2019

Python loops

Natalia Novak
Bronx Community College, City University of New York

Loops

- While Loops
- For Loops
- Nested loops
- Break and continue

Loops

What is a loop?

Loops are all about repeating the same behavior for some number of times or until some condition is met.

Example: recall our Candy Quest

```
repeat until “red candy”
   if “bottle cap”:
      jump
   else:
      walk
```

While loops

Two types of loop in Python

In Python we have two types of loops:

while loop: executes a block of code as long as the loop's expression is True.

```
while <condition>:
   # loop body
   # loop body

# statements to execute when condition
# becomes False
```

While loops

Two types of loop in Python

In Python we have two types of loops:

for loop: a counted loop, executes a block of code n times

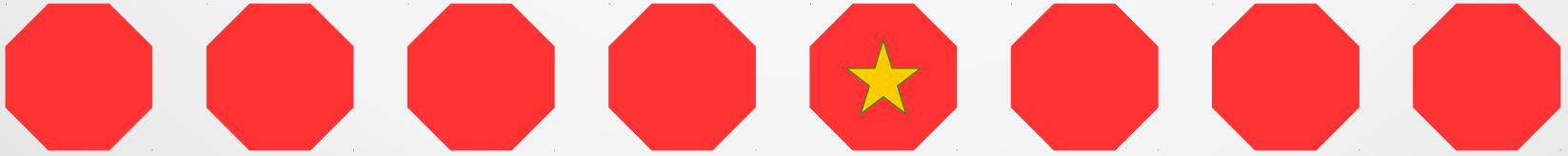
```
for variable in container:
    # Loop body
    # Loop body

# Statements to execute after the for loop
# is complete
```

While loops

While loops

Let’s start with while loops :

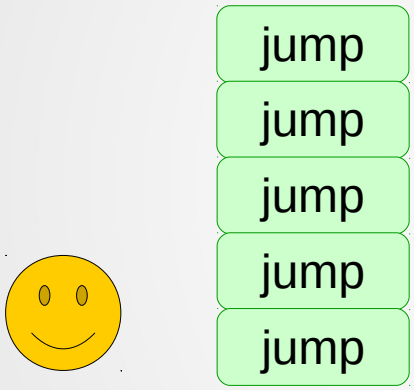
jump

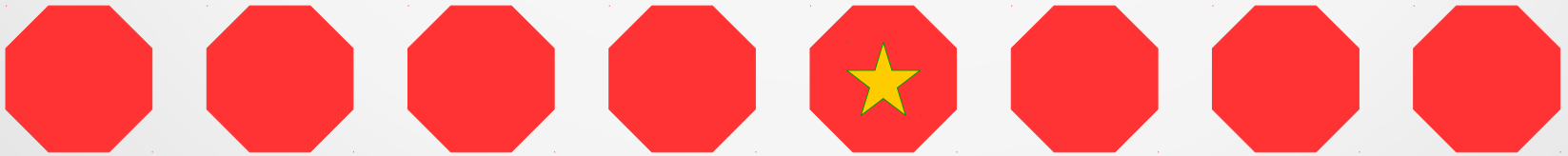


While loops

While loops

Let’s start with while loops :



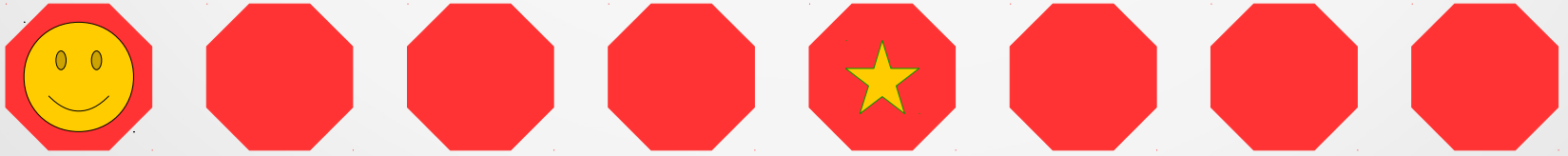


While loops

While loops

Let’s start with while loops :

