Accelerating Information Technology Innovation

http://aiti.mit.edu

Kenya Summer 2011
Lecture 3 – Control Structures, Decisions

Beyond sequential execution

- So far, all our programs have looked like this:

```
<do thing 1>
<do thing 2>
<do thing 3>
…
```

Start with first command.
Execute commands in order until
there are no more.

But often sequential execution is not enough.

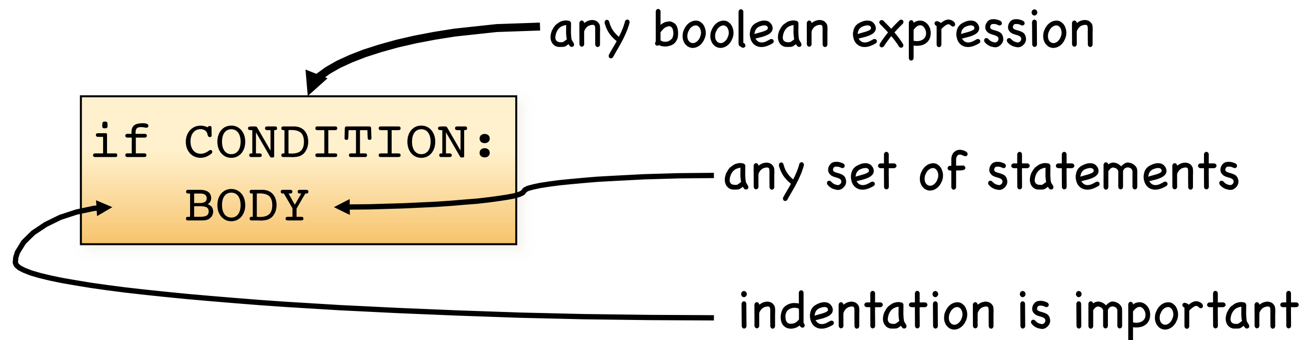
```
if <something>:
  <do thing 1>
else:
  <do thing 2>
```

If something is true, execute
the first command. Otherwise,
execute the second command.

Control statements

- Conditionals: control which set of statements is executed.
  - if / else
- Iteration: control how many times a set of statements is executed.
  - while loops
  - for loops

The if statement



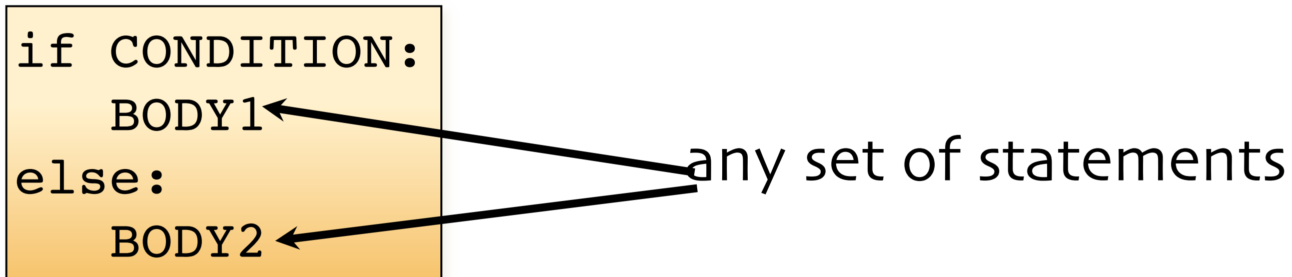
- If the condition is True, the body gets executed.
- Otherwise, nothing happens.

```
if x < 0:
    print 'x is negative'
```

- NOTE: IDLE editor helps with indentation.

