2/7/2008

>>> Overview

* boolean
* while
* random
* tuples

>>> boolean

Just like Java, there are boolean values.
These values are True and False.

True
False

<
>
<=
>=
==
!=
or
and
not

```
>>> True
True
>>> False
False
>>> 2==3
False
>>> "this"=="this"
True
>>> 2==3 and 4==4
False
>>> x = not 1 == 2
>>> x
True
```

>>> while

The while loop translates nicely from Java to Python.

sentinel.py

```
1   sum = 0
2   number = input(“Enter a number (-1 to quit)? ")
3
4   while number !=  -1:
5           sum += number
6           number = input(" Enter a number (-1 to quit)? ")
7
8   print "The total is " + str(sum)
9
10
```

Sentinel.java

```
1       Scanner console = new Scanner(System.in);
2       int sum = 0;
3       System.out.print("Enter a number (-1 to quit): ");
4       int number = console.nextInt();
5
6       while (number != -1) {
7           sum = sum + number;
8           System.out.print("Enter a number (-1 to quit): ");
9           number = console.nextInt();
10      }

        System.out.println("The total is " + sum);
```

>>> random

Just like in Java, python also has random object. Here is an example:

```
>>> from random import *
>>> randint(0,9)
1
>>> randint(0,9)
4
>>> choice(range(10))
7
```

random.**randint**(a,b)

returns an int between a and b inclusive

random.**choice**(seq)

returns a random element of the sequence

>>> tuples as points

Python does not have Point Objects. Instead we use tuples. A tuple is able to hold multiple values. These values can correspond to the x and y coordinates of a point.

The syntax for a tuple is:

<variable name> = (value1, value 2, ..., valueN)

For a point, we only need two values.

```
>>> p = (3, 5)
>>> p
(3, 5)
```

Creates a tuple where the first value is 3 and the second value is 5. This can represent a 2D point where the “x” value is 3 and the “y” value is 5.

>>> retrieving tuple values

If we wish to use the values in a tuple, we can assign each value to a vairable.

```
>>> p = (3, 5)
>>> p
(3, 5)

>>> (x, y) = p
>>> x
3
>>> y
5
```

This creates two new variables x and y, and assigns the first value in our tuple to x, and the second value to y.

>>> parameters and returns

Tuples can be passed just like any other variable. Once inside a method, we will want to access its values.

Example:

```
def equal(p1, p2):
    (x1, y1) = p1
    (x2, y2) = p2
    return x1==x2 and y1==y2
```

Additionally, we can return tuples. Assume we wanted to add two two. This does not make much sense for points, but does for 2D vectors.

```
def addVectors(p1, p2):
    (x1, y1) = p1
    (x2, y2) = p2
    return (x1 + x2, y1 + y2)
```

NOTE: Tuples are “immutable.” This means that the values within a tuple cannot be altered once it has been created. Because of this, if we would like to change the value of our tuples, we must create a new tuple with the values we want, and use it instead.

>>> point distance method

```
# Calculates the distance between two points
def distance(p1, p2):
        (x1, y1) = p1
        (x2, y2) = p2
        dx = abs(x1 - x2)
        dy = abs(y1 - y2)
        return sqrt(dx * dx + dy * dy)
```

>>> mini-yahtzee

```
# plays until 3 dice have the same value

from random import *

def miniYahtzee():
        d1 = 0
        d2 = 1
        d3 = 2
        count = 0

        while not(d1 == d2 == d3):
                d1 = randint(1, 6)
                d2 = randint(1, 6)
                d3 = randint(1, 6)
                print str(d1), str(d2), str(d3)
                count += 1

        print “Mini-Yahtzee in” + str(count) + “moves”
```

>>> graphic example - rectangles

```
from random import *
from drawingpanel import *

def drawRandomRect():
    x = randint(0,490)
    y = randint(0,490)
    randomColor = choice(("red", "orange", "yellow", "green", "blue", "purple"))
    size = randint(1,100)

    g.create_rectangle(x, y, x+size, y+size, fill=randomColor)

    return randomColor == "red"

#main
panel = DrawingPanel(500, 500)
g = panel.get_graphics()
reds = 0

while reds < 20:
    if drawRandomRect():
        reds += 1
```

>>> Homework #5

Random walk is becoming random slither!

- No DEBUG mode
- Remember to use raw_input() for gathering a whole string of user input
- Random-Slither will be green and will change shades of green.
- Colors can be represented as RGB tuples
- Since Tkinter takes Strings as color arguments, our tuple needs to be converted to a String of hex values (like web colors)

Example

```
red = 0
green = 255
 blue = 0
 hexColor = "#%02x%02x%02x" % (red, green, blue)
create_oval(0, 0, 100, 100, fill=hexColor, outline=hexColor)
```

To create a single pixel, make a rectangle where x1 equals x2 and y1 equals y2:

```
create_rectangle(50, 50, 50, 50)
```