jump
jump
jump
jump
jump

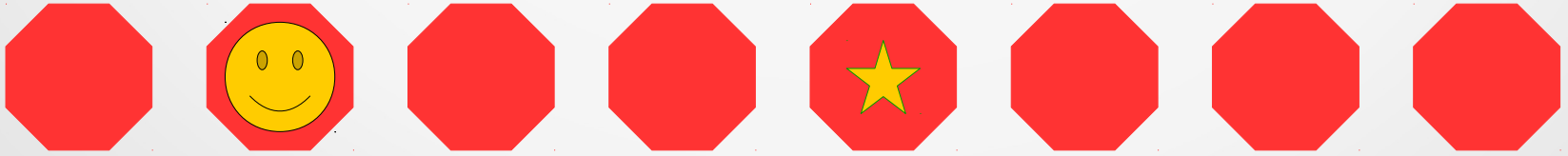


While loops

While loops

Let’s start with while loops :

jump
jump
jump
jump
jump

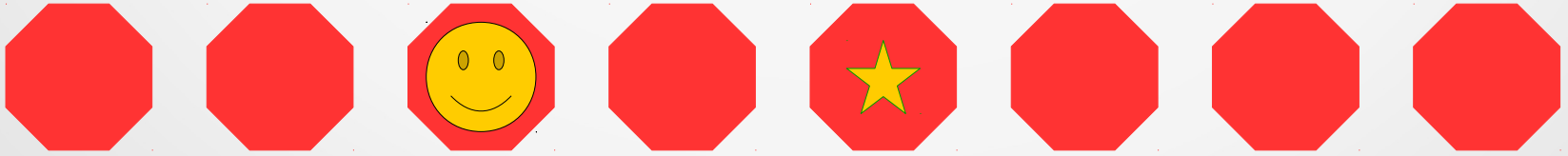


While loops

While loops

Let’s start with while loops :

jump
jump
jump
jump
jump

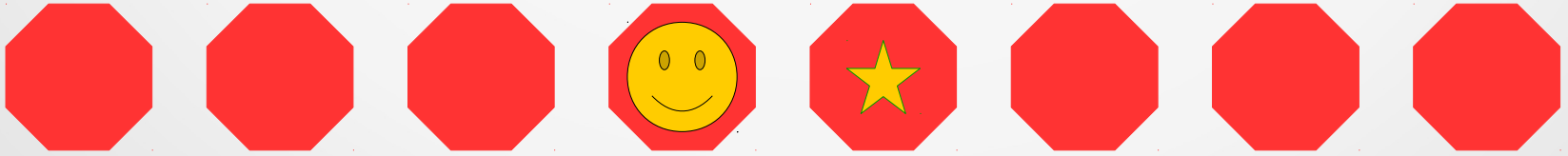


While loops

While loops

Let’s start with while loops :

```
jump
jump
jump
jump
jump
```

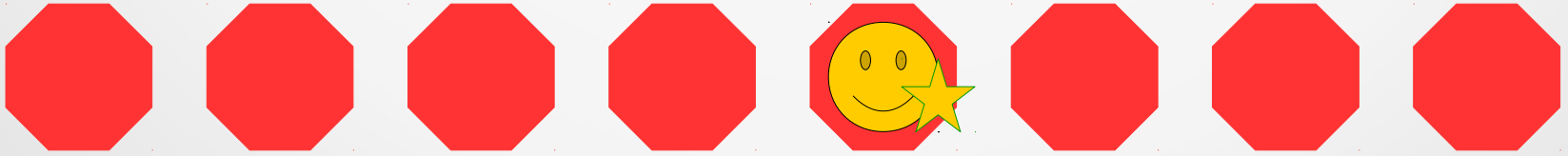


While loops

While loops

Let’s start with while loops :