4

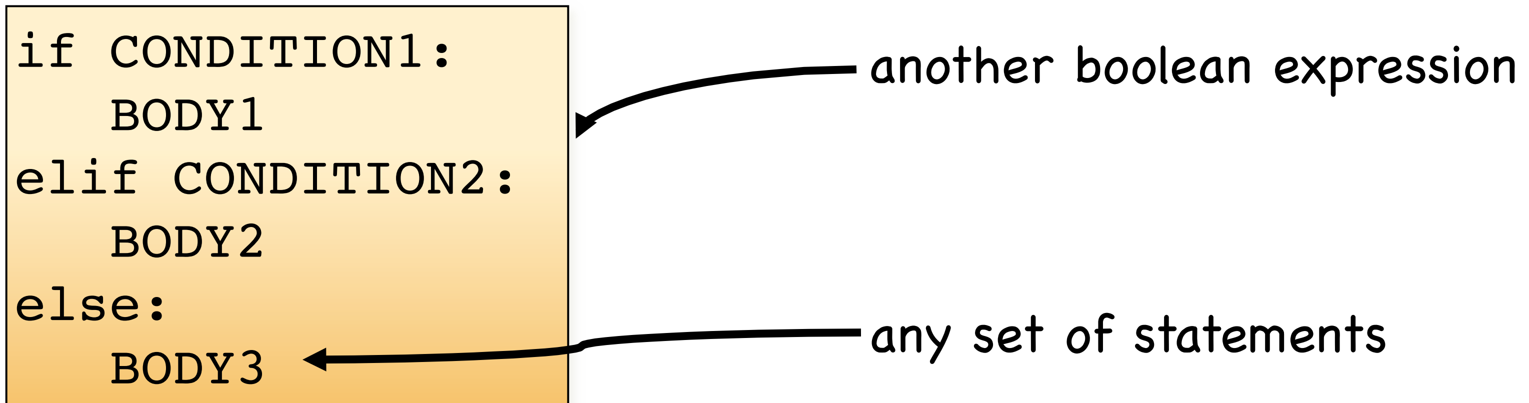
The if/else statement



- If the condition is True, body1 gets executed.
- Otherwise, body2 gets executed.

```
if x < 0:
    print 'x is negative'
else:
    print 'x is positive or zero'
```

Chained conditionals



- If the condition1 is True, body1 gets executed.
- Otherwise, if condition2 is True, body2 gets executed.
- If neither condition is True, body3 gets executed.

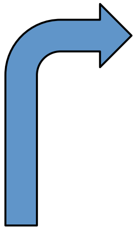
An example

```
a = False
b = True
if a and b:
   print 'I love red.'
elif a or b:
   print 'I love green.'
else:
   print 'I love blue.'
   print 'I also love purple.'
```

What does this output? `I love green.`

An example

```
a = False
b = True
if a and b:
   print 'I love red.'
elif a or b:
   print 'I love green.'
else:
   print 'I love blue.'
print 'I also love purple.'
```



What does this output?

```
I love green.
I also love purple.
```

Nested conditionals

```
if is_adult:
    if is_senior_citizen:
        print 'Admission $2 off.'
    else:
        print 'Full price.'
else:
    print 'Admission $5 off.'
```

outer conditional

inner conditional

- Can get confusing. Indentation helps to keep the code readable and the python interpreter happy!

Another example

```
x = 4
y = -3
if x < 0:
    if y > 0:
    print x + y
    else:
        print x – y
else:
    print x * y
```

