CEng 240 – Spring 2021
Week 5

Sinan Kalkan

Conditional and Repetitive Execution

Disclaimer: Figures without reference are from either from "Introduction to programming concepts with case studies in Python" or "Programming with Python for Engineers", which are both co-authored by me.

Variables, Values and Aliasing in Python

Previously on CENG240!

- Every data (whether constant or not) has an identifier (an integer) in Python:

```
>>> a = 1
>>> b = 1
>>> id(1)
135720760
>>> id(a)
135720760
>>> id(b)
135720760
```

This is called Aliasing.

- If the type of the data is mutable, there is a problem!!!

```
>>> a = ['a', 'b']
>>> b = a
>>> id(a)
3083374316L
>>> id(b)
3083374316L
>>> b[0] = 0
>>> a
[0, 'b']
```

Previously on CENG240!

```
a = 4
b = [1,2,3,a]
a = 8
print b
```

```
>>> a=[1,2]
>>> b=[1,2,a]
>>> a
[1, 2]
>>>
>>> b
[1, 2, [1, 2]]
>>> a.append(3)
>>> b
[1, 2, [1, 2, 3]]
>>> a
[1, 2, 3]
```

Actions for I/O

Previously on CENG240!

```
>>> s = input("Now enter your text: ")
Now enter your text: This is the text I entered
>>> print(s)
This is the text I entered
```

```
print(item1, item2, ..., itemN)
```

```
>>> print("I am {0} tall, {1} years old and have {2} eyes".format(1.86, 20, "brown"))
I am 1.86 tall, 20 years old and have brown eyes
```

```
>>> age = 20
>>> height = 1.70
>>> eye_color = "brown"
>>> print(f"I am {height} tall, {age} years old and have {eye_color} eyes")
I am 1.7 tall, 20 years old and have brown eyes
```

Actions for I/O

```
>>> print("I am %f tall, %d years old and have %s eyes" % (1.7569, 20, "blue"))
I am 1.756900 tall, 20 years old and have blue eyes

>>> print("I am %.2f tall, %d years old and have %s eyes" % (1.7569, 20, "blue"))
I am 1.76 tall, 20 years old and have blue eyes
```

- %f → Data identifier
- We have the following identifiers in Python:

Identifier	Description
d, i	Integer
f, F	Floating point
e, E	Floating point in exponent form
s	Using the `str()` function
r	Using the `repr()` function
%	The % character itself

Previously on CENG240!

Actions that are ignored:

Comments

```
>>> 3 + 4 # We are adding two numbers here
7
```

```
"""
This is a multi-line comment.
We are flexible with the number of lines &
  characters,
    spacing. Python
      will ignore them.
"""
```

Previously on CENG240!

Actions that are ignored:

pass statement

```
if <condition>:
        pass # @TODO fill this part
else:
        statement-1
        statement-2
        ...
```

Actions in packages

Previously on CENG240!

```
>>> pi
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pi' is not defined
>>> sin(pi)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sin' is not defined
>>> from math import *
>>> pi
3.141592653589793
>>> sin(pi)
1.2246467991473532e-16
```

Library	Description
math	Mathematical functions and definitions
cmath	Mathematical functions and definitions for complex numbers
fractions	Rational numbers and arithmetic
random	Random number generation
statistics	Statistical functions
os	Operating system functionalities
time	Time access and conversion functionalities

pip install numpy

Previously on CENG240!

Actions in packages

```
>>> import math
>>> math.sin(math.pi)
1.2246467991473532e-16
```

```
>>> import math as m
>>> m.sin(m.pi)
1.2246467991473532e-16
```

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'a
```

Previously in CENG240!

Writing your actions:

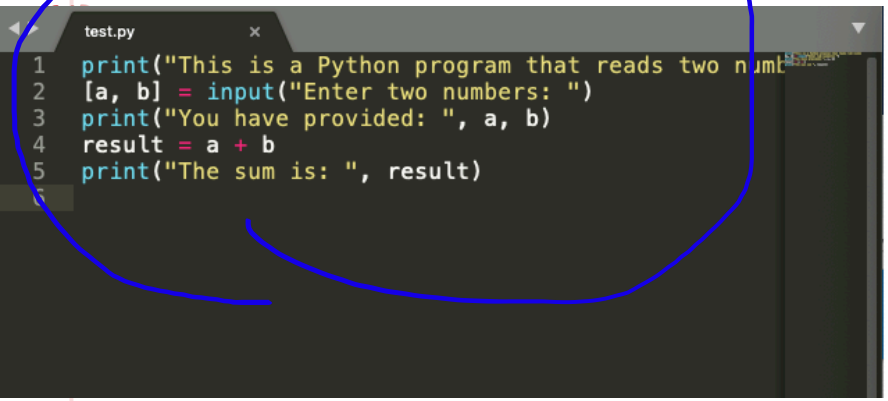
(1) Interact with the interpreter

```
$ python3
Python 3.8.5 (default, Jul 21 2020, 10:48:26)
[Clang 11.0.3 (clang-1103.0.32.62)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Python is fun")
Python is fun
>>> print("Now I am done")
Now I am done
>>> quit()
$
```

Writing your actions:

(2) Putting your actions into a script file

Previously in CENG240!