jump
jump
jump
jump
jump

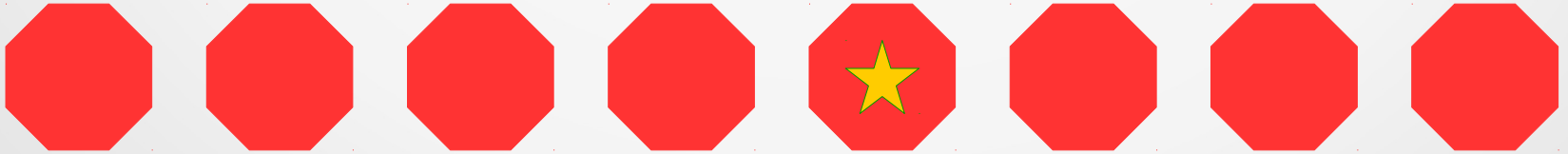


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

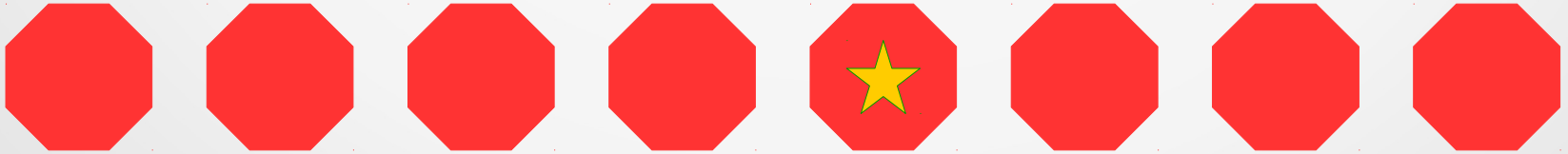


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

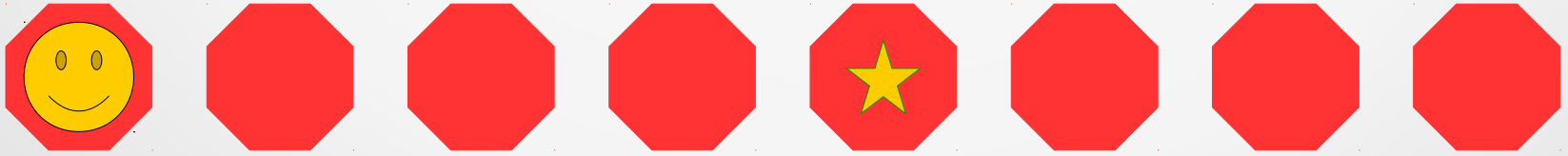


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

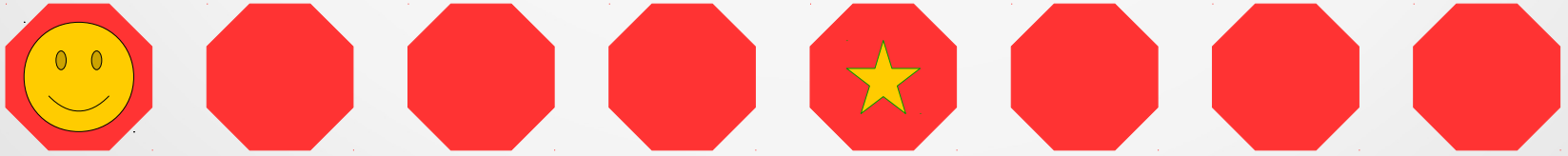


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

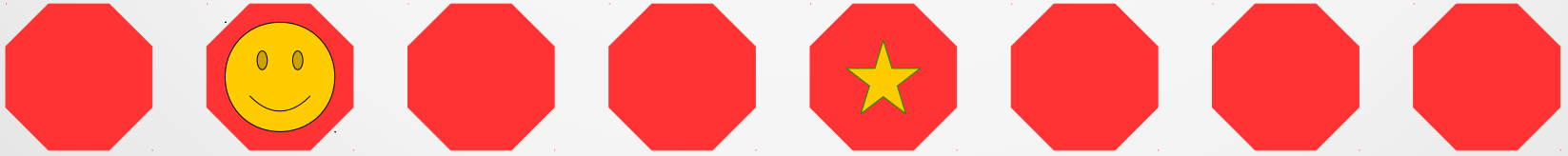


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

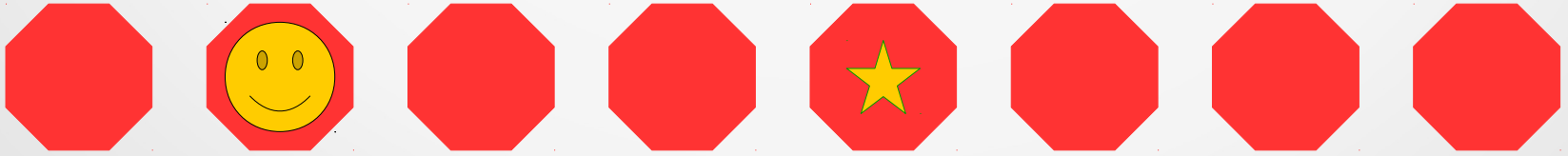


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

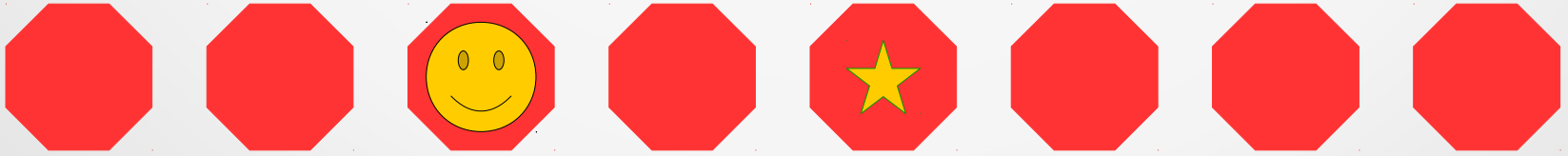


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

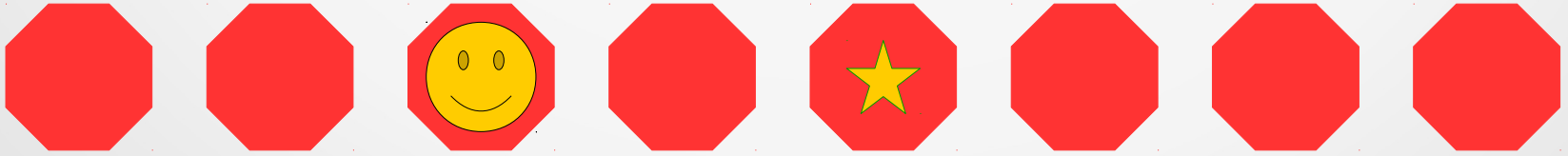


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

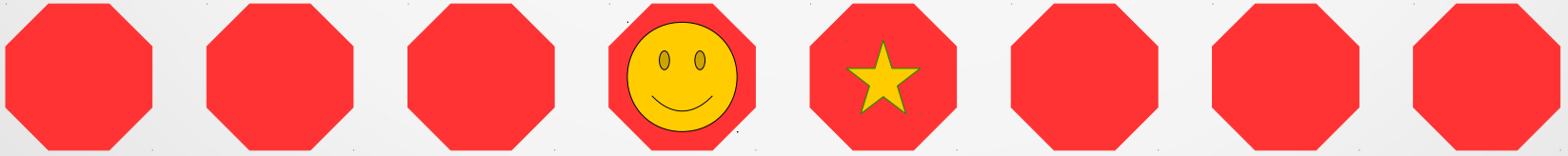


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

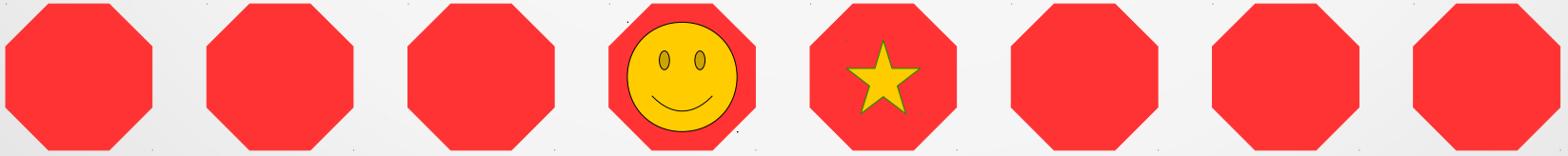


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

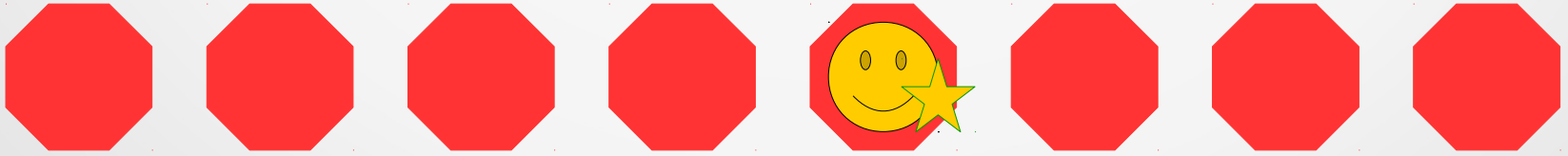


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

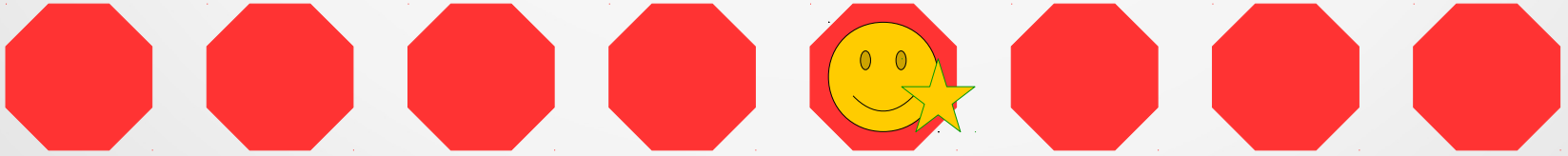


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```