What does this output? -12

Common if errors

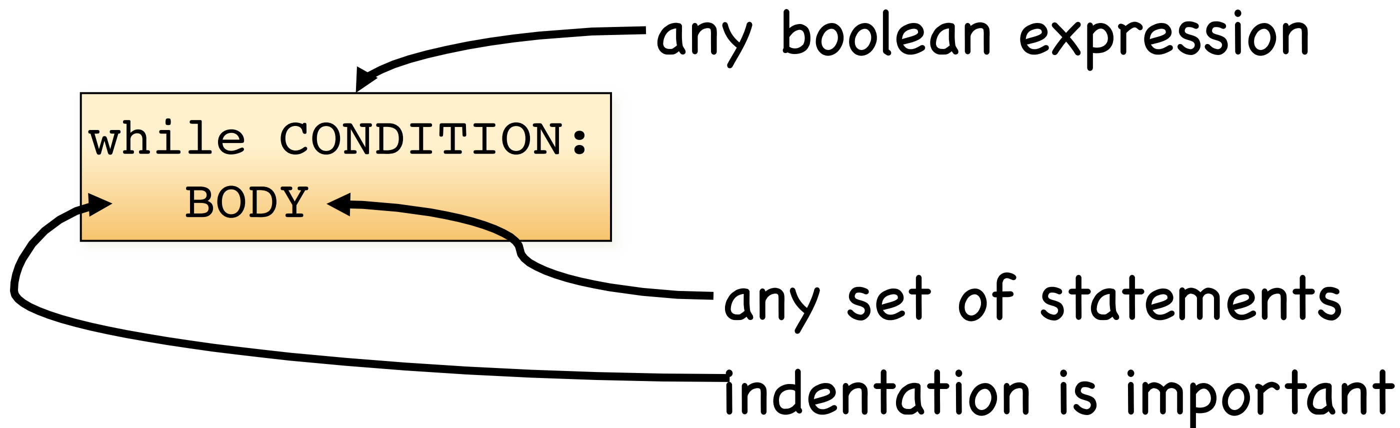
- Syntax errors
  - Mixing up = and == in the condition

```
b = False
if b = False
    print b
  print 'inside if maybe'
```

```
SyntaxError: invalid syntax
```

```
IndentationError: unindent does not match
any outer indentation level
```

The while loop



- As long as the condition is true, the body gets executed repeatedly.
- The first time the condition is false, execution ends.

12

The while loop

```
i = 0
while i < 3:
    print i
    i = i + 1
```

- What does this output?

0

1

2

Side note: if the condition is false the first time it is tested, the body is never executed

The break statement

- Immediately exits the innermost loop.

```
while True:
   line = raw_input('>>> ')
   if line == 'done':
       break
   print line
print 'Done!'
```

```
>>> not done
not done
>>> done
Done!
```

(An if statement is not a loop!)

What' will happen with this code?

```
i = 0
while i < 3:
   print i
```

- It will loop forever (aka Infinite loop)! How do we fix it?

```
i = 0
while i < 3:
   print i
   i = i + 1
```


15

The infinite loop

```
i = 4
while i > 0:
    print i
    i = i + 1
```

- This code also loops forever!
- Why? And how do you fix this?

```
i = 4
while i > 0:
    print i
    i = i - 1
```

Lists

- A list is a sequence of values.
- Each element (value) is identified by an index.
- The elements of the list can be of any type.

Lists

- A list is a sequence of values.
- Each element (value) is identified by an index.
- The elements of the list can be of any type.

```
tens = [10, 20, 30, 40]
coins = ['dime', 'nickel', 'quarter', 'penny']
empty = []
```

Lists

- A list is a sequence of values.
- Each element (value) is identified by an index.
- The elements of the list can be of any type.

```
tens = [10, 20, 30, 40]
cities= [’Nairobi', ‘Mombasa', ‘Kisumu', ’Nakuru']
empty = []
```

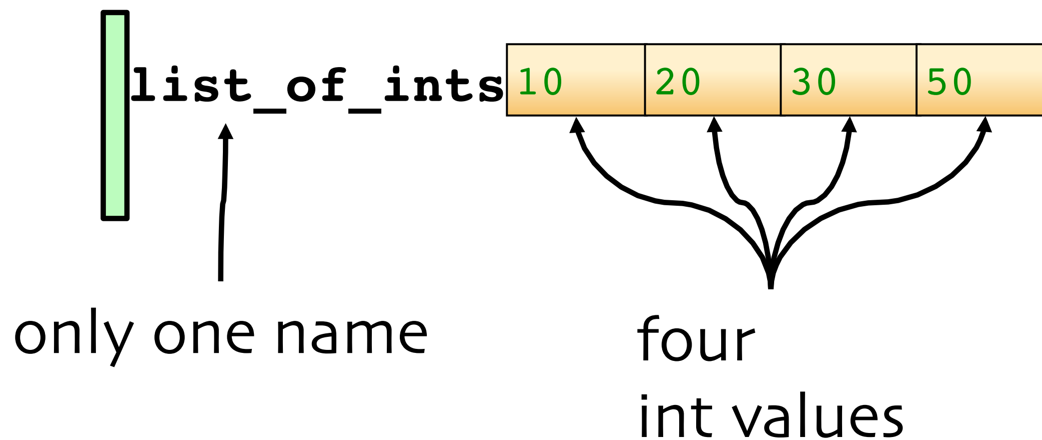
- Lists can have mixed types in them, even other lists (nested).

```
mixed = ['hello', 2.0, 5, [10, 20]]
```

