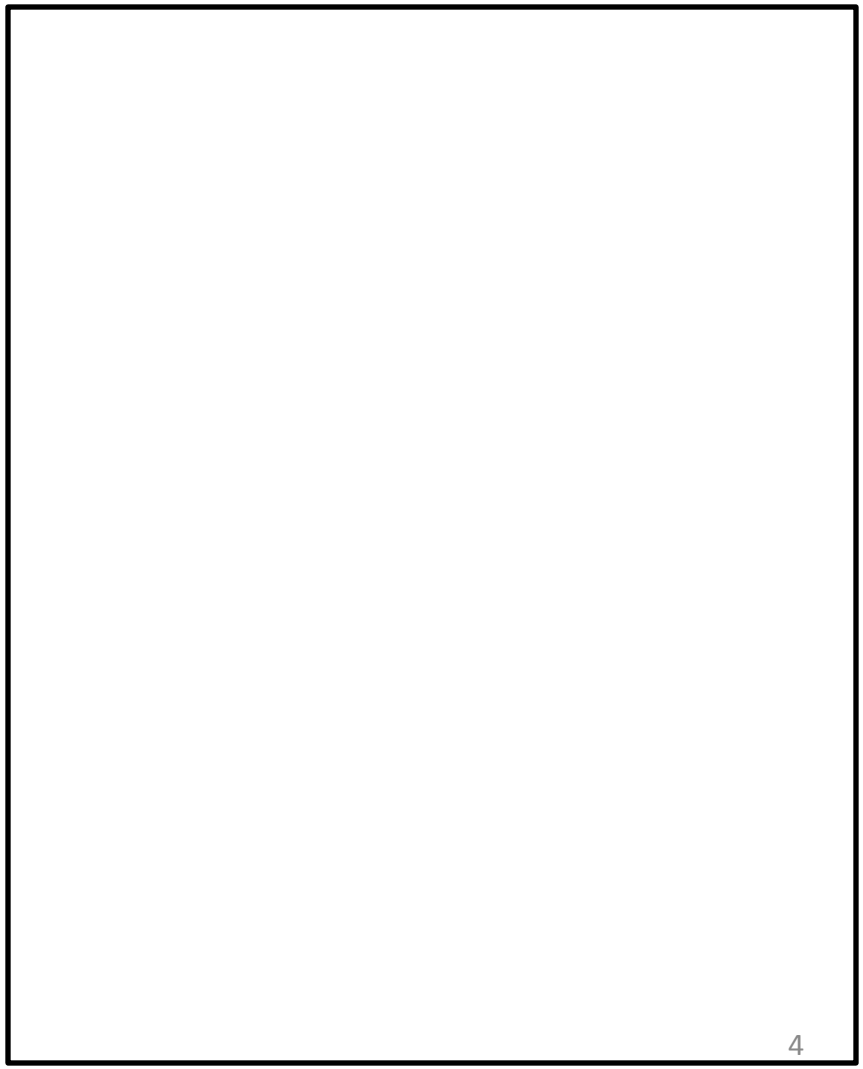
goo.gl/z78iMt

Error-ների տեսակները



Error-ների տեսակները

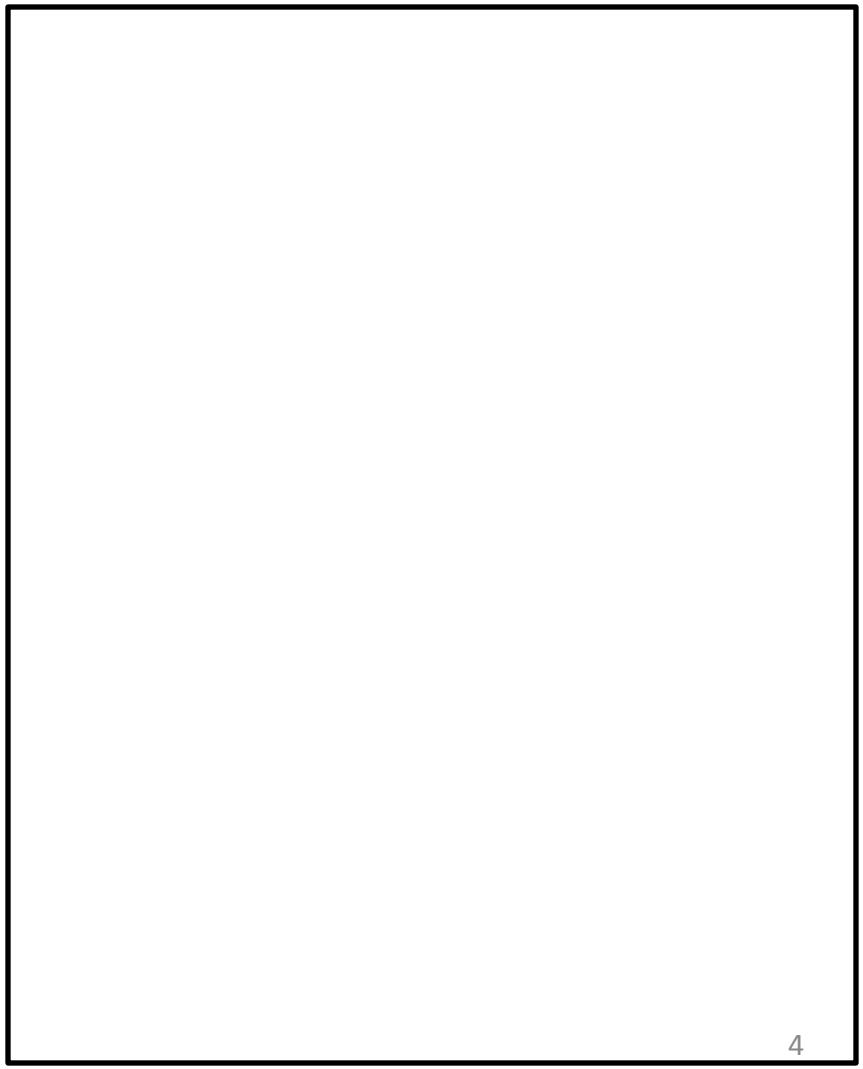
- error-ի 3 տեսակ կա:



Error-ների տեսակները

- error-ի 3 տեսակ կա:
- syntax errors

```
excuse = 'I'm sick'
print("hello world)
a = 3 + 5 7
```



Error-ների տեսակները

- error-ի 3 տեսակ կա:
- syntax errors

```
excuse = 'I'm sick'
print("hello world)
a = 3 + 5 7
```

```
>>> excuse='I'm sick'
SyntaxError: invalid syntax
>>> print("Hello world)
SyntaxError: EOL while
   scanning string literal
>>> a = 3 + 5 7
SyntaxError: invalid syntax
```