```
(base) sinankalkan@skalkan2 Downloads % ls test.py
test.py
(base) sinankalkan@skalkan2 Downloads % python3 test.py
This is a Python program that reads two numbers from the user, adds
 the numbers and prints the result


Enter two numbers:
```

Previously in CENG240!

Writing your actions:

(2) Putting your actions into a script file

```
from sys import argv

print("The arguments of this script are:\n", argv)

exec(argv[1]) # Get a
exec(argv[2]) # Get b

print("The sum of a and b is: ", a+b)
```

which can be run as follows:

```
$ python3 test.py a=10 b=20
The arguments of this script are:
 ['test.py', 'a=10', 'b=20']
The sum of a and b is:  30
```

Previously on CENG240!

Writing your actions:

(3) Your actions in a module

```
a = 10
b = 8
sum = a + b
print("a + b with a =", a, " and b =", b, " is: ", sum)
```

In another Python script or in the interpreter, you can directly type:

```
>>> from test import *
a + b with a = 10  and b = 8  is:  18
>>> a
10
>>> b
8
```

To reload:

```
>>> from importlib import reload
>>> reload(test)
```

This Week

- Conditional Execution
  - if statement
  - Conditional expression
- Repetitive Execution
  - while/for statement
  - break/continue statements
  - Set/list comprehension

Administrative Notes

- No quiz this week!
- Lab 2 and Lab 1 makeup
- Midterm: 1 June, Tuesday, 17:40

CONDITIONAL EXECUTION

Conditional Statements in Python

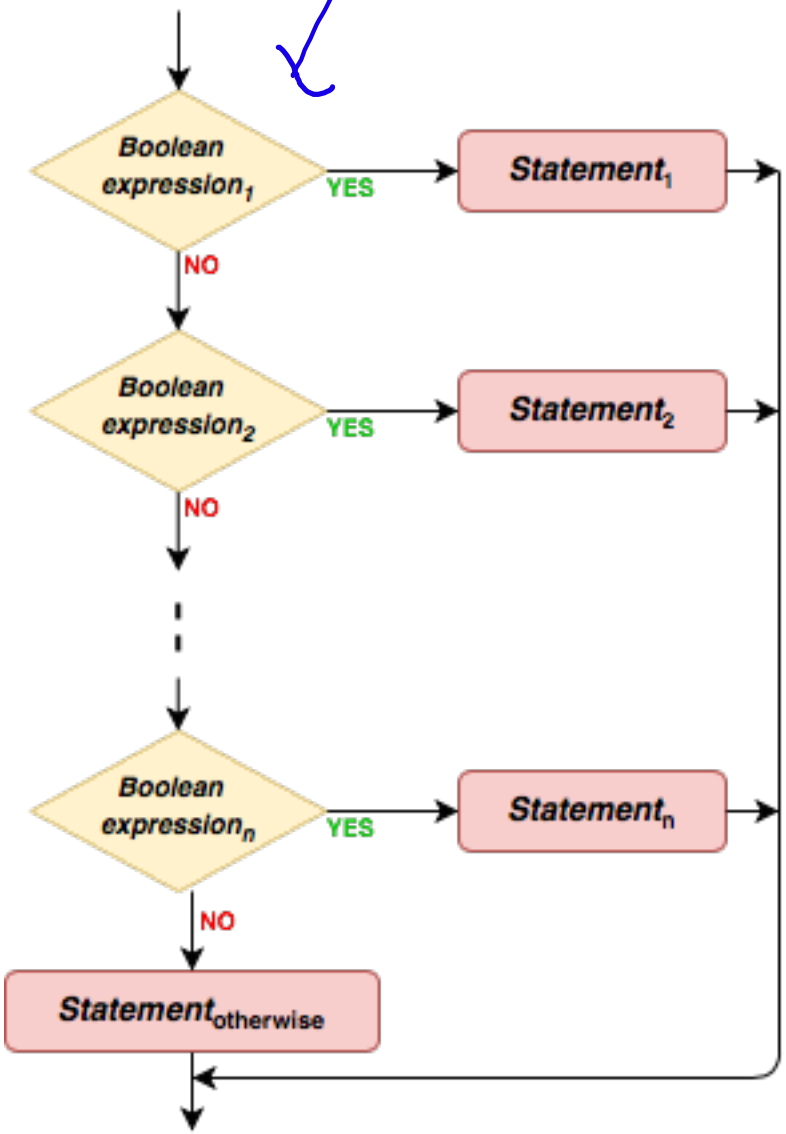
```
1 if <condition-expression> :
2     <statements-1>
3 else:
4     <statements-2>
```

- the syntax is important!
- indentation is extremely important!
- "else"-part can be omitted!

You can indent your Python code using tabs or space. However, it is a good programming practice to use only one of them while indenting your code: *i.e.*, do not mix them!

Multiple If Statements in Python

