



Lecture 5

`while` loops; logic; random numbers; tuples

Revised from CS Washington Lecture notes
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 does not enter
- There is also a similar `for / else` statement

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

Loop break

- The **break statement** in **Python** terminates the current **loop** and resumes execution at the next **statement**, just like the traditional **break** found in C. The most common use for **break** is when some external condition is triggered requiring a hasty exit from a **loop**. The **break statement** can be used in both **while** and **for loops**.

```
for letter in 'Python':    # First Example
    if letter == 'h':
        break
    print ('Current Letter :', letter)
```

while loop Exercise

- Write a Python program to find the product of all odd number from 1 to 100 using while instead of for 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
```

Random Numbers

```
from random import *
```

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

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

- Write a `Dice` program to roll two dices until the sum reaches 7.

2 + 4 = 6

3 + 5 = 8

5 + 6 = 11

1 + 1 = 2

4 + 3 = 7

You won after 5 tries!

Hint: 1) import random package, use `randint` to generate two random integers from 1 to 6
2) If the sum of these two is 7, break the loop, otherwise, continue to read numbers.

Exercise 2

- Write a `guess` program to guess the correct number generated by the computer from 1 to 100.

The output of the monitor:

I'm thinking of a number between 1 and 100...

Your guess? 50

It's lower.

Your guess? 25

It's lower.

Your guess? 10

It's lower.

Your guess? 5

It's higher.

Your guess? 7

You got it right in 5 guesses

Do you want to play again? y

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 `Dice` program to roll two dices until the sum reaches 7.

2 + 4 = 6

3 + 5 = 8

5 + 6 = 11

1 + 1 = 2

4 + 3 = 7

You won after 5 tries!

Rewrite this exercise define a function to roll two dices and return them as a tuple