Error-ների տեսակները

- error-ի 3 տեսակ կա:
- syntax errors

  excuse = 'I'm sick'

  print("hello world)

  a = 3 + 5 7
- logical errors

  your logic ’5’+’6’ = 11

  computer logic ’5’+’6’ =’56’

```
>>> excuse='I'm sick'
SyntaxError: invalid syntax
>>> print("Hello world)
SyntaxError: EOL while
   scanning string literal
>>> a = 3 + 5 7
SyntaxError: invalid syntax
```

Error-ների տեսակները

- error-ի 3 տեսակ կա:
- syntax errors

  excuse = 'I'm sick'

  print("hello world)

  a = 3 + 5 7
- logical errors

  your logic ’5’+’6’ = 11

  computer logic ’5’+’6’ =’56’
- exceptions

  open(’sample.txt’)

  no ’sample.txt’

```
>>> excuse='I'm sick'
SyntaxError: invalid syntax
>>> print("Hello world)
SyntaxError: EOL while
   scanning string literal
>>> a = 3 + 5 7
SyntaxError: invalid syntax
```

Error-ների տեսակները

- error-ի 3 տեսակ կա:
- syntax errors

  excuse = 'I'm sick'

  print("hello world)

  a = 3 + 5 7
- logical errors

  your logic ’5’+’6’ = 11

  computer logic ’5’+’6’ =’56’
- exceptions

  open(’sample.txt’)

  no ’sample.txt’

```
>>> excuse='I'm sick'
SyntaxError: invalid syntax
>>> print("Hello world)
SyntaxError: EOL while
   scanning string literal
>>> a = 3 + 5 7
SyntaxError: invalid syntax
>>> open('sample.txt')
Traceback (most recent call
   last):
  File "<pyshell#3>", line
   1, in <module>
    open('sample.txt')
FileNotFoundError:
```

Syntax errors

Syntax errors սխալներ են որոնք code-ի սխալ ֆորմատի/գրառման արդյունք են

Syntax errors

Syntax errors սխալներ են որոնք code-ի սխալ ֆորմատի/գրառման արդյունք են

- Այս սխալները գեներացվում են մեքենայական լեզվի թարգմանման ժամանակ և մինչև հաշվարկի սկիզբը

Syntax errors

Syntax errors սխալներ են որոնք code-ի սխալ ֆորմատի/գրառման արդյունք են

- Այս սխալները գեներացվում են մեքենայական լեզվի թարգմանման ժամանակ և մինչև հաշվարկի սկիզբը

```
>>> (3+4] SyntaxError:
invalid syntax
>>> if x == 5
SyntaxError: invalid syntax
>>> print 'hello'
SyntaxError: invalid syntax
>>> lst = [4;5;6]
SyntaxError: invalid syntax
>>> for i in range(10):
print(i)
SyntaxError: expected an indented
block
```

Errors

Ծրագիրը մտնում է փակուղի

```
>>> 3/0
Traceback (most recent call last):
  File "<pyshell#56>", line 1, in
<module>
    3/0
ZeroDivisionError: division by
zero
```

Errors

Ծրագիրը մտնում է փակուղի

```
>>> lst
Traceback (most recent call last):
  File "<pyshell#57>", line 1, in
<module>
    lst
NameError: name 'lst' is not
defined
```

Errors

Ծրագիրը մտնում է փակուղի

```
>>> lst = [12, 13, 14]
>>> lst[3]
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in
<module>
    lst[3]
IndexError: list index out of
range
```

Errors

Ծրագիրը մտնում է փակուղի

```
>>> lst * lst
Traceback (most recent call last):
  File "<pyshell#60>", line 1, in
<module>
    lst * lst
TypeError: can't multiply sequence
by non-int of type 'list’
```

Errors

Ծրագիրը մտնում է փակուղի

```
>>> int('4.5')
Traceback (most recent call last):
  File "<pyshell#61>", line 1, in
<module>
    int('4.5')
ValueError: invalid literal for
int() with base 10: '4.5'
```

Errors

- երբ error է լինում => “error” object է ստեղծվում

Errors

- երբ error է լինում => “error” object է ստեղծվում
  - ունի տեսակ որը հենց error-ի տեսակն է
  - պարունակում է սխալի մասին information

Errors

- երբ error է լինում => “error” object է ստեղծվում
  - ունի տեսակ որը հենց error-ի տեսակն է
  - պարունակում է սխալի մասին information
- The default behavior (լռակյաց վարքագիծը)

Errors

- երբ error է լինում => “error” object է ստեղծվում
  - ունի տեսակ որը հենց error-ի տեսակն է
  - պարունակում է սխալի մասին information
- The default behavior (լռակյաց վարքագիծը)
  - տպել սխալը
  - կանգնեցնել program-ը

Errors

- երբ error է լինում => “error” object է ստեղծվում
  - ունի տեսակ որը հենց error-ի տեսակն է
  - պարունակում է սխալի մասին information
- The default behavior (լռակյաց վարքագիծը)
  - տպել սխալը
  - կանգնեցնել program-ը
- The “error” object-ը անվանում են exception (բացառություն)

Errors

- երբ error է լինում => “error” object է ստեղծվում
  - ունի տեսակ որը հենց error-ի տեսակն է
  - պարունակում է սխալի մասին information
- The default behavior (լռակյաց վարքագիծը)
  - տպել սխալը
  - կանգնեցնել program-ը
- The “error” object-ը անվանում են exception (բացառություն)
- creation of an exception <=> raise an exception

built-in Exception classes

Exception	Explanation
KeyboardInterrupt	բարձրացվում է երբ Ctrl-C էք սեղմում
OverflowError	բարձրացվում է երբ float-ի արժեքը հիշողության մեջ չի տեղավորվում
ZeroDivisionError	բարձրացվում է երբ 0-ի եք բաժանում
IOError	Raised when an I/O operation fails for an I/O-related reason բարձրացվում է Input/Output-ի հետ կապված օպեռացիայի ձախողման արդյունքում
IndexError	բարձրացվում է երբ index-ը թույլատրելիից մեծ է
NameError	բարձրացվում է երբ օգտագործվում է գոյություն չունեցող, չսահմանված անուն
TypeError	բարձրացվում է երբ function-ը կիրառվում է սխալ տեսակի object-ի վրա
ValueError	բարձրացվում է երբ function-ը /օպեռացիան ունի ճիշտ տեսակ բայց սխալ արժեք

Exercise 1

բարձրացրեք հետևյալ exception-ները

KeyboardInterrupt
OverflowError
ZeroDivisionError
IOError
IndexError
NameError
TypeError
ValueError

Exercise 1

բարձրացրեք հետևյալ exception-ները

KeyboardInterrupt
OverflowError
ZeroDivisionError
IOError
IndexError
NameError
TypeError
ValueError

```
>>> raise Exception("I know python!")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in
<module>
    raise Exception("I know python!")
Exception: I know python!
```

