Lecture 22

Topics: *Chapter 10. Defining Classes*
Chapter 11. Data Collections

More about classes

11.1 Example problem: simple statistics

11.2 Applying lists

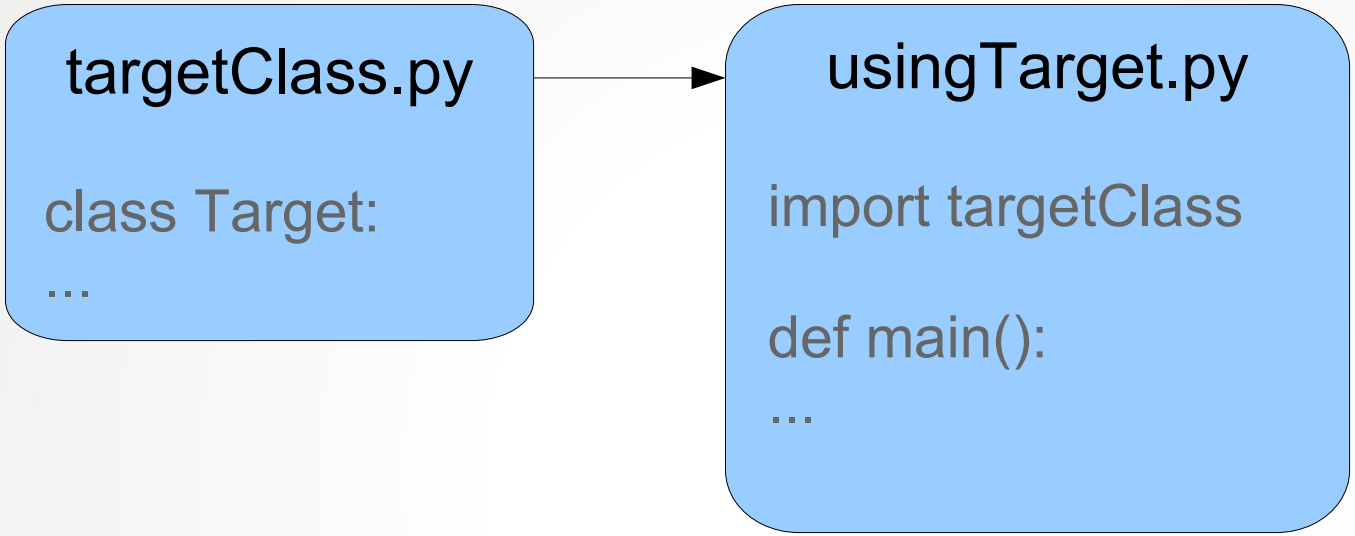
11.3 Lists of records

More about classes

Let's consider defining a `Target` class and using it.

Target		
self.window self.anchor self.primary	self.r self.n self.secondary	self.rings
__init__(self,window, point,radius,rings,primary_color="black", secondary_color="white") show (self) hide (self) move(self,dx,dy)		

More about classes



see targetClass.py and usingTarget.py

More about classes

A convention

We mentioned that instance variables should only be accessed or modified through the interface methods of the class, i.e.

bad:

```
item = Thing(…)
item.x = 10
```

good:

```
item = Thing(...)
item.setX(10)
```

Therefore, it is convenient to mark the instance variables as *"private"* by using an underscore (_) to begin the instance variable name with.

Same convention for *"private"* methods of the class.

More about classes: A convention example

```
class Thing:
  def __init__(self):
    self._name = "my Name"
    self._age = 28
    self._phone = "(718) 465-3576"
  def setName(self,newName):
    self._name = newName
  def getName(self):
    return self._name
  def setAge(self,newAge):
    self._age = newAge
  def getAge(self):
    return self._age
  def setPhone(self,newPhone):
    self._phone = newPhone        ...
```

~~print(person._name,person._age,person._phone)~~

```
print(person.getName(),person.getAge(),person.getPhone())
```

```
class Thing:
  def __init__(self):
    self._name = "my Name"
    self._age = 28
    self._phone = "(718) 465-3576"
  def setName(self,newName):
    self._name = newName
  def getName(self):
    return self._name
  def setAge(self,newAge):
    self._age = newAge
  def getAge(self):
    return self._age
  def setPhone(self,newPhone):
    self._phone = newPhone
```

...