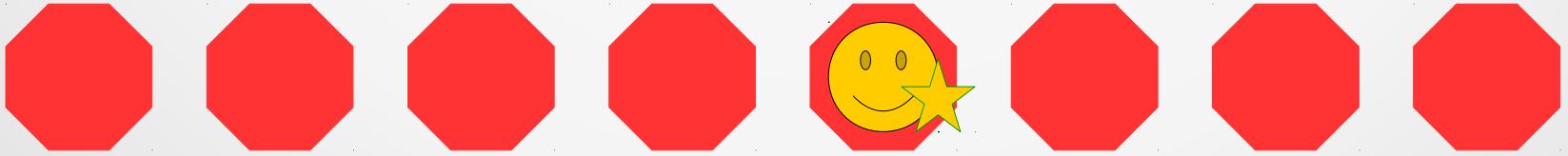


While loops

While loops

Let’s start with while loops :

```
while the star is not reached:
    jump
```



While loops

While loops

Let's talk about while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:
```

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
```

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
```

x=7

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
```

x=7

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
```

x=4

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
```

x=4

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
***
```

x=4

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
***
```

x=1

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
***
```

x=1

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
***
***
```

x=1

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
***
***
```

x=-2

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
***
***
```

x=-2

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

```
Enter a positive integer:7
***
***
***
the end
```

x=-2

While loops

While loops

Let's start with while loops.

```
x = int(input("Enter a positive integer:"))

while x > 0:
    print("***")
    x = x - 3
print("the end")
```

Each execution of the loop body is called an *iteration*, and looping is also called *iterating*

While loops

While loops: common errors

An *infinite loop* is a loop that will always execute because the loop's expression is always True.

A common error is to accidentally create an infinite loop by assuming equality will be reached.

While loops

While loops: common errors

An *infinite loop* is a loop that will always execute because the loop's expression is always True.

A common error is to accidentally create an infinite loop by assuming equality will be reached.

```
x = int(input("Enter a positive integer:"))

while x = 0:
   print("***")
   x = x - 3
print("the end")
```

While loops

While loops: common errors

An *infinite loop* is a loop that will always execute because the loop's expression is always True.

A common error is to accidentally create an infinite loop by assuming equality will be reached.

```
x = int(input("Enter a positive integer:"))

while x = 0:
    print("***")
    x = x - 3
print("the end")
```

x = 7, 4, -1, -4, ...