Exercise 1

բարձրացրեք հետևյալ exception-ները

```
>>> raise
KeyboardInterrupt("Cntr-C")
Traceback (most recent call
last):...
KeyboardInterrupt: Cntr-C
>>> raise
OverflowError("MyFloat")
Traceback (most recent call
last):...
OverflowError: MyFloat
>>> raise ZeroDivisionError
Traceback (most recent call
last):...
ZeroDivisionError
>>> raise IOError
Traceback (most recent call
last):...
OSError
```

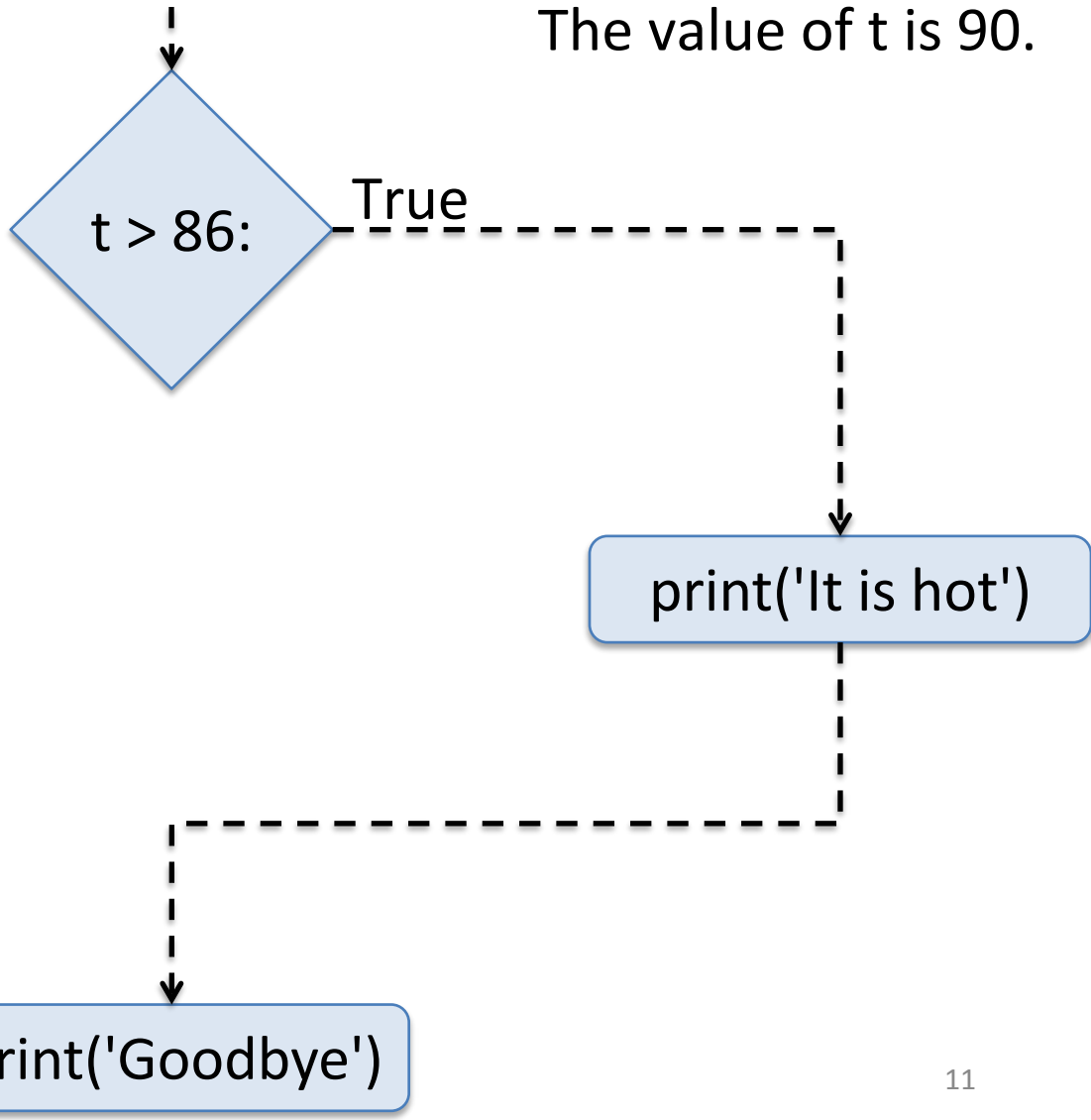
```
>>> raise IndexError
Traceback (most recent call
last):...
IndexError
>>> raise NameError
Traceback (most recent call
last):...
NameError
>>> raise TypeError
Traceback (most recent call
last):...
TypeError
>>> raise ValueError
Traceback (most recent call
last):...
ValueError
```

Multy-way if statement

```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32:
        print('It is cool')
    else:
        print('It is
freezing’)
    print('Goodbye')
```

Multy-way if statement

```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32:
        print('It is cool')
    else:
        print('It is
freezing’)
    print('Goodbye')
```