More about classes: A convention example

```
class Thing:
  def __init__(self):
    self._name = "my Name"
    self._age = 28
    self._phone = "(718) 465-3576"
  def setName(self,newName):
    self._name = newName
  def getName(self):
    return self._name
  def setAge(self,newAge):
    self._age = newAge
  def getAge(self):
    return self._age
  def setPhone(self,newPhone):
    self._phone = newPhone
```

~~person._age = 70~~

person.setAge(40)

...

More about classes

~~person._phone = "(718) 675-7684"~~

person.setPhone("(718) 675-7685")

```
class Thing:
  def __init__(self):
    self._name = "my Name"
    self._age = 28
    self._phone = "(718) 465-3576"
  def setName(self,newName):
    self._name = newName
  def getName(self):
    return self._name
  def setAge(self,newAge):
    self._age = newAge
  def getAge(self):
    return self._age
  def setPhone(self,newPhone):
    self._phone = newPhone
```

...

11.1 Example problem: simple statistics

Classes alone are not enough to satisfy all of our data-handling needs.

Many real-world programs deal with large collections of similar information:

- Words in a document
- Students in a course
- Data from an experiment
- Customers of a business
- Cards in a deck

In Chapter 11 we learn techniques that help us manipulate collections like these.

11.1 Example problem: simple statistics

Simple Statistics Program

Let's write a program that will compute the *average* (*mean*), the *median*, and the *standard deviation*.

The sequence of numbers will be read from a file.

11.1 Example problem: simple statistics

Simple Statistics Program

Let's write a program that will compute the *average* (*mean*), the *median*, and the *standard deviation*.

The sequence of numbers will be read from a file.

[Def] The *mean* (*average*) of *n* values is their sum divided by *n*.

1, 4, 7, 9, 12, 10 $\longrightarrow$ $\frac{43}{6} \approx 7.27$

11.1 Example problem: simple statistics

Simple Statistics Program

Let's write a program that will compute the *average* (*mean*), the *median*, and the *standard deviation*.

The sequence of numbers will be read from a file.

[Def] The *mean* (*average*) of *n* values is their sum divided by *n*.

1, 4, 7, 9, 12, 10 $\longrightarrow$ $\frac{43}{6} \approx 7.27$

[Def] The *median* of an ordered collection of values is the middle number. If there are two middle numbers then their average is taken.

1, 4, 7, 9, 10, 12 $\longrightarrow$ $\frac{7+9}{2} = 8$

1, 4, 6, 7, 9, 10, 12 $\longrightarrow$ 7

11.1 Example problem: simple statistics

Simple Statistics Program

Let's write a program that will compute the *average* (*mean*), the *median*, and the *standard deviation*.

The sequence of numbers will be read from a file.

[Def] The *standard deviation* *s*, is defined as

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

Where $\bar{x}$ is the mean, x_i represents the i^{th} data value, and n is the number of data values.

11.1 Example problem: simple statistics

Simple Statistics Program

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

Example of calculations:

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5		
x_2	3		
x_3	1		
x_4	6		
x_5	7		
x_6	9		
x_7	11		
SUM:	42		

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5		
x_2	3		
x_3	1		
x_4	6		
x_5	7		
x_6	9		
x_7	11		
SUM:	42		

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	
x_2	3		
x_3	1		
x_4	6		
x_5	7		
x_6	9		
x_7	11		
SUM:	42		

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	
x_2	3	6 – 3 = 3	
x_3	1		
x_4	6		
x_5	7		
x_6	9		
x_7	11		
SUM:	42		

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	
x_2	3	6 – 3 = 3	
x_3	1	6 – 1 = 5	
x_4	6		
x_5	7		
x_6	9		
x_7	11		
SUM:	42		

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	
x_2	3	6 – 3 = 3	
x_3	1	6 – 1 = 5	
x_4	6	6 – 6 = 0	
x_5	7	6 – 7 = -1	
x_6	9	6 – 9 = -3	
x_7	11	6 – 11 = -5	
SUM:	42		