While Loops

While loops

Let's take a look at the programs with various loops:

`whileLoopExamples1.py`

`whileLoopExamples2.py`

`whileLoopExamples3.py`

`whileLoopExamples4.py`

While Loops

In-class work

See the handout, items 1-3

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for variable in container:
    # Loop body
    # Loop body

# Statements to execute after the for loop
# is complete
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20, 40,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20, 40,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20, 40, 70,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20, 40, 70,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    →print(2*item, end=", ")
print("finished!")
```

20, 40, 70, 86,

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

20, 40, 70, 86,

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    →print(2*item, end=", ")
print("finished!")
```

20, 40, 70, 86, 112,

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20, 40, 70, 86, 112,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20, 40, 70, 86, 112, 180,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:
    print(2*item, end=", ")
print("finished!")
```

```
20, 40, 70, 86, 112, 180,
```

For loops

Another type of loop in Python

for loop: a counted loop, executes a block of code n times;
Iterates over the elements in a container

```
for item in [10,20,35,43,56,90]:

    print(2*item, end=", ")

print("finished!")
```

20, 40, 70, 86, 112, 180,
finished!

For loops

Another type of loop in Python

Example:

```
myD = { 172: "Friday", 823: "Tuesday", 564: "Monday", 923: "Saturday", 435: "Wednesday", 712: "Sunday", 384: "Thursday"}

for k in myD:
    print("%d corresponds to %s" % (k,myD[k]))
print("finished!")
```

For loops

Another type of loop in Python

Example:

```
myD = { 172: "Friday", 823: "Tuesday", 564: "Monday", 923: "Saturday", 435: "Wednesday", 712: "Sunday", 384: "Thursday"}

for k in myD:
    print("%d corresponds to %s" % (k,myD[k]))
print("finished!")
```

```
172 corresponds to Friday
823 corresponds to Tuesday
564 corresponds to Monday
923 corresponds to Saturday
...
```

`For` loops

Example: Assume that we are given a list of decimal values in a Python list, named `data`. And we are asked to write a program that finds the average of all the values in the list `data`.

For loops

Example: Assume that we are given a list of decimal values in a Python list, named `data`. And we are asked to write a program that finds the average of all the values in the list `data`.

Here is an order of operations to do:

- add the values from list `data` one by one to a variable `s`,

- `count` the number of values in the list `data`,

- get the average by dividing `s` by `count`.

For loops

Example: Assume that we are given a list of decimal values in a Python list, named `data`. And we are asked to write a program that finds the average of all the values in the list `data`.

Here is a program for that:

```
s = 0
count = 0
for val in data:
   s += val
   count += 1
average = s / count

print("The average of the values in the list",
data,"is", average)
```

For loops

Example: Assume that we are given a list of decimal values in a Python list, named `data`. And we are asked to write a program that finds the average of all the values in the list `data`.

Here is a program for that:

```
s = 0
count = 0
for val in data:
   s += val
   count += 1
average = s / count

print("The average of the values in the list",
data,"is", average)
```

How can we simplify the program?

hint: I don't need to count the number of values in the list `data`

For loops

Example: Assume that we are given a list of decimal values in a Python list, named `data`. And we are asked to write a program that finds the average of all the values in the list `data`.

Here is a program for that:

```
s = 0
count = 0
for val in data:
    s += val
    count += 1
average = s / count
print("The average of the values in the list",
data,"is", average)
```

How can we simplify the program?

hint: I don't need to count the number of values in the list `data`

```
use len
```

For loops

Example: Assume that we are given a list of decimal values in a Python list, named `data`. And we are asked to write a program that finds the average of all the values in the list `data`.

Here is a program for that:

```
s = 0
for val in data:
   s += val
average = s / len(data)

print("The average of the values in the list",
data,"is", average)
```

`For` loops

Example: Assume that we are given a list of decimal values in a Python list, named `data`. And we are asked to write a program that finds the average of all the values in the list `data`.

In-class work:

Exercise 1: modify the program we wrote to find the average of all negative numbers in the list `data`.
Send the code to my e-mail.

For loops

Example: Now let's consider a small database with names associated with the e-mail addresses:

```
contactInfo = {
"Mark Huggard" : "Mhuggard@org.com",
"Alice True" : "ATrue@org2.com",
"Hunter O'Brien" : "HOBrien@org.com",
"Jane Cole" : "JaneCole@org6.com",
"Frank Dove" : "FrankDove@org7.com"
}
```

For loops

Example: Now let's consider a small database with names associated with the e-mail addresses:

```
contactInfo = {
"Mark Huggard" : "Mhuggard@org.com",
"Alice True" : "ATrue@org2.com",
"Hunter O'Brien" : "HOBrien@org.com",
"Jane Cole" : "JaneCole@org6.com",
"Frank Dove" : "FrankDove@org7.com"
}
```

Now, let's write the code fragment to:
1) extract all e-mail addresses from `contactInfo` and store them in a Python list, named `emails`, then
2) display alphabetically sorted list of e-mails.

For loops

Example: Now let's consider a small database with names associated with the e-mail addresses:

```
contactInfo = {
"Mark Huggard" : "Mhuggard@org.com",
"Alice True" : "ATrue@org2.com",
"Hunter O'Brien" : "HOBrien@org.com",
"Jane Cole" : "JaneCole@org6.com",
"Frank Dove" : "FrankDove@org7.com"
}

emails = []
for k in contactInfo: # extract all e-mail addresses
   emails.append(contactInfo[k])
emails.sort() # sort alphabetically
print(emails) # display
```

Counting using the `range()` function

`range()` function

Range	Generated sequence	Explanation
range(5)	0 1 2 3 4	every integer from 0 to 4
range(0, 5)	0 1 2 3 4	every integer from 0 to 4
range(3, 7)	3 4 5 6	every integer from 3 to 6
range(10, 13)	10 11 12	every integer from 10 to 12
range(0, 5, 1)	0 1 2 3 4	every 1 integers from 0 to 4
range(0, 5, 2)	0 2 4	every 2 integers from 0 to 4
range(5, 0, -1)	5 4 3 2 1	every 1 integer from 5 to 1
range(5, 0, -2)	5 3 1	every 2 integers from 5 to 1

Counting using the `range()` function

Example: Prices of various items change with time. Let's write the program that will allow us to calculate the price of the smart phone few years later, if we are given this year price, and each year's inflation rate in percent.

```
price = float(input("Enter phone's price:"))
n = int(input("Enter the number of years:"))
rate = float(input("Enter the inflation
rate:"))
rate_decimal = rate/100
```

Counting using the `range()` function

Example: Prices of various items change with time. Let's write the program that will allow us to calculate the price of the smart phone few years later, if we are given this year price, and each year's inflation rate in percent.

```
price = float(input("Enter phone's price:"))
n = int(input("Enter the number of years:"))
rate = float(input("Enter the inflation
rate:"))
rate_decimal = rate/100

newPrice = price
for i in range(0,n):
  newPrice = newPrice + newPrice*rate_decimal

print("The phone's price will be $%f" %
newPrice)
```

Counting using the `range()` function

In-class activity

Do the exercises 4-6 from the handout

While vs. for loops

Consider the following scenarios:

(a) iterate until the user enters a letter ‘p’ or a letter ‘q’

(b) iterate 50 times

(c) iterate until x is less than 19

(d) find the product of all values in a list

What looping mechanism to use? while or for?

While vs. for loops

Consider the following scenarios:

(a) iterate until the user enters a letter ‘p’ or a letter ‘q’
    while loop

(b) iterate 50 times
    for loop

(c) iterate until x is less than 19
    while loop

(d) find the product of all values in a list
    for loop

What looping mechanism to use? while or for?

While vs. for loops

Here is a guideline:

- The number of iterations is computable before entering the loop → use the for loop
  examples: counting down from X to 0, printing a string N times, etc.

- Accessing the elements of a container → use the for loop
  examples: when adding 1 to every element in a list, or printing the key of every entry in a dict, etc.

- The number of iterations is not computable before entering the loop→ use the while
  example: when iterating until a user enters a particular character.

While vs. for loops

Example: Given a string **`myString`**, let's write a program to count the number of occurrences of letter 'a' in it, without using built-in function `count()`.

While vs. for loops

Example: Given a string **`myString`**, let's write a program to count the number of occurrences of letter 'a' in it, without using built-in function `count()`.

1) What looping mechanism to use (while or for)?

While vs. for loops

Example: Given a string **`myString`**, let's write a program to count the number of occurrences of letter 'a' in it, without using built-in function `count()`.

1) What looping mechanism to use (while or for)? for loop

While vs. for loops

Example: Given a string **`myString`**, let's write a program to count the number of occurrences of letter 'a' in it, without using built-in function `count()`.

1) What looping mechanism to use (while or for)? for loop

2) let's write the code:

```
counter = 0
for ch in myString:
   if ch == 'a': counter += 1
# at the end of loop's executions, variable
# counter will have the number of occurrences
# of letter 'a' in myString
```

While vs. for loops

In-class activity

Do the exercises 4-5 from the handout

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
   print()
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 10

y → 1

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
    →print(x+y,end = " ")
    print()
```

x → 10

y → 1

```
11
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

y → 2

```
11 12
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 10

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
   print()
```

x → 20

```
11 12 13 14 15 16 17 18 19
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 20

y → 1

```
11 12 13 14 15 16 17 18 19
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
    →print(x+y,end = " ")
   print()
```

x → 20

y → 1

```
11 12 13 14 15 16 17 18 19
21
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 20

y → 2

```
11 12 13 14 15 16 17 18 19
21
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
  ⟶ print(x+y,end = " ")
   print()
```

x → 20

y → 2

```
11 12 13 14 15 16 17 18 19
21 22
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
  ⟶ print(x+y,end = " ")
   print()
```

x → 20

y → 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 30

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 30

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 30

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
  print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
  print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

How many entries are in the table on the left?

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
  print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

How many entries are in the table on the left?

$4 \times 9 = 36$

Nested loops

Example: two-letter domain names

```
print('Two-letter domain names:')

letter1 = 'a'
letter2 = '?'

while letter1 <= 'z':  # Outer loop

    letter2 = 'a'

    while letter2 <= 'z':  # Inner loop
        print('%s%s.info' % (letter1, letter2))
        letter2 = chr(ord(letter2) + 1)
    letter1 = chr(ord(letter1) + 1)
```

Nested loops

Example: two-letter domain names

```
print('Two-letter domain names:')

letter1 = 'a'
letter2 = '?'

while letter1 <= 'z':  # outer loop

    letter2 = 'a'

    while letter2 <= 'z':  # inner loop
        print('%s%s.info' %(letter1, letter2))
        letter2 = chr(ord(letter2) + 1)
    letter1 = chr(ord(letter1) + 1)
```

```
Two-letter domain names:
aa.info
ab.info
ac.info
ad.info
ae.info
af.info
ag.info
ah.info
...
zx.info
zy.info
zz.info
```

Break and continue

A break statement in a loop causes an immediate exit from the loop.

```
while True:
   x = int(input("Enter an integer > 10:"))
   if x > 10:
      break
print("you made it!")
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 1
i = 5
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 1
i = 5
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

```
s = 1
i = 5

5 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 1
i = 5

5 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
   → continue

print("s =", s)
```

```
s = 1
i = 5

5 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 1
i = 6
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

```
s = 1
i = 6

6 % 2 = 0
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

s = s*(6//2) = 1*3 = 3
i = 6

6 % 2 = 0

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 3
i = 7
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

```
s = 3
i = 7

7 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 3
i = 7

7 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 3
i = 8
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

```
s = 3
i = 8

8 % 2 = 0
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

s = 3 * (8//2) = 12
i = 8

8 % 2 = 0

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 12
i = 9

.....
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s =  181440
i = 19
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

s = 181440
i = 19

s = 181440

Nested loops, break and continue

In-class activity

Do the exercises 9-11 from the handout

This OER material was produced as a result of the CS04ALL CUNY OER project.