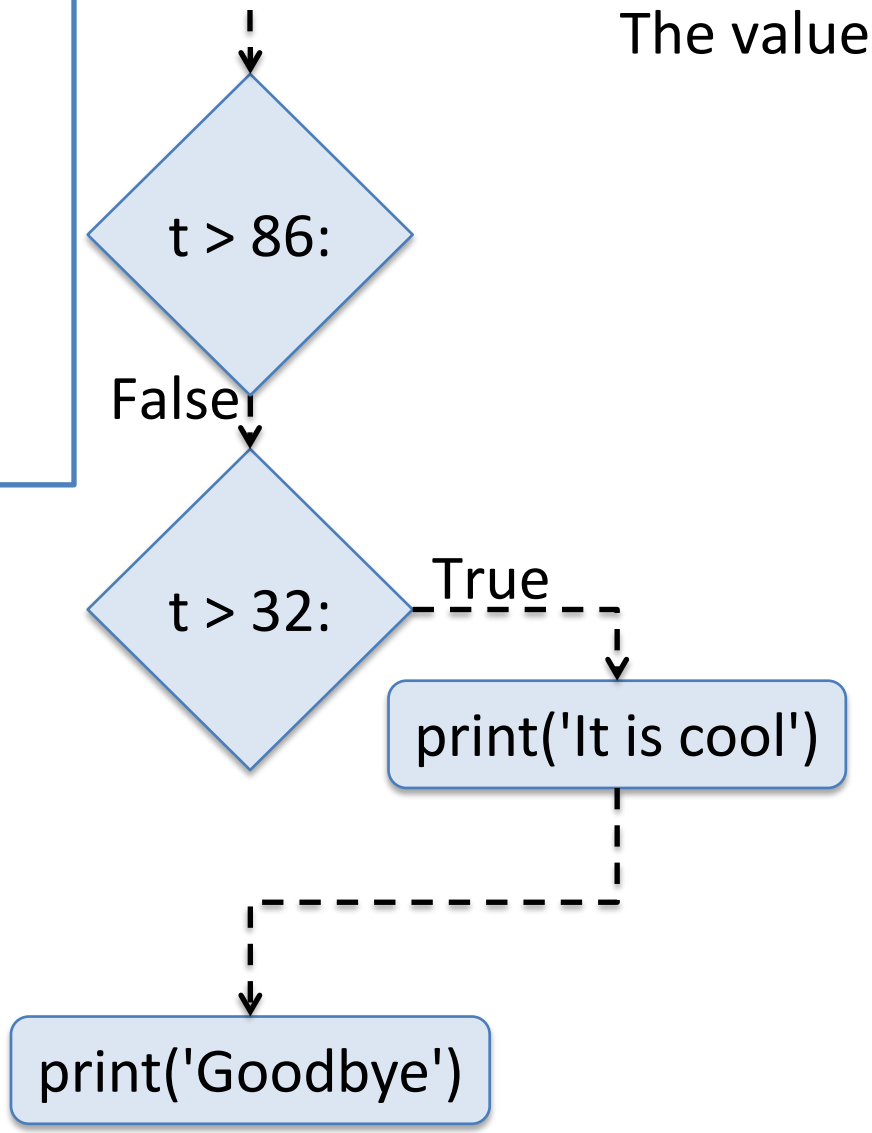


Multy-way if statement

```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32:
        print('It is cool')
    else:
        print('It is
freezing’)
    print('Goodbye')
```

Multy-way if statement

```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32:
        print('It is cool')
    else:
        print('It is
freezing’)
    print('Goodbye')
```



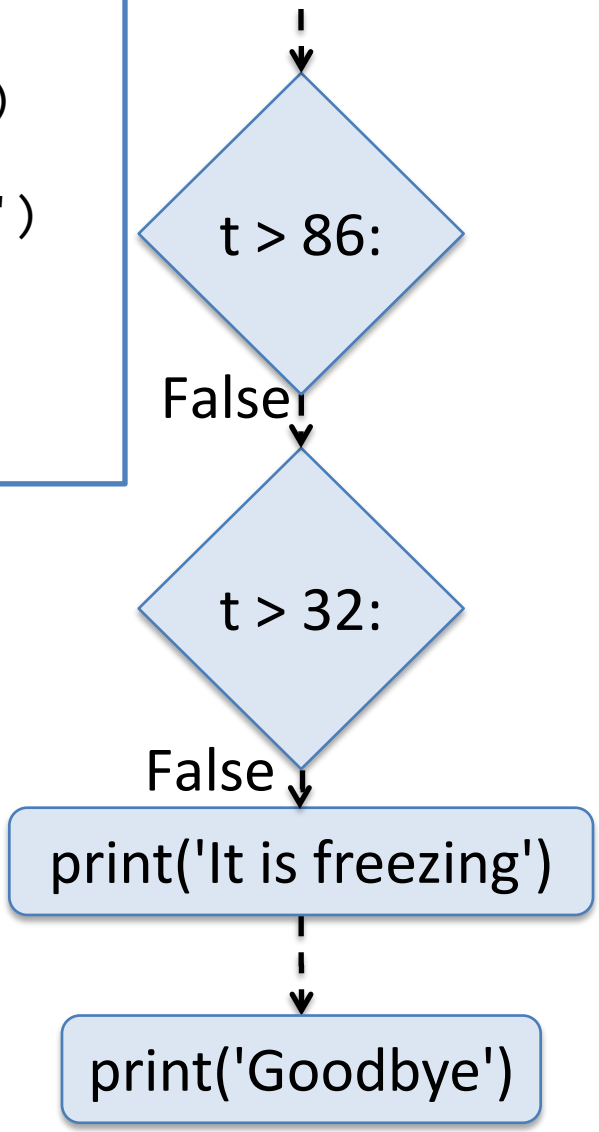
The value of t is 50.

Multy-way if statement

```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32:
        print('It is cool')
    else:
        print('It is
freezing’)
    print('Goodbye')
```

Multy-way if statement

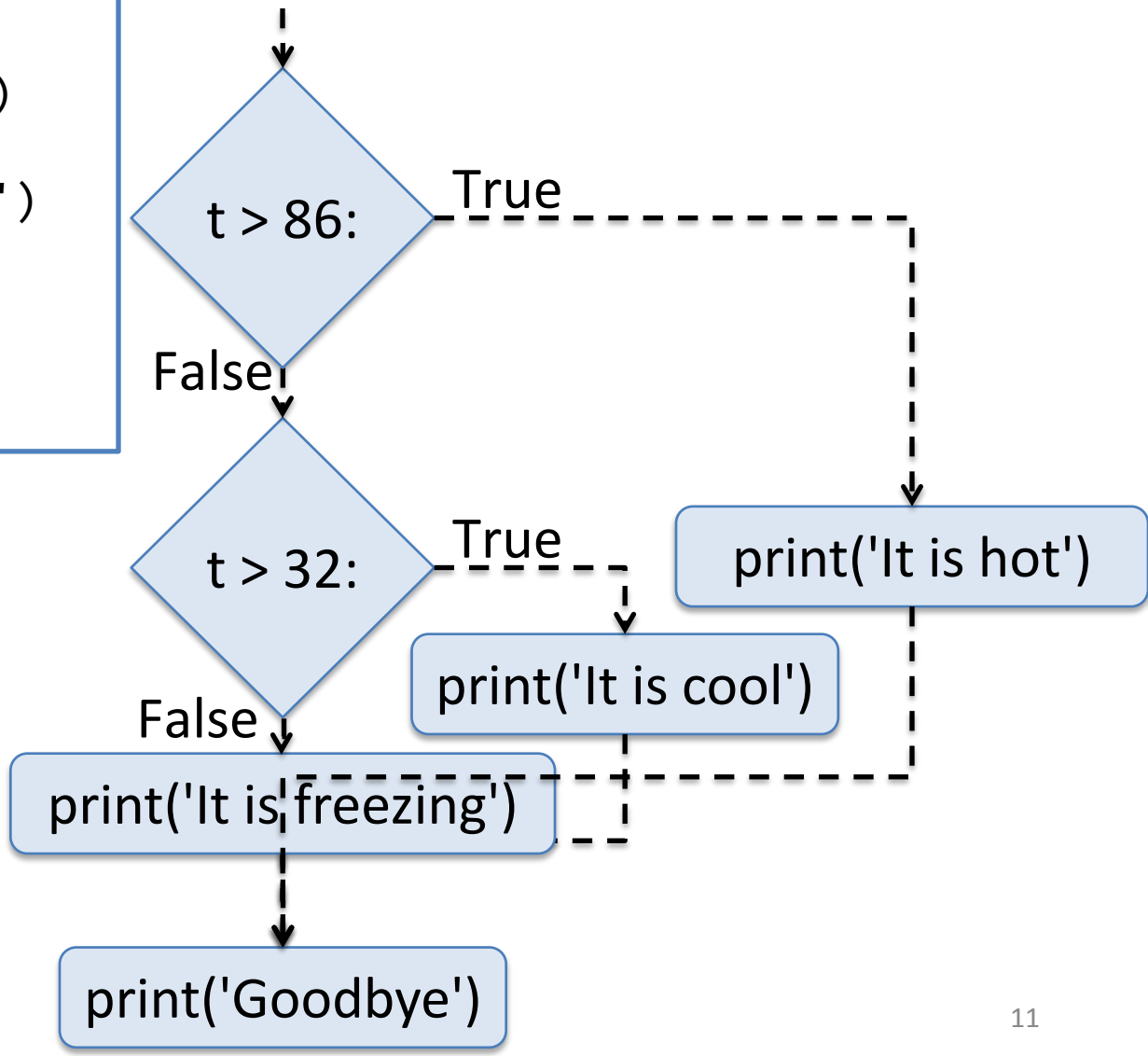
```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32:
        print('It is cool')
    else:
        print('It is
freezing’)
    print('Goodbye')
```



The value of t is 20.