Creating a list

- Use the [] brackets

```
list_of_ints = [10,20,30,50]
```



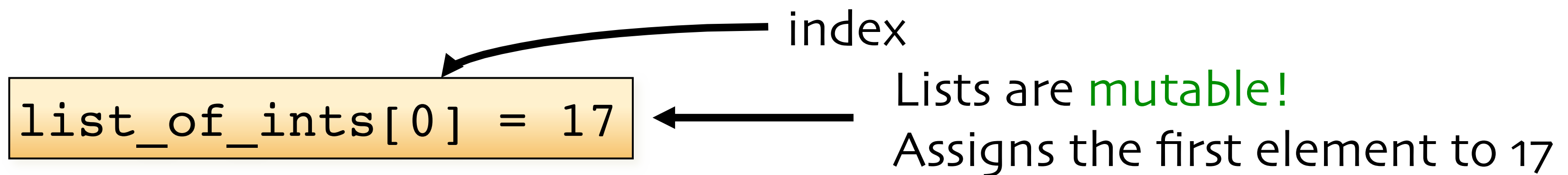
List operators

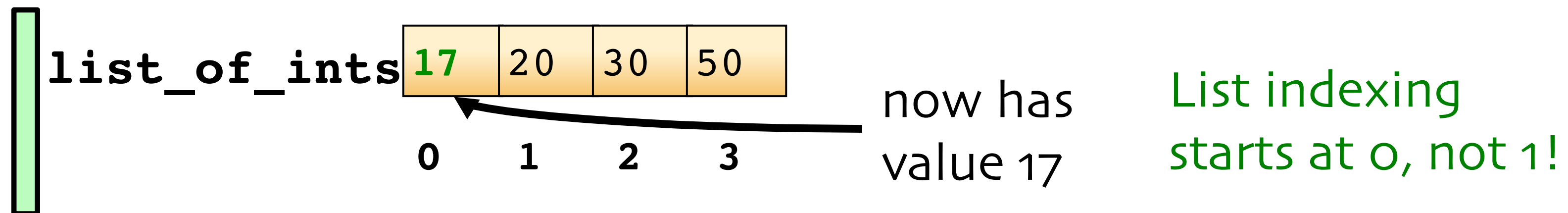
- Applied to lists, produce lists

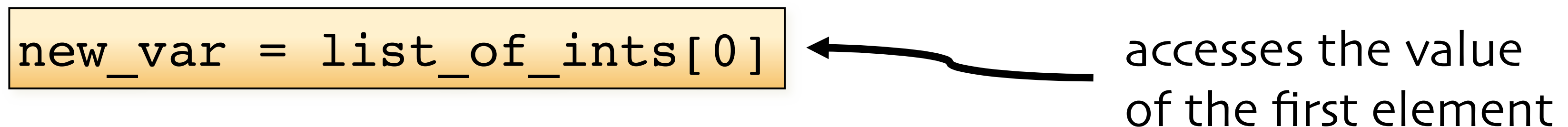
Operator	Operation
+	Concatenation
*	Repetition
<list>[ ]	Indexing
<list>[:]	Slicing

