Review

WTP 2010
Computer Science
Day 6: Monday, July 5

If Statement

```
....
if (player1 == 'rock') and (player2 == 'scissors'):
    print 'Player 1'
if (player1 == 'paper') and (player2 == 'rock'):
    print 'Player 1'
if (player1 == 'scissors') and (player2 == 'paper'):
    print 'Player 1'
....
```

What kind of operator is **==** ?

What kind of operator is **and** ?

If Statement

```
....
if (player1 == 'rock') and (player2 == 'scissors'):
    print 'Player 1'
elif (player1 == 'paper') and (player2 == 'rock'):
    print 'Player 1'
elif (player1 == 'scissors') and (player2 == 'paper'):
    print 'Player 1'
....
```


If Statement

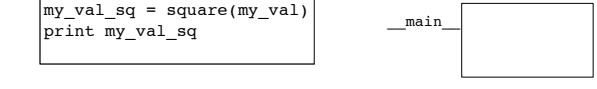
```
....
if ((player1 == 'rock' and player2 == 'scissors') or
    (player1 == 'paper' and player2 == 'rock') or
    (player1 == 'scissors' and player2 == 'paper')):
    print 'Player 1'
....
```


Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```

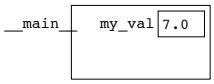

Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



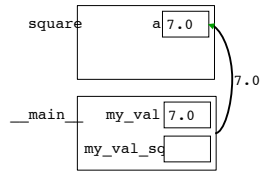
Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



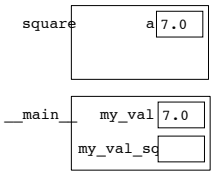
Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



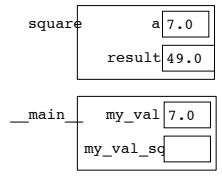
Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