Multy-way if statement

```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32:
        print('It is cool')
    else:
        print('It is
freezing’)
    print('Goodbye')
```



Condition ordering

Ինչն է սխալ temperature() function-ի հետ

```
def temperature(t):
    if t > 32:
        print('It is cool')
    elif t > 86:
        print('It is hot')
    else: # t <= 32
        print('It is freezing')
    print('Goodbye')
```

Condition ordering

Ինչն է սխալ temperature() function-ի հետ

```
def temperature(t):
    if t > 32:
        print('It is cool')
    elif t > 86:
        print('It is hot')
    else: # t <= 32
        print('It is freezing')
    print('Goodbye')
```

պայմանները պետք է լինեն իրարամերժ

Condition ordering

Ինչն է սխալ temperature() function-ի հետ

```
def temperature(t):
    if t > 32:
        print('It is cool')
    elif t > 86:
        print('It is hot')
    else: # t <= 32
        print('It is freezing')
    print('Goodbye')
```

```
def temperature(t):
    if 86 >= t > 32:
        print('It is cool')
    elif t > 86:
        print('It is hot')
    else: # t <= 32
        print('It is freezing')
    print('Goodbye')
```

պայմանները պետք է լինեն իրարամերժ

- ուղղակիորեն (explicitly)

Condition ordering

Ինչն է սխալ temperature() function-ի հետ

```
def temperature(t):
    if t > 32:
        print('It is cool')
    elif t > 86:
        print('It is hot')
    else: # t <= 32
        print('It is freezing')
    print('Goodbye')
```

```
def temperature(t):
    if 86 >= t > 32:
        print('It is cool')
    elif t > 86:
        print('It is hot')
    else: # t <= 32
        print('It is freezing')
    print('Goodbye')
```

պայմանները պետք է լինեն իրարամերժ

- ուղղակիորեն (explicitly)
- անուղղակիորեն (implicitly)

```
def temperature(t):
    if t > 86:
        print('It is hot')
    elif t > 32: # 86 >= t > 32
        print('It is cool')
    else: # t <= 32
        print('It is freezing')
    print('Goodbye')
```

Exercise 2A

Գրեք ծրագիր որը ստանալով անձի բարձրությունը (m) և զանգվածը (kg) հաշվում անձի BMI-ը և տպում է գնահատականը

$$bmi = \frac{m}{h^2}$$

Exercise 2A

Գրեք ծրագիր որը ստանալով անձի բարձրությունը (m) և զանգվածը (kg) հաշվում անձի BMI-ը և տպում է գնահատականը

body mass index:

bmi<18.5 => underweight

bmi>25.0 => overweight

18.5<bmi<25.0 => normal

$$bmi = \frac{m}{h^2}$$

Exercise 2A

Գրեք ծրագիր որը ստանալով անձի բարձրությունը (m) և զանգվածը (kg) հաշվում անձի BMI-ը և տպում է գնահատականը

body mass index:

bmi<18.5 => underweight

bmi>25.0 => overweight

18.5<bmi<25.0 => normal

$$bmi = \frac{m}{h^2}$$

```
bmi = weight/height**2

if bmi < 18.5:
        print('Underweight')
elif bmi < 25:
        print('Normal')
else: # bmi >= 25
        print('Overweight')
```

Exercise 2B

Գրեք function bmi() որը:

- ընդունում է անձի բարձրությունը (m) և զանգվածը (kg)
- հաշվում անձի BMI-ը և տպում է գնահատականը

Exercise 2B

Գրեք function bmi() որը:

- ընդունում է անձի բարձրությունը (m) և զանգվածը (kg)
- հաշվում անձի BMI-ը և տպում է գնահատականը

```
>>> bmi(60, 1.75)
Normal
>>> bmi(40, 1.75)
Underweight
>>> bmi(100, 1.75)
Overweight
```

Exercise 2B

Գրեք function bmi() որը:

- ընդունում է անձի բարձրությունը (m) և զանգվածը (kg)
- հաշվում անձի BMI-ը և տպում է գնահատականը

```
>>> bmi(60, 1.75)
Normal
>>> bmi(40, 1.75)
Underweight
>>> bmi(100, 1.75)
Overweight
```

```
def bmi(weight, height):
  'prints BMI report'

  bmi = weight/height**2

  if bmi < 18.5:
    print('Underweight')
  elif bmi < 25:
    print('Normal')
  else: # bmi >= 25
    print('Overweight')
```

Iteration

for loop statement:

```
for item in <sequence>:
    <indented code-block>
<non-indented code-block>
```

Iteration

for loop statement:

```
for item in <sequence>:
    <indented code-block>
<non-indented code-block>
```

<indented code-block> հաշվվում է <sequence>–ի ամեն անդամի(item) համար

Iteration

for loop statement:

```
for item in <sequence>:
    <indented code-block>
<non-indented code-block>
```

<indented code-block> հաշվվում է <sequence>–ի ամեն անդամի(item) համար

- եթե <sequence>-ը string է => item-ները char-եր են

Iteration

for loop statement:

```
for item in <sequence>:
    <indented code-block>
<non-indented code-block>
```

<indented code-block> հաշվվում է <sequence>–ի ամեն անդամի(item) համար

- եթե <sequence>-ը string է => item-ները char-եր են
- եթե <sequence>-ը list է => item-ները list-ում գտնվող object-ներն են

Iteration

for loop statement:

```
for item in <sequence>:
    <indented code-block>
<non-indented code-block>
```

<indented code-block> հաշվվում է <sequence>–ի ամեն անդամի(item) համար

- եթե <sequence>-ը string է => item-ները char-եր են
- եթե <sequence>-ը list է => item-ները list-ում գտնվող object-ներն են

<non-indented code-block> հաշվվում է <sequence>-ի բոլոր անդամների կանչվելուց հետո

Iteration

for loop statement:

```
for item in <sequence>:
    <indented code-block>
<non-indented code-block>
```