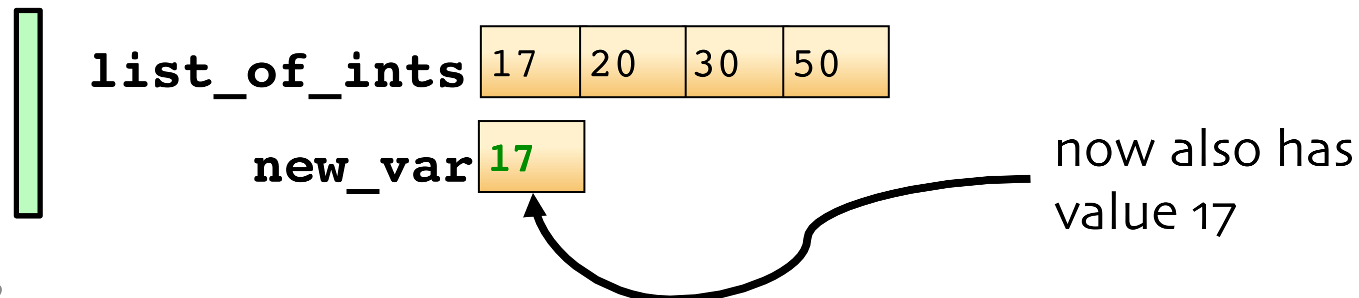
Accessing list elements

- Individual elements are accessed using the [] operator.









Printing a list

- We can use the print function to output the contents of the list:

```
cities = [’Nairobi', ’Mombasa', ’Kisumu']
numbers = [17, 123]
empty = []
print cities, numbers, empty
```

```
[’Nairobi', ’Mombasa', ’Kisumu'] [17, 123] [ ]
```

An example

```
1 numbers = [10, 20, 30]
2 letters = ['a', 'b', 'c']
3 number = numbers[0]
4 letter = letters[2]
5 print 'number =', number
6 print 'letter:', letter
7 print 'letters:', letters
```

```
number = 10
letter: 'c'
letters: ['a', 'b', 'c']
```

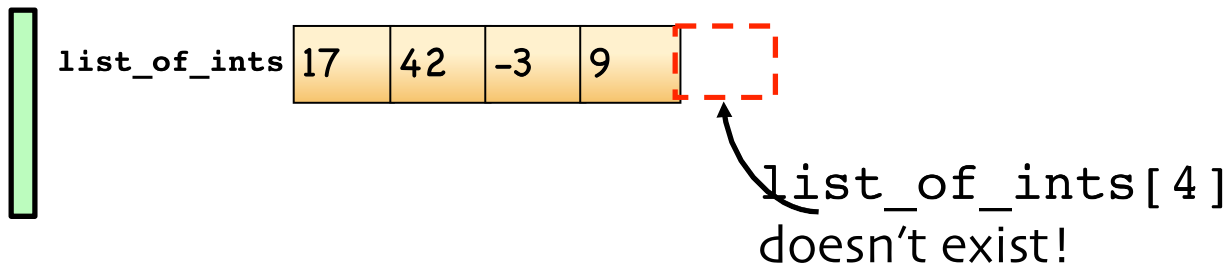
Another example

```
1  numbers = [10, 20, 30]
2  letters = ['a', 'b', 'c']
3  mixed =  letters + numbers
4  print mixed                 ['a','b','c',10,20,30]
5  print letters*2             ['a','b','c','a','b','c']
6  numbers[2] = letters
7  print numbers               [10, 20, ['a','b','c']]
8  print numbers[:2]           [10, 20]
9  numbers[1:] = [40, 50]
10 print numbers               [10, 40, 50]
```

Out-of-range errors

- You will get a runtime error if you try to access an element that does not exist!

```
list_of_ints = [17, 9, 42, -2]
print list_of_ints[4]
```




```
IndexError: list index out of range
```