```
if Boolean expression1 : Statement1
elif Boolean expression2 : Statement2
  ⋮
elif Boolean expressionn : Statementn
else : Statementotherwise
```



Example

$$s = \begin{cases} (x+1)^2, & x < 1 \\ x - 0.5, & 1 \le x < 10 \\ \sqrt{x + 0.5}, & 10 \le x < 100 \\ 0, & otherwise \end{cases}$$

```
x = 10

if x < 1: s = (x+1)**2
elif x < 10: s = x-0.5
elif x < 100: s = (x+0.5)**0.5
else: s = 0

print("s is: ", s)
```

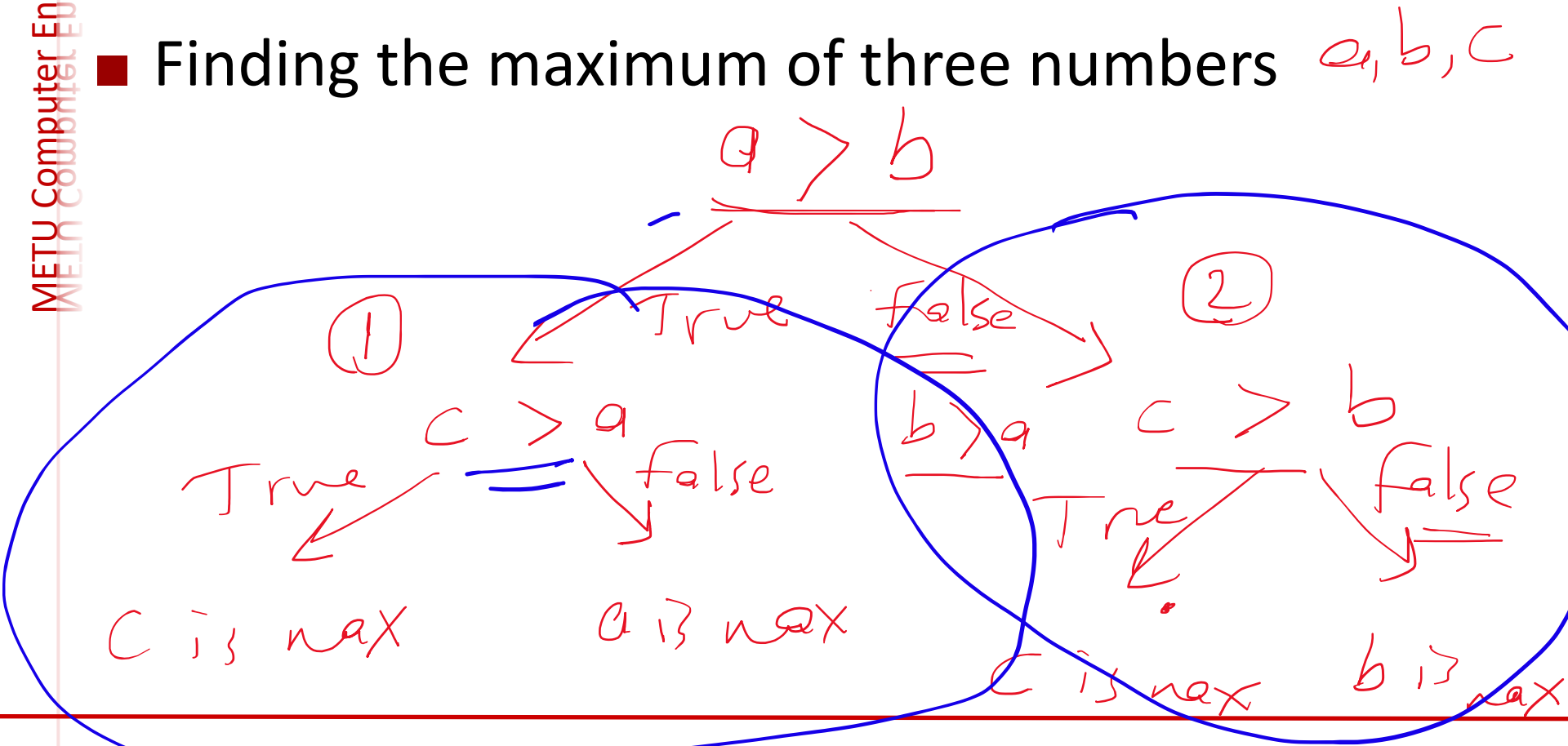
```
s is:  3.24037034920393
```

Multiple **Nested** If Statements in Python

