



Week 5

`while` loops; logic; random numbers; tuples

Special thanks to Scott Shawcroft, Ryan Tucker, and Paul Beck for their work on these slides.

Except where otherwise noted, this work is licensed under:

<http://creativecommons.org/licenses/by-nc-sa/3.0>

while Loops

```
while test:  
    statements
```

```
>>> n = 91  
>>> factor = 2      # find first factor of n  
  
>>> while n % factor != 0:  
...     factor += 1  
...  
  
>>> factor  
7
```

while / else

```
while test:  
    statements
```

```
else:  
    statements
```

- Executes the `else` part if the loop never enters
- There is also a similar `for / else` statement

```
>>> n = 91  
>>> while n % 2 == 1:  
...     n += 1  
...     else:  
...         print n, "was even; no loop."  
...  
91 was even; no loop.
```

bool

- Python's logic type, equivalent to `boolean` in Java
 - `True` and `False` start with capital letters

```
>>> 5 < 10
True

>>> b = 5 < 10
>>> b
True

>>> if b:
...     print "The bool value is true"
...
The bool value is true

>>> b = not b
>>> b
False
```

Logical Operators

Operator	Meaning	Example	Result
<code>==</code>	equals	<code>1 + 1 == 2</code>	True
<code>!=</code>	does not equal	<code>3.2 != 2.5</code>	True
<code><</code>	less than	<code>10 < 5</code>	False
<code>></code>	greater than	<code>10 > 5</code>	True
<code><=</code>	less than or equal to	<code>126 <= 100</code>	False
<code>>=</code>	greater than or equal to	<code>5.0 >= 5.0</code>	True

Operator	Example	Result
<code>and</code>	<code>2 == 3 and -1 < 5</code>	False
<code>or</code>	<code>2 == 3 or -1 < 5</code>	True
<code>not</code>	<code>not -1 < 5</code>	False

Random Numbers

```
from random import *
```

```
randint(min, max)
```

- returns a random integer in range [**min**, **max**] inclusive

```
choice(sequence)
```

- returns a randomly chosen value from the given sequence
 - the sequence can be a range, a string, ...

```
>>> from random import *
>>> randint(1, 5)
2
>>> randint(1, 5)
5
>>> choice(range(4, 20, 2))
16
>>> choice("hello")
'e'
```

Exercise

- Rewrite the `Dice` program from Java to Python:

```
2 + 4 = 6
```

```
3 + 5 = 8
```

```
5 + 6 = 11
```

```
1 + 1 = 2
```

```
4 + 3 = 7
```

```
You won after 5 tries!
```

Tuple

tuple_name = (value, value, ..., value)

- A way of "packing" multiple values into one variable

```
>>> x = 3
>>> y = -5
>>> p = (x, y, 42)
>>> p
(3, -5, 42)
```

name, name, ..., name = tuple_name

- "unpacking" a tuple's contents into multiple variables

```
>>> a, b, c = p
>>> a
3
>>> b
-5
>>> c
42
```


Using Tuples

- Useful for storing multi-dimensional data (e.g. (x, y) points)

```
>>> p = (42, 79)
```

- Useful for returning more than one value

```
>>> from random import *
>>> def roll2():
...     die1 = randint(1, 6)
...     die2 = randint(1, 6)
...     return (die1, die2)
...
>>> d1, d2 = roll2()
>>> d1
6
>>> d2
4
```

Tuple as Parameter

```
def name( (name, name, ..., name), ... ):  
    statements
```

- Declares tuple as a parameter by naming each of its pieces

```
>>> def slope((x1, y1), (x2, y2)):  
...     return (y2 - y1) / (x2 - x1)  
...  
>>> p1 = (2, 5)  
>>> p2 = (4, 11)  
>>> slope(p1, p2)  
3
```

Tuple as Return

```
def name (parameters) :  
    statements  
    return (name, name, ..., name)
```

```
>>> from random import *  
>>> def roll2():  
...     die1 = randint(1, 6)  
...     die2 = randint(1, 6)  
...     return (die1, die2)  
...  
>>> d1, d2 = roll2()  
>>> d1  
6  
>>> d2  
4
```

Exercise

- Write a program that performs a graphical "random walk".
 - Create a DrawingPanel of size 150x150.
 - Draw a circle of radius 75.
 - Start a "walker" pixel at (75, 75), the circle's center.
 - Every 10 ms, randomly move the walker by 1 pixel in either the x or y direction and redraw the walker.
 - Once the walker reaches the edge of the circle, stop walking.
- Key questions:
 - How do we randomly move by 1 pixel?
 - How do we know when we have reached the edge of the circle?