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	1
x_2	3	6 – 3 = 3	9
x_3	1	6 – 1 = 5	25
x_4	6	6 – 6 = 0	0
x_5	7	6 – 7 = -1	1
x_6	9	6 – 9 = -3	9
x_7	11	6 – 11 = -5	25
SUM:	42		

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	1
x_2	3	6 – 3 = 3	9
x_3	1	6 – 1 = 5	25
x_4	6	6 – 6 = 0	0
x_5	7	6 – 7 = -1	1
x_6	9	6 – 9 = -3	9
x_7	11	6 – 11 = -5	25
SUM:	42		70

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

$$s = \sqrt{\frac{70}{6}} \approx 3.42$$

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	1
x_2	3	6 – 3 = 3	9
x_3	1	6 – 1 = 5	25
x_4	6	6 – 6 = 0	0
x_5	7	6 – 7 = -1	1
x_6	9	6 – 9 = -3	9
x_7	11	6 – 11 = -5	25
SUM:	42		70

median

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

$$s = \sqrt{\frac{70}{6}} \approx 3.42$$

11.1 Example problem: simple statistics

Simple Statistics Program

Example of calculations:

	x	$\bar{x}$-x_i	$(\bar{x}$-$x_i)^2$
x_1	5	6 – 5 = 1	1
x_2	3	6 – 3 = 3	9
x_3	1	6 – 1 = 5	25
x_4	6	6 – 6 = 0	0
x_5	7	6 – 7 = -1	1
x_6	9	6 – 9 = -3	9
x_7	11	6 – 11 = -5	25
SUM:	42		70

$$s = \sqrt{\frac{\sum (\bar{x} - x_i)^2}{n-1}}$$

$$\bar{x} = \frac{42}{7} = 6$$

$$s = \sqrt{\frac{70}{5}} \approx 3.42$$

Answer:
mean $\bar{x} = 6$
median is *6*
standard deviation
$s \approx 3.42$

11.2 Applying lists

Simple Statistics Program

Let's write a program that will compute the *average* (*mean*), the *median*, and the *standard deviation*.

The sequence of numbers will be read from a file.

Design / Outline of the program:
get file name from the user,
read data from file, return list of values (sorted), `readData(fname)`
close the file,
find the mean, `getMean(listOfValues)`
find the median, `getMedian(listOfValues)`
find the standard deviation, `getS(listOfValues)`

see simpleStats.py

11.2 Applying lists

Lists review

Python lists provide very flexible mechanism for handling arbitrarily large sequences of data.

- A list is a sequence of items stored in a single object
- Items in a list can be accessed by indexing, and sublists can be accessed by slicing (*see page 367*)
- Lists are mutable; individual items or entire slices can be replaced through assignments statements
- Lists support a number of convenient and frequently used methods (*see page 369*)
- Lists will grow and shrink as needed

11.3 Lists of records

Recall the constructor of the `Target` class:

```
...
self.rings = []

step = round(self.r / self.n)

for i in range(self.n):
    ring = Circle(self.anchor,self.r - i*step)
    if i%2 == 0:
        ring.setFill(self.primary)
    else:
        ring.setFill(self.secondary)
    self.rings.append(ring)
...
```

A list of circles is generated

11.3 Lists of records

We can also create a list of student's records and sort them by their GPA.

Let's write a program that will sort a file of students according to their GPA.

11.3 Lists of records

We can also create a list of student's records and sort them by their GPA.

Let's write a program that will sort a file of students according to their GPA.

Design / basic algorithm:
get the file name
read student information into a list
sort the list by GPA
get the output file name
write the sorted student information into a file

We will borrow the definition of the `Student` class and a stand-alone method `makeStudent` from `studentsGPA.py`
(see Lecture 21)

11.3 Lists of records

sorting the list by GPA

```
records.sort(key = Student.getGPA)
```

list of objects of type `Student`

built-in method

a way to specify based on what to order the elements (key must be a function that takes an item from the list `records` *and returns a value)*

method of class `Student` *that returns a float value*

see student.py and studentOrderingByGPA.py