```
1 if <condition-expression-1> :
2         <statements-1>
3         if <condition-expression-2>:
4                 <statements-2>
5         else:
6                 <statements-3>
7 else:
8         <statements-4>
```

Example

- Finding the maximum of two numbers
- Finding the maximum of three numbers



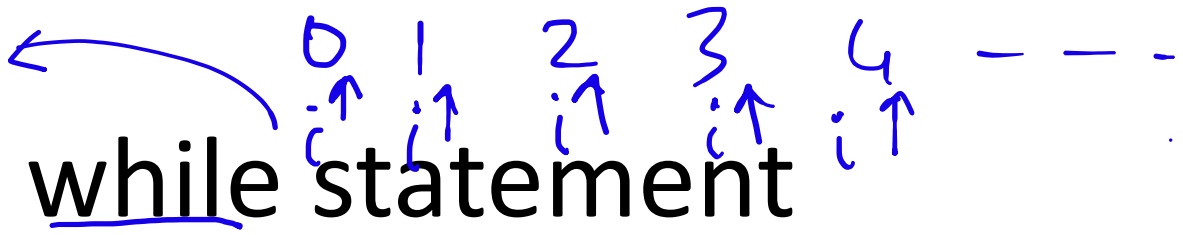
Conditional Expression in Python

<exp-1> if <cond-exp> else <exp-2>

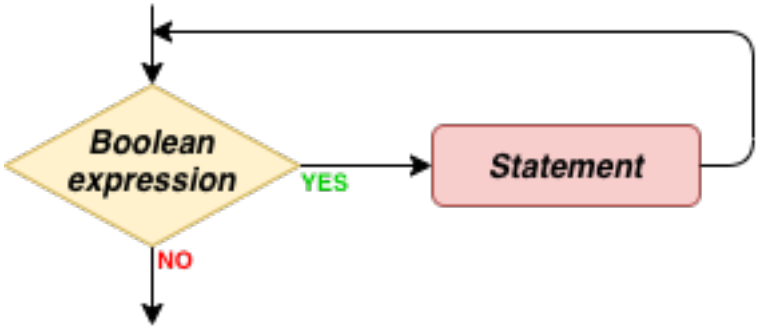
Note that this is an expression not a statement!!

REPETITIVE EXECUTION

while statement



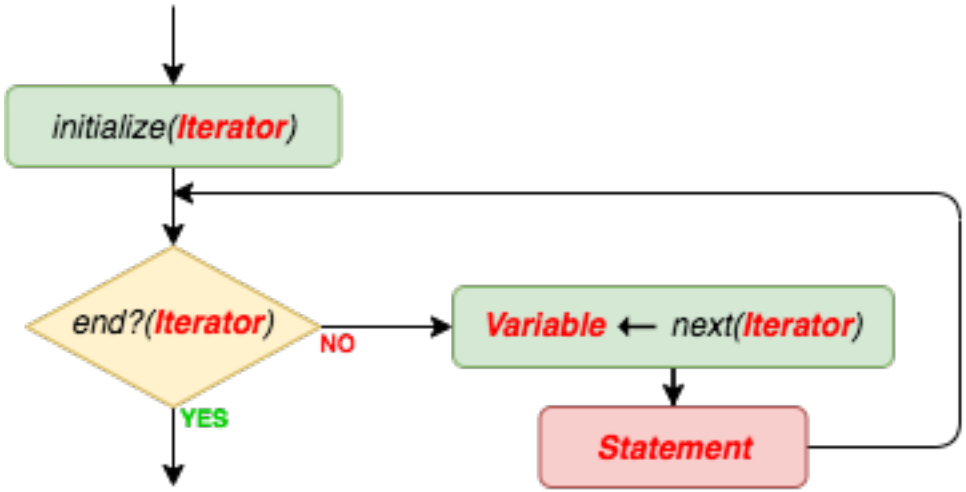