<indented code-block> հաշվվում է <sequence>–ի ամեն անդամի(item) համար

- եթե <sequence>-ը string է => item-ները char-եր են
- եթե <sequence>-ը list է => item-ները list-ում գտնվող object-ներն են

<non-indented code-block> հաշվվում է <sequence>-ի բոլոր անդամների կանչվելուց հետո

for loop –ի օգտագործման տարբեր ձևեր/մոդելներ/patterns/шаблон-ներ կան

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

```
>>> name = 'Apple'
>>> for char in name:
        print(char)

A
p
p
l
e
>>>
```

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

```
>>> name = 'Apple'
>>> for char in name:
        print(char)

A
p
p
l
e
>>>
```

```
for word in ['stop', 'desktop',
'post', 'top']:
    if 'top' in word:
        print(word)

stop
desktop
top
>>>
```

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

text file-ի իտեռացիան ըստ տառերի

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

text file-ի իտեռացիան ըստ տառերի

```
infile = open('test.txt')
content = infile.read()
for char in content:
    print(char, end='')
```

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

text file-ի իտեռացիան ըստ տառերի

```
infile = open('test.txt')
content = infile.read()
for char in content:
    print(char, end='')
```

text file-ի իտեռացիան ըստ տողերի

iteration loop pattern

EN: explicit sequence iteration

AM: բացահայտ հաջորդականության իտերացիա

text file-ի իտեռացիան ըստ տառերի

```
infile = open('test.txt')
content = infile.read()
for char in content:
    print(char, end='')
```

text file-ի իտեռացիան ըստ տողերի

```
infile = open('test.txt')
lines = infile.readlines()
for line in lines:
    print(line, end='')
```

Counter loop pattern

```
>>> for i in range(10):
        print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
```

Counter loop pattern

```
>>> for i in range(10):
        print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
```

```
>>> for i in range(7, 100, 17):
        print(i, end=' ')

7 24 41 58 75 92
```

Counter loop pattern

```
>>> for i in range(10):
        print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
```

```
>>> for i in range(7, 100, 17):
        print(i, end=' ')

7 24 41 58 75 92
```

```
>>> for i in range(len('world')):
      print(i, end=' ')

0 1 2 3 4
```

Counter loop pattern

```
>>> for i in range(10):
        print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
```

```
>>> for i in range(7, 100, 17):
        print(i, end=' ')

7 24 41 58 75 92
```

```
>>> for i in range(len('world')):
      print(i, end=' ')

0 1 2 3 4
```

այս օրինակները նկարագրում են counter loop pattern-ը

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

animal = 'cat'

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

animal = 'dog'

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

animal = 'fish'

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

animal = 'bird'

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

i = 0 print(pets[0])

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

i = 1 print(pets[1])

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

i = 2 print(pets[2])

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

i = 3 print(pets[3])

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

explicit

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

Counter loop pattern

EN: implicit sequence iteration of numbers

AM: թաքնված հաջորդականության իտերացիա թվերով

explicit

implicit

```
>>> pets = ['cat', 'dog', 'fish', 'bird']
```

```
>>> for animal in pets:
        print(animal, end=' ')
cat dog fish bird
```

```
>>> for i in range(len(pets)):
        print(pets[i], end=' ')
cat dog fish bird
```

Counter loop pattern

թաքնված հաջորդականության իտերացիա թվերով...
ինչի համար կյանքը բարդացնել?

Counter loop pattern

թաքնված հաջորդականության իտերացիա թվերով...
ինչի համար կյանքը բարդացնել?

Գրենք function `is_sorted(num_list)` որը:

Counter loop pattern

թաքնված հաջորդականության իտերացիա թվերով...
ինչի համար կյանքը բարդացնել?

Գրենք function `is_sorted(num_list)` որը:

- համեմատում է list-ի անդամները

Counter loop pattern

թաքնված հաջորդականության իտերացիա թվերով...
ինչի համար կյանքը բարդացնել?

Գրենք function `is_sorted(num_list)` որը:

- համեմատում է list-ի անդամները
- returns True եթե sequence-ը աճող է

Counter loop pattern

թաքնված հաջորդականության իտերացիա թվերով...
ինչի համար կյանքը բարդացնել?

Գրենք function `is_sorted(num_list)` որը:

- համեմատում է list-ի անդամները
- returns True եթե sequence-ը աճող է
- returns False հակառակ դեպքում

Counter loop pattern

թաքնված հաջորդականության իտերացիա թվերով...
ինչի համար կյանքը բարդացնել?

Գրենք function `is_sorted(num_list)` որը:

- համեմատում է list-ի անդամները
- returns True եթե sequence-ը աճող է
- returns False հակառակ դեպքում

```
>>> is_sorted([2, 4, 6, 8, 10])
True
>>> is_sorted([2, 4, 6, 3, 10])
False
```

Counter loop pattern

թաքնված հաջորդականության իտերացիա թվերով...
ինչի համար կյանքը բարդացնել?

Գրենք function `is_sorted(num_list)` որը:

- համեմատում է list-ի անդամները
- returns True եթե sequence-ը աճող է
- returns False հակառակ դեպքում

```
>>> is_sorted([2, 4, 6, 8, 10])
True
>>> is_sorted([2, 4, 6, 3, 10])
False
```

```
Hint: lst[i]>lst[i+1]=>false
```

Counter loop pattern

```
def is_sorted(lst):
    for i in range(0, len(lst)-1):
    if lst[i] > lst[i+1]:
            return False
    return True

t = is_sorted([2, 4, 6, 8, 10])
print(t)

f = is_sorted([2, 4, 6, 3, 10])
print(f)
```

checksorted.py

Exercise 3

Գրեք function `is_arithmetic()` որը:

- վերցնում է թվերի list
- returns True եթե list-ը arithmetic sequence է
- False otherwise

```
>>> is_arithmetic([3, 6, 9, 12, 15])
True
>>> is_arithmetic([3, 6, 9, 11, 14])
False
>>> is_arithmetic([3])
True
```