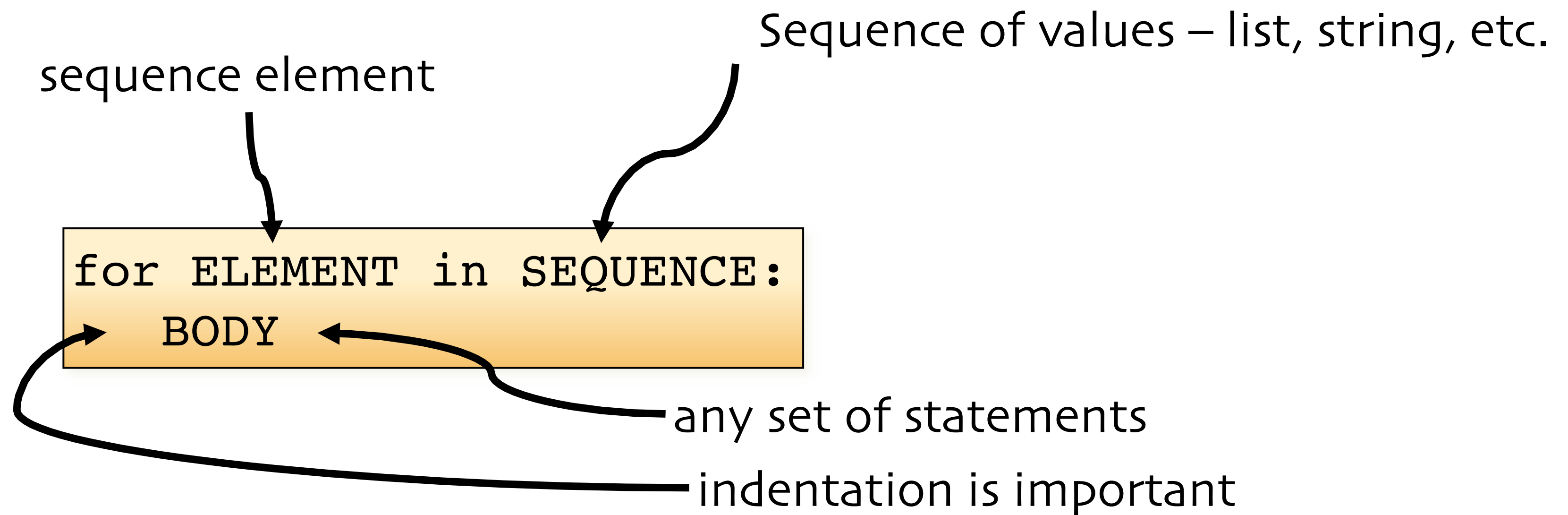
Lists vs. Strings

- Lists are mutable - their contents can be modified
- Strings are immutable

```
name = 'Lenny'
name[0] = 'J'
```