```
while Boolean expression : Statement
```




```
while <condition-1>:
    statement-1
    statement-2
    ...
    while <condition-2>:
        statement-inner-1
        statement-inner-2
        ...
        statement-inner-M
    ... # statements after the second while
    statement-N
```

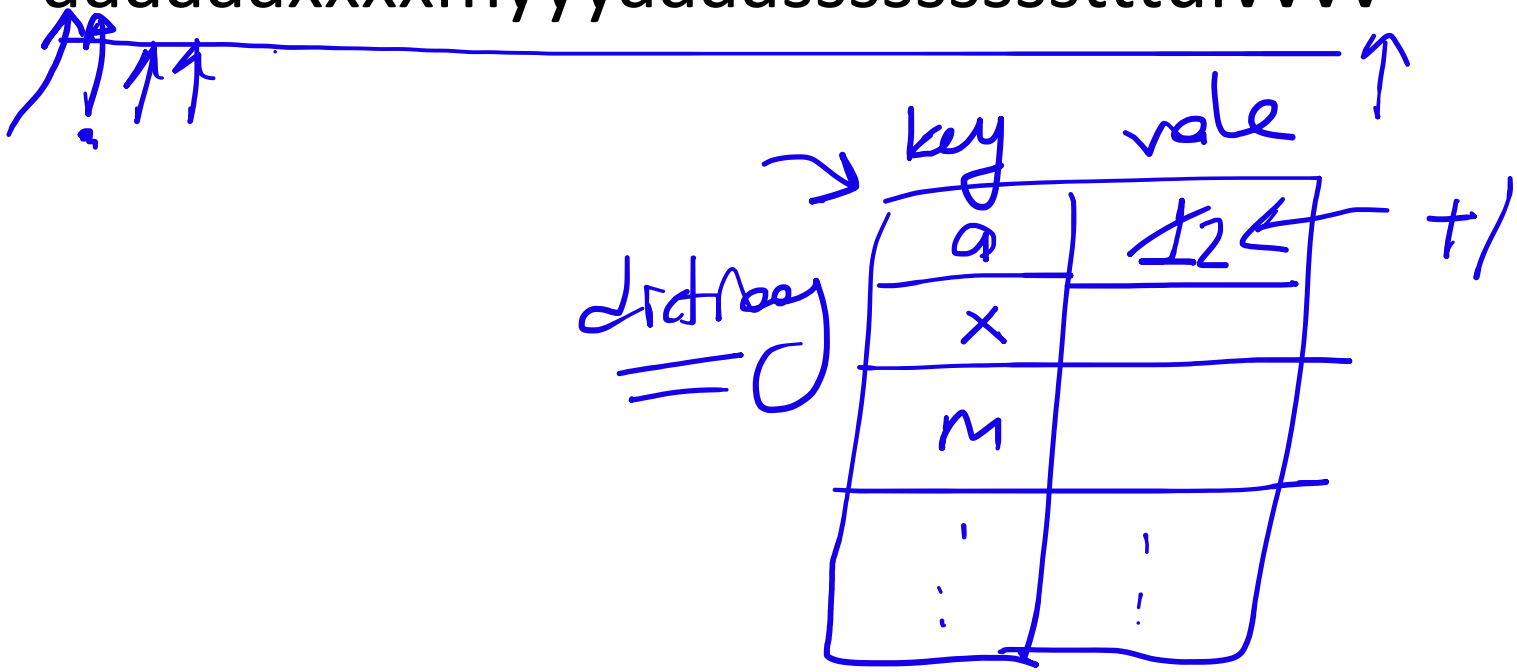
for statement