Exercise 3

```
def is_arithmetic(lst):
    if len(lst) < 2:
        return True

    diff = lst[1] - lst[0]
    for i in range(1, len(lst)-1):
        if lst[i+1] - lst[i] != diff:
            return False

    return True

print(is_arithmetic([3, 6, 9, 11, 14]))
```

arithmetic.py

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
```

lst = [3 , 2 , 7 , 1 , 9]

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
```

lst = `[ 3 , 2 , 7 , 1 , 9 ]`

res = 0

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = `[ 3 , 2 , 7 , 1 , 9 ]`

res = 0

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

res = 0

num = 3

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 3

res = 0

res = res + num (= 3)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 2

res = 0

res = res + num (= 3)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 2

res = 0

res = res + num (= 3)

res = res + num (= 5)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 7

res = 0

res = res + num (= 3)

res = res + num (= 5)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 7

res = 0

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

res = 0

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

num = 1

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 1

res = 0

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

res = res + num (= 13)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

res = 0

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

res = res + num (= 13)

num = 9

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 9

res = 0

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

res = res + num (= 13)

res = res + num (= 22)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res = res + num
>>> res
22
```

lst = [3 , 2 , 7 , 1 , 9]

res = 0

accumulator

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

res = res + num (= 13)

res = res + num (= 22)

num = 9

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res += num
>>> res
22
```

lst = [3 , 2 , 7 , 1 , 9]

num = 9

accumulator

res = 0

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

res = res + num (= 13)

res = res + num (= 22)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի գումարը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 0
>>> for num in lst:
    res += num
>>> res
22
```

shorthand notation

lst = [3 , 2 , 7 , 1 , 9]

res = 0

accumulator

res = res + num (= 3)

res = res + num (= 5)

res = res + num (= 12)

res = res + num (= 13)

res = res + num (= 22)

num = 9

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

lst = `[ 3 , 2 , 7 , 1 , 9 ]`

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = `[ 3 , 2 , 7 , 1 , 9 ]`

res = 1

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

res = 1

num = 3

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 3

res = 1

res *= num (= 3)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = `[ 3 , 2 , 7 , 1 , 9 ]`

num = `2`

res = 1

res *= num (= 3)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 2

res = 1

res *= num (= 3)

res *= num (= 6)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 7

res = 1

res *= num (= 3)

res *= num (= 6)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 7

res = 1

res *= num (= 3)

res *= num (= 6)

res *= num (= 42)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

res = 1

res *= num (= 3)

res *= num (= 6)

res *= num (= 42)

num = 1

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 1

res = 1

res *= num (= 3)

res *= num (= 6)

res *= num (= 42)

res *= num (= 42)

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

res = 1

res *= num (= 3)

res *= num (= 6)

res *= num (= 42)

res *= num (= 42)

num = 9

Accumulator loop pattern

Accumulate (կուտակել) ամեն ցիկլում

For example: list-ի թվերի պատիկը

```
>>> lst = [3, 2, 7, 1, 9]
>>> res = 1
>>> for num in lst:
    res *= num
```

lst = [3 , 2 , 7 , 1 , 9]

num = 9

res = 1

res *= num (= 3)

res *= num (= 6)

res *= num (= 42)

res *= num (= 42)

res *= num (= 378)

Short hand operators

operator	example
+=	c +=a <=> c = c + a
-=	c -=a <=> c = c – a
*=	c *=a <=> c = c * a
/=	c /=a <=> c = c / a
%=	c %=a <=> c = c % a
**=	c **=a <=> c = c ** a
//=	c //=a <=> c = c // a

Exercise 4

Գրեք function `factorial()` որը:

- վերցնում է դրական integer n որպես input
- returns n!

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \ldots \times 3 \times 2 \times 1$$
$$0! = 1$$

```
>>> factorial(0)
1
>>> factorial(1)
1
>>> factorial(3)
6
>>> factorial(6)
720
```

Solution 4

```
def factorial(n):
    '''returns n!
    >>> factorial(6)
    720
    '''
    res = 1
    for i in range(2, n+1):
        res *= i
    return res

factorial(3)
```

ex4.py

Exercise 5A

Գրեք ծրագիր որը:

- տվյալ ֆռազայից `'To Be Honest'` ստանում է acronym `'TBH'` և տպում այն
- hint: use `split()` and `upper()` methods

Exercise 5A

Գրեք ծրագիր որը:

- տվյալ ֆռազայից `'To Be Honest'` ստանում է acronym `'TBH'` և տպում այն
- hint: use `split()` and `upper()` methods

```
phrase = 'To Be Honest'
words = phrase.split()

res = ''
for word in words:
    res = res + word[0]

print(res.upper())
```

Exercise 5B

Գրեք function acronym() որը:

- վերցնում է ֆռազա (string) as input
- վերադարձնում է ֆռազայի acronym-ը

```
>>> acronym('Random access memory')
'RAM'
>>> acronym("GNU's not UNIX")
'GNU'
```

Solution 5B

```
def acronym(phrase):
    '''returns the acronym'''
    words = phrase.split()
    res = ''
    for word in words:
        res = res + word[0]
    return res.upper()

print(acronym('too long; didn\'t read'))
```

Exercise 6

Գրեք ծրագիր որը:

- վերցնում է դրական integer-ի և տպում նրա բաժանարարների լիստը
- **6**-ի բաժանարարներն են `[1, 2, 3, 6]`

Exercise 6

Գրեք ծրագիր որը:

- վերցնում է դրական integer-ի և տպում նրա բաժանարարների լիստը
- **6**-ի բաժանարարներն են `[1, 2, 3, 6]`

```
n = 6
res = [] # accumulator
for i in range(1, n+1):
    if n % i == 0:
        res.append(i)

print(res)
```

Exercise 6

Գրի function divisors() որը:

- վերցնում է դրական integer n
- returns n-ի բաժանարարների list

```
>>> divisors(1)
[1]
>>> divisors(6)
[1, 2, 3, 6]
>>> divisors(11)
[1, 11]
```

Solution 6B

```
def divisors(n):
    'return the list of divisors of n'

    res = []    # accumulator
    for i in range(1, n+1):
        if n % i == 0:
            res.append(i)
    return res

divisors(6) # call
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
def nested(n):
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
>>>
```

```
def nested(n):
    for i in range(n):
        print(i, end=' ')
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>> nested(n)
0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
>>>
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
    print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ’)
        print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

```
def nested2(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
    print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

երբ j = 0 i = 0 print 0

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

```
def nested2(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
    print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

երբ j = 0 i = 0 print 0

երբ j = 1 i = 0,1 print 0 1

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

```
def nested2(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
    print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

երբ j = 0 i = 0 print 0

երբ j = 1 i = 0,1 print 0 1

երբ j = 2 i = 0,1,2 print 0 1 2

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

```
def nested2(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
    print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

երբ j = 0 i = 0 print 0

երբ j = 1 i = 0,1 print 0 1

երբ j = 2 i = 0,1,2 print 0 1 2

երբ j = 3 i = 0,1,2,3 print 0 1 2 3

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

```
def nested2(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
    print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

երբ j = 0 i = 0 print 0

երբ j = 1 i = 0,1 print 0 1

երբ j = 2 i = 0,1,2 print 0 1 2

երբ j = 3 i = 0,1,2,3 print 0 1 2 3

երբ j = 4 i = 0,1,2,3,4 print 0 1 2 3 4

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

```
def nested2(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
    print()
```

Nested loop pattern

EN: Nesting a loop inside another

AM: Ցիկլը ցիկլի մեջ դնել

```
>>> n = 5
>>>
nested(n)
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

```
>>> n = 5
>>>
nested2(n)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

երբ j = 0 i = 0 print 0

երբ j = 1 i = 0,1 print 0 1

երբ j = 2 i = 0,1,2 print 0 1 2

երբ j = 3 i = 0,1,2,3 print 0 1 2 3

երբ j = 4 i = 0,1,2,3,4 print 0 1 2 3 4