TypeError: object doesn't support item assignment

The for loop

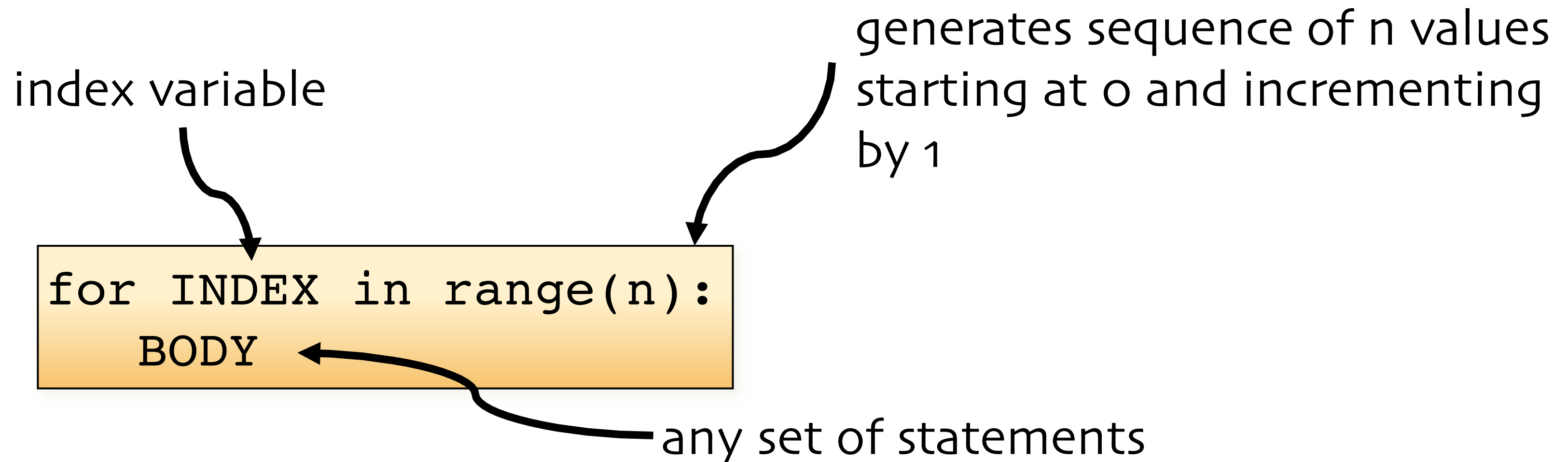


- Example:

```
for i in [0,1,2,3]:
    print i
```

```
0
1
2
3
```

Using range

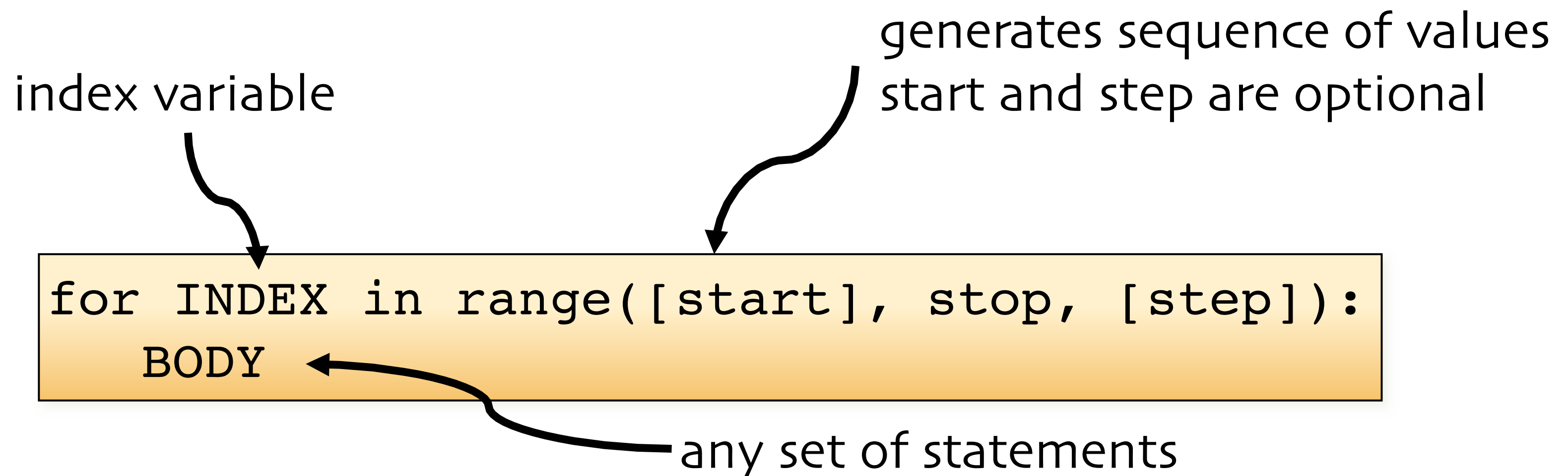


- What does this output?

```
for i in range(4):
    sq = i * i
    print i, sq
```

```
0  0
1  1
2  4
3  9
```

Using range



- What does this output?

```
for i in range(1, 7, 2):
    print i
```

1

3

5

For loop and strings

- Iterating through the characters of a string

```
str1 = 'stressed'
for c in str1:
    print c,
```

```
s t r e s s e d
```


31

For loop and strings

- What is the output?

```
str1 = 'stressed'
res = ''
for c in str1:
    res = c + res
print res
```

Iteration #	c	res
0	s	s
1	t	ts
2	r	rts
3	e	erts
4	s	serts
5	s	sserts
6	e	esserts
7	d	desserts

```
desserts
```