```
for Variable in Iterator : Statement
```

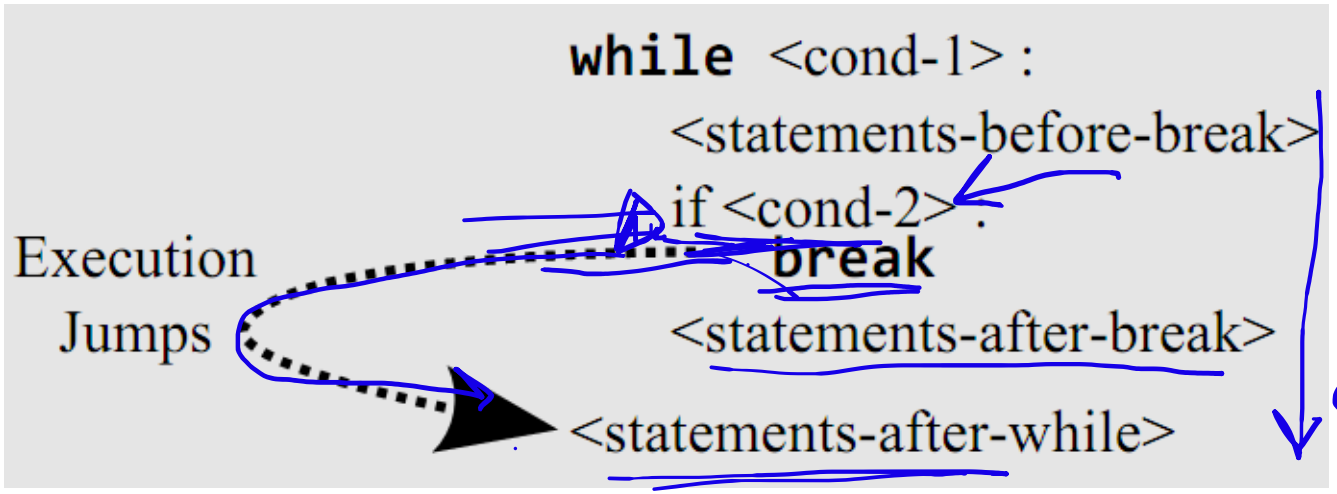


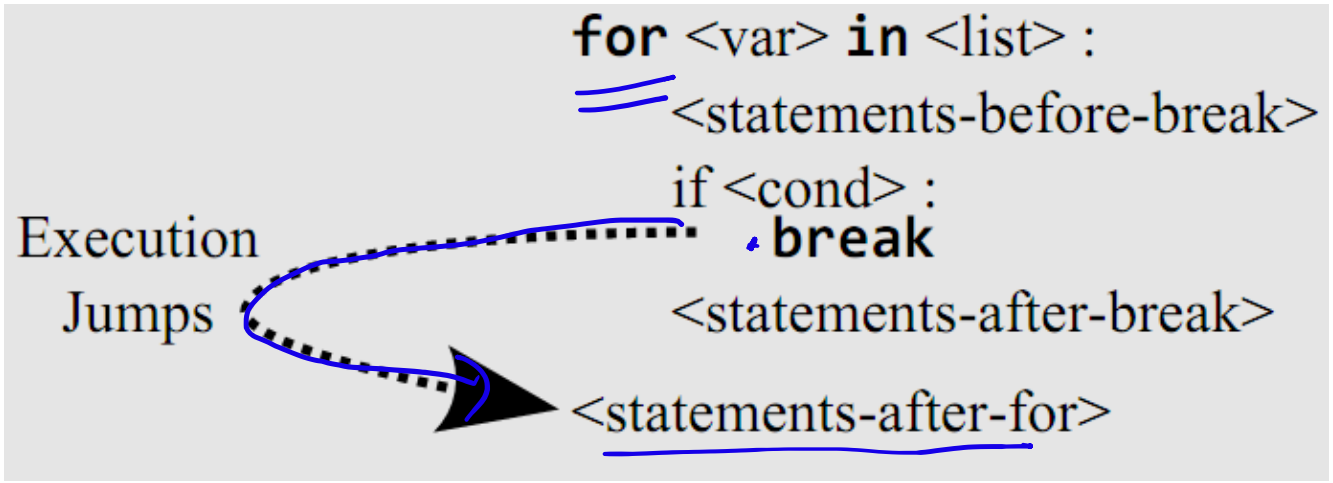
Example

- Count the occurrences of characters in a string, e.g.