```
def nested(n):
    for j in range(n):
        for i in range(n):
            print(i, end=' ')
        print()
```

```
def nested2(n):
    for j in range(n):
        for i in range(j+1):
            print(i, end=' ')
        print()
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

3	5	7	9
0	2	1	6
3	8	3	1

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[3, 5, 7, 9] =

3	5	7	9
0	2	1	6
3	8	3	1

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[3, 5, 7, 9] =

[0, 2, 1, 6] =

3	5	7	9
0	2	1	6
3	8	3	1

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[3, 5, 7, 9] =
[0, 2, 1, 6] =
[3, 8, 3, 1] =

3	5	7	9
0	2	1	6
3	8	3	1

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
  [0, 2, 1, 6]
  [3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
       [0,2,1,6],
       [3,8,3,1]]
>>>
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
  [0, 2, 1, 6]
  [3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
        [0,2,1,6],
        [3,8,3,1]]
>>> lst
[[3, 5, 7, 9],
[0, 2, 1, 6],
[3, 8, 3, 1]]
>>>
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
[0, 2, 1, 6]
[3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
        [0,2,1,6],
        [3,8,3,1]]
>>> lst
[[3, 5, 7, 9],
[0, 2, 1, 6],
[3, 8, 3, 1]]
>>> lst[0]
[3, 5, 7, 9]
>>>
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
[0, 2, 1, 6]
[3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
        [0,2,1,6],
        [3,8,3,1]]
>>> lst
[[3, 5, 7, 9],
[0, 2, 1, 6],
[3, 8, 3, 1]]
>>> lst[0]
[3, 5, 7, 9]
>>> lst[1]
[0, 2, 1, 6]
>>>
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
[0, 2, 1, 6]
[3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
        [0,2,1,6],
        [3,8,3,1]]
>>> lst
[[3, 5, 7, 9],
[0, 2, 1, 6],
[3, 8, 3, 1]]
>>> lst[0]
[3, 5, 7, 9]
>>> lst[1]
[0, 2, 1, 6]
>>> lst[2]
[3, 8, 3, 1]
>>>
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
[0, 2, 1, 6]
[3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
        [0,2,1,6],
        [3,8,3,1]]
>>> lst
[[3, 5, 7, 9],
[0, 2, 1, 6],
[3, 8, 3, 1]]
>>> lst[0]
[3, 5, 7, 9]
>>> lst[1]
[0, 2, 1, 6]
>>> lst[2]
[3, 8, 3, 1]
>>> lst[0][0]
3
>>>
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
[0, 2, 1, 6]
[3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
        [0,2,1,6],
        [3,8,3,1]]
>>> lst
[[3, 5, 7, 9],
[0, 2, 1, 6],
[3, 8, 3, 1]]
>>> lst[0]
[3, 5, 7, 9]
>>> lst[1]
[0, 2, 1, 6]
>>> lst[2]
[3, 8, 3, 1]
>>> lst[0][0]
3
>>> lst[1][2]
1
>>>
```

Երկչափ 2D list-եր

list [3, 5, 7, 9] <=> 1-D table

ինչպես ներկայացնել 2-D table?

[[3, 5, 7, 9]
[0, 2, 1, 6]
[3, 8, 3, 1]]

	0	1	2	3
0	3	5	7	9
1	0	2	1	6
2	3	8	3	1

2-D table <=> 1-D table-ների list է

```
>>> lst =
[[3,5,7,9],
        [0,2,1,6],
        [3,8,3,1]]
>>> lst
[[3, 5, 7, 9],
[0, 2, 1, 6],
[3, 8, 3, 1]]
>>> lst[0]
[3, 5, 7, 9]
>>> lst[1]
[0, 2, 1, 6]
>>> lst[2]
[3, 8, 3, 1]
>>> lst[0][0]
3
>>> lst[1][2]
1
>>> lst[2][0]
3
>>>
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def print2D(t):
    'prints values in 2D list t as a 2D table'
```

```
>>> table = [[3, 5, 7, 9],
             [0, 2, 1, 6],
             [3, 8, 3, 1]]
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def print2D(t):
    'prints values in 2D list t as a 2D table'
    # for every row of t
        # for every object in the row
            # print object
```

```
>>> table = [[3, 5, 7, 9],
             [0, 2, 1, 6],
             [3, 8, 3, 1]]
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def print2D(t):
    'prints values in 2D list t as a 2D table'
    for row in t:
        for item in row
            print(item, end=' ')
```

```
>>> table = [[3, 5, 7, 9],
             [0, 2, 1, 6],
             [3, 8, 3, 1]]
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def print2D(t):
    'prints values in 2D list t as a 2D table'
    for row in t:
        for item in row
            print(item, end=' ')
        print()
```

```
>>> table = [[3, 5, 7, 9],
             [0, 2, 1, 6],
             [3, 8, 3, 1]]
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def print2D(t):
    'prints values in 2D list t as a 2D table'
    for row in t:
        for item in row
            print(item, end=' ')
        print()
```

(iteration loop pattern)

```
>>> table = [[3, 5, 7, 9],
             [0, 2, 1, 6],
             [3, 8, 3, 1]]
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def incr2D(t):
    'increments each number in 2D list t'
```

```
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
>>> incr2D(t)
>>> print2D(t)
4 6 8 10
1 3 2 7
4 9 4 2
>>>
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def incr2D(t):
    'increments each number in 2D list t'

    # for every row index i
        # for every column index j
            t[i][j] += 1
```

```
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
>>> incr2D(t)
>>> print2D(t)
4 6 8 10
1 3 2 7
4 9 4 2
>>>
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def incr2D(t):
    'increments each number in 2D list t'
    # nrows = number of rows in t
    # ncols = number of columns in t

    for i in range(nrows):
        for j in range(ncols):
            t[i][j] += 1
```

```
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
>>> incr2D(t)
>>> print2D(t)
4 6 8 10
1 3 2 7
4 9 4 2
>>>
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def incr2D(t):
    'increments each number in 2D list t'
    nrows = len(t)
    ncols = len(t[0])

    for i in range(nrows):
        for j in range(ncols):
            t[i][j] += 1
```

```
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
>>> incr2D(t)
>>> print2D(t)
4 6 8 10
1 3 2 7
4 9 4 2
>>>
```

Nested loop pattern and 2D lists

2-D list-ի հետ աշխատելուց օգտագործում են ներդրված ցիկլեր մոդելը (nested loop pattern)

```
def incr2D(t):
    'increments each number in 2D list t'
    nrows = len(t)
    ncols = len(t[0])

    for i in range(nrows):
        for j in range(ncols):
            t[i][j] += 1
```

```
>>> print2D(table)
3 5 7 9
0 2 1 6
3 8 3 1
>>> incr2D(t)
>>> print2D(t)
4 6 8 10
1 3 2 7
4 9 4 2
>>>
```

(counter loop pattern)

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.