Algorithmic Approaches for Biological Data, Lecture #5

Katherine St. John

City University of New York
American Museum of Natural History

3 February 2016

Outline

More on Loops

- Indefinite Loops

Outline

More on Loops

- Indefinite Loops
- Looping Through Strings

Review: `for`-loops

Standard form:

`for x in <list>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Roughly, the `for` loop:

Standard form:

`range(stop)`
`range(start,stop)`
`range(start, stop, step)`

Review: `for`-loops

Standard form:

`for x in <list>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Standard form:

`range(stop)`
`range(start,stop)`
`range(start, stop, step)`

Roughly, the `for` loop:

1. assigns next value of list to `x`,

Review: `for`-loops

Standard form:

`for x in <list>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Standard form:

`range(stop)`
`range(start,stop)`
`range(start, stop, step)`

Roughly, the `for` loop:

1. assigns next value of list to `x`,
2. does the body statements, and

Review: `for`-loops

Standard form:

`for x in <list>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Standard form:

`range(stop)`
`range(start,stop)`
`range(start, stop, step)`

Roughly, the `for` loop:

1. assigns next value of list to `x`,
2. does the body statements, and
3. then if there's still list items goes back to #1; else ends loop.

Another Type of Loop

- Allows you to repeat a non-fixed number of times.

Another Type of Loop

- Allows you to repeat a non-fixed number of times.
- Form of indefinite loops.

Another Type of Loop

- Allows you to repeat a non-fixed number of times.
- Form of indefinite loops.
- Example: checking input for errors:

```
num = input('Please enter a positive number:  ')
```

Another Type of Loop

- Allows you to repeat a non-fixed number of times.
- Form of indefinite loops.
- Example: checking input for errors:

```
num = input('Please enter a positive number:  ')
while num <=0:
```

Another Type of Loop

- Allows you to repeat a non-fixed number of times.
- Form of indefinite loops.
- Example: checking input for errors:

```
num = input('Please enter a positive number:  ')
while num <=0:
    print "Entered negative number"
```

Another Type of Loop

- Allows you to repeat a non-fixed number of times.
- Form of indefinite loops.
- Example: checking input for errors:

```
num = input('Please enter a positive number:  ')
while num <=0:
    print "Entered negative number"
    num = input('Please enter a positive number:  ')
```

Another Type of Loop

- Allows you to repeat a non-fixed number of times.
- Form of indefinite loops.
- Example: checking input for errors:

```
num = input('Please enter a positive number:  ')
while num <=0:
    print "Entered negative number"
    num = input('Please enter a positive number:  ')

print "Thank you!  Number entered is", num
```

while loops

Standard form:

`while <test>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Roughly, the `while` loop:

while loops

Standard form:

`while <test>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Roughly, the `while` loop:

1. checks the test condition

while loops

Standard form:

`while <test>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Roughly, the `while` loop:

1. checks the test condition
2. if true, does body statements & goes back to #1

while loops

Standard form:

`while <test>:`
`  command1`
`  command2`
`  ...`
`  commandN`

Roughly, the `while` loop:

1. checks the test condition
2. if true, does body statements & goes back to #1
3. if false, leaves loop
   else ends loop.

More examples

- `PythonTutor` example

Standard form:
`while <test>:` command1 command2 ... commandN

More examples

- `PythonTutor` example
- Newton's Method in `ThinkCS`

Standard form:

`while <test>:`
  command1
  command2
  ...
  commandN

More examples

Standard form:
`while <test>:` command1 command2 ... commandN

- `PythonTutor` example
- Newton's Method in `ThinkCS`

  Idea: Computes square root using the approximation:

  `better = 1/2*(approx + n/approx)`

More examples

Standard form:

`while <test>:`
`  command1`
`  command2`
`  ...`
`  commandN`

- `PythonTutor` example
- Newton's Method in `ThinkCS`

  Idea: Computes square root using the approximation:

  `better = 1/2*(approx + n/approx)`

  - If n is 10, start with approx $= 10/2 = 5$

More examples

Standard form:

`while <test>:`
  command1
  command2
  ...
  commandN

- `PythonTutor` example
- Newton's Method in `ThinkCS`

  Idea: Computes square root using the approximation:

  `better = 1/2*(approx + n/approx)`

  - If n is 10, start with approx $= 10/2 = 5$
  - better $= 0.5\ (5 + 10/5) = 0.5\ (7) = 3.5$

More examples

Standard form:

`while <test>:`
`  command1`
`  command2`
`  ...`
`  commandN`

- `PythonTutor` example
- Newton's Method in `ThinkCS`

  Idea: Computes square root using the approximation:

  `better = 1/2*(approx + n/approx)`

  - If n is 10, start with approx $= 10/2 = 5$
  - better $= 0.5\ (5 + 10/5) = 0.5\ (7) = 3.5$
  - better $= 0.5(3.5 + 10/3.5) = 3.1786$

More examples

Standard form:

`while <test>:`
  command1
  command2
  ...
  commandN

- `PythonTutor` example
- Newton's Method in `ThinkCS`

  Idea: Computes square root using the approximation:

  `better = 1/2*(approx + n/approx)`

  - If n is 10, start with approx $= 10/2 = 5$
  - better $= 0.5\ (5 + 10/5) = 0.5\ (7) = 3.5$
  - better $= 0.5(3.5 + 10/3.5) = 3.1786$
  - better $= \ldots = 3.1623$

More examples

Standard form:

`while <test>:`
`  command1`
`  command2`
`  ...`
`  commandN`

- `PythonTutor` example
- Newton's Method in `ThinkCS`

  Idea: Computes square root using the approximation:

  `better = 1/2*(approx + n/approx)`

  - If n is 10, start with approx $= 10/2 = 5$
  - better $= 0.5\ (5 + 10/5) = 0.5\ (7) = 3.5$
  - better $= 0.5(3.5 + 10/3.5) = 3.1786$
  - better $= \ldots = 3.1623$
  - ...

Group Work

In pairs/triples: Our randomly walking turtle (from the first day) can wander off the screen. Modify that program so that:

Group Work

In pairs/triples: Our randomly walking turtle (from the first day) can wander off the screen. Modify that program so that:

- The program stops if the turtle goes farther than x = 100 or y = 100, and similarly, in the negative direction x = -100 or y = -100.

Group Work

In pairs/triples: Our randomly walking turtle (from the first day) can wander off the screen. Modify that program so that:

- The program stops if the turtle goes farther than x = 100 or y = 100, and similarly, in the negative direction x = -100 or y = -100.
- That is, the turtle continues walking only when:
  $-100 < x < 100$ and $-100 < y < 100$

Group Work

In pairs/triples: Our randomly walking turtle (from the first day) can wander off the screen. Modify that program so that:

- The program stops if the turtle goes farther than x = 100 or y = 100, and similarly, in the negative direction x = -100 or y = -100.
- That is, the turtle continues walking only when:
  -100 $<$ x $<$ 100 and -100 $<$ y $<$ 100
- Some useful turtle methods are:
  `turtle.xcor()` and `turtle.ycor()`.

Looping Through Strings (preview)

What do these do? (PythonTutor example)

- ```
  for ch in "Teddy Roosevelt":
      print ch
  ```
- ```
  name = raw_input('Please enter your name:  ')
  for c in name:
      print c, "!",
  ```

Recap

- Lab today at 3pm.

Recap

- Lab today at 3pm.
- Email lab reports to `kstjohn@amnh.org`

Recap

- Lab today at 3pm.
- Email lab reports to `kstjohn@amnh.org`
- Challenges available at `rosalind.info` (use emailed link to access course page).