  "aaaaaaxxxxmyyyaaaassssssssstttuivvvv"



Break statements

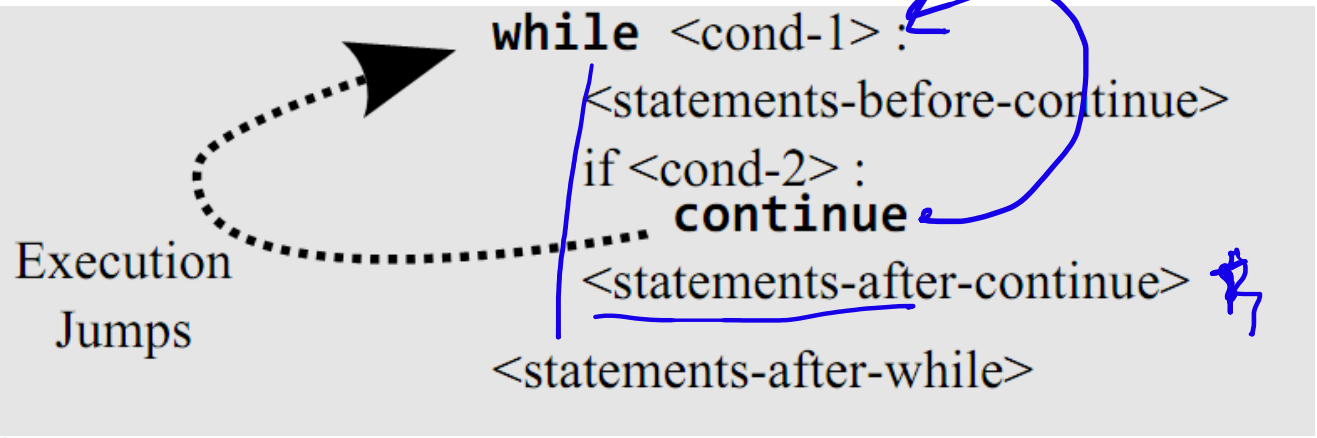


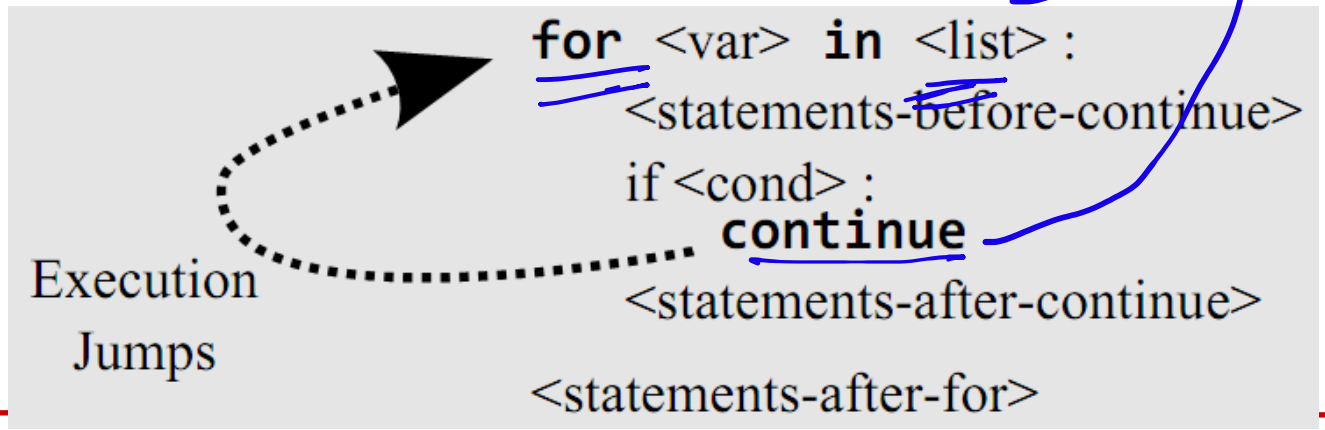


“break” example