For loop and lists

- Iterating through the elements of a list

```
desserts = [['ice cream', 3.5], ['chocolate cake', 2], ['pudding', 3]]

for dessert_list_item in desserts:
     print dessert_list_item[0], 'costs $', dessert_list_item[1]
```

```
ice cream costs $ 3.5
chocolate cake costs $ 2
pudding costs $ 3
```

Exercise

- Is there another way we can get the same output from the ‘desserts’ list?

```
desserts = [['ice cream', 3.5], ['chocolate cake', 2], ['pudding', 3]]

for dessert_list_item in desserts:
    print dessert_list_item[0], 'costs $', dessert_list_item[1]
```

```
ice cream costs $ 3.5
chocolate cake costs $ 2
pudding costs $ 3
```

Combining for and if

```
for i in range(6):
   if i % 2 == 0:
      print i, 'is even.'
   else:
      print i, 'is odd.'
```

- What does this output?

```
0 is even.
1 is odd.
2 is even.
3 is odd.
4 is even.
5 is odd.
```

Nested for loops

```
for i in range(1,6):
   for j in range(1, 6):
   prod = i * j
   # use comma to print all on one line
      print prod,
   print
```

must use new index variable for inner loop

- What does this output?

```
1   2   3   4   5
2   4   6   8   10
3   6   9   12  15
4   8   12   16   20
5   10   15   20   25
```

For vs While

- For loop is primarily used
    - for iterating over a sequence of values
    - when we know the number of iterations in advance
- While loop is primarily used
    - when we don't know the number of iterations in advance (they could be controlled by user input)

37

Questions?

Control Flow

Do it Once with If…Elif…Else

```
def printFirstLetter(words):
    if words[0].isalpha():
        print words
    elif words[0].isspace():
        print ″space"
    else:
        pass
```

```
>>> printFirstLetter
("Hello")
Hello

>>> printFirstLetter(" ")
space

>>> printFirstLetter("555″)
```

Loops: Break vs Continue

```
def parsewords_while(words):
    currentword = ""
    index = 0
    while index < len(words):
        if words[index].isalpha():
            currentword += words[index]
        elif words[index].isspace():
            print currentword
            currentword = ""
        index += 1
    print currentword

>>> parsewords_while("How are y555ou")
How are you
```

Loops

While

```
def parsewords_while(words):
    currentword = ""
    index = 0
    while index < len(words):
        if words[index].isalpha():
            currentword += words[index]
        elif words[index].isspace():
            print currentword
            currentword = ""
            index += 1
     print currentword

>>> parsewords_while("How are y555ou")
How are you
```

Loops

For

```
def parsewords_for(words):
    currentword = ””
    for index in range(len(words)):
        if words[index].isalpha():
            currentword+=words[index]
        elif words[index].isspace():
            print currentword,
            currentword = ””
    print currentword
```

```
>>> parsewords_for(”How
are y555ou")
How are you


>>> range(5)
[0, 1, 2, 3, 4]
>>> range(2,6)
[2, 3, 4, 5]
>>> range(-10,5,2)
[-10, -8, -6, -4, -2, 0,
2, 4]
```

Loops

For(Original)

```
def parsewords_for(words):
    currentword = ""

    for index in range(len(words)):
        if words[index].isalpha():
            currentword += words[index]
        elif words[index].isspace():
            print currentword,
            currentword = ""

    else:
        print currentword

>>> parsewords_for("How are y555ou")
How are you
```

For (Alternate)

```
def parsewords_for(words):
    currentword = ""

    for char in words:
        if char.isalpha():
            currentword += char
        elif char.isspace():
            print currentword,
            currentword = ""

    else:
        print currentword

>>> parsewords_for("How are y555ou")
How are you
```

Nested Loops

```
def parsewords_for(words):
    currentword = ""
    for char in words:
        if char.isalpha():
            currentword += char
        elif char.isspace():
            for letter in currentword:
                print letter,
            currentword = ""
    print currentword

>>> parsewords_for("How are y555ou")
H o w a r e y o u
```

Why we need control structures

- Decide what to do next
- Do certain actions for certain events
- Repeat a series of actions
- Break a series of actions