```
1 x = 4
2 List = [1, 4, -2, 3, 8]
3 for m in List:
4         print m
5         if m == x:
6                 print "I have found a match"
7                 break
```

Continue statements

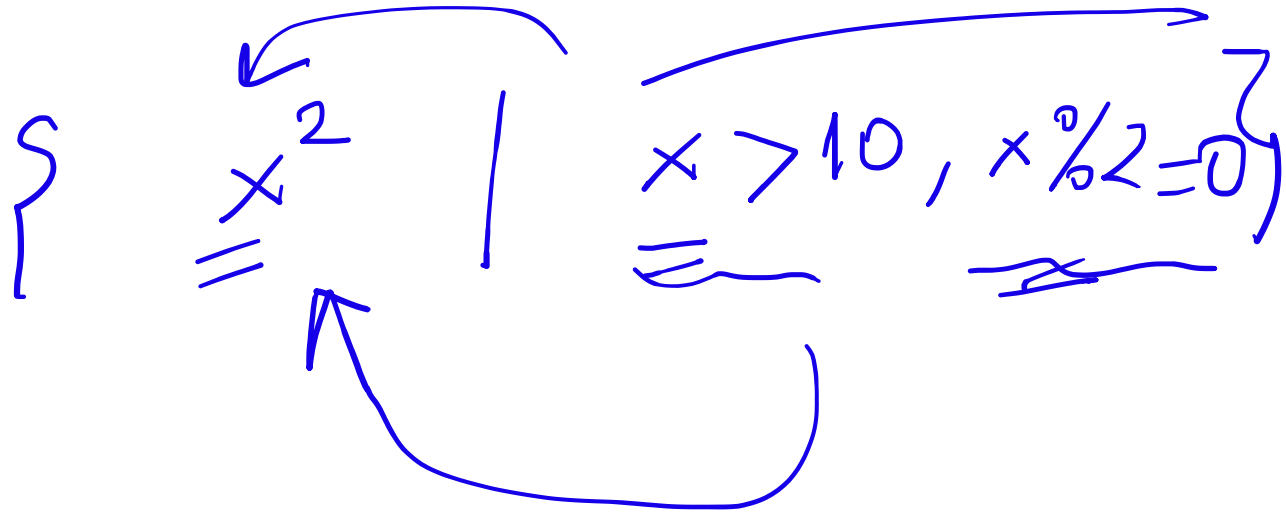




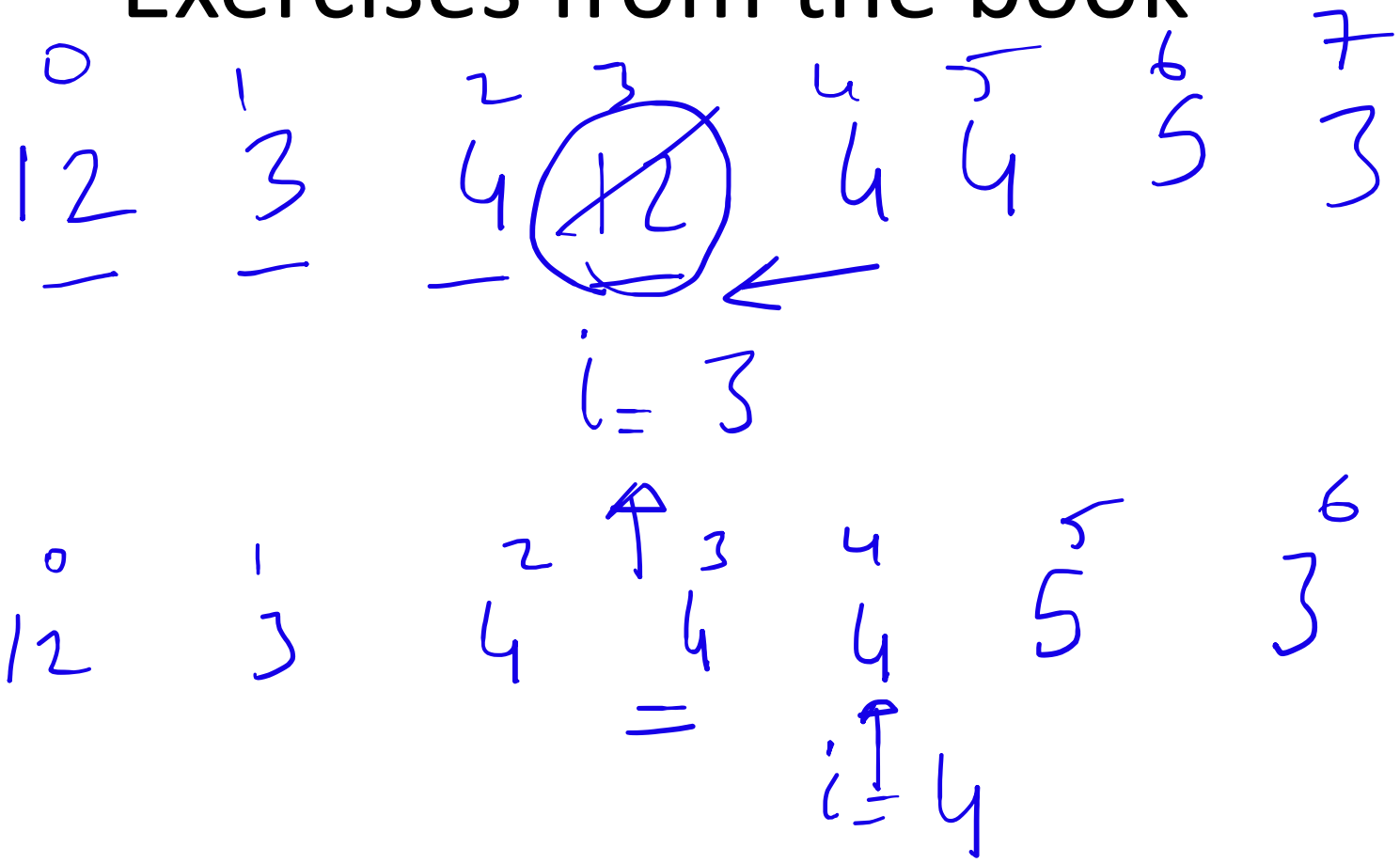
- <var> will point to the next item in the list.

Set and list comprehension

- [x**3 **for** x **in** range(8)]
- {x**3 **for** x **in** range(8)}



Exercises from the book



Final Words: Important Concepts

- Variables; Aliasing problem; Naming Variables
- Actions for interacting with the environment
- Comments, pass statements
- Actions in packages (libraries)
- Writing your actions (interpreter vs. script/modules)

THAT'S ALL FOLKS!
STAY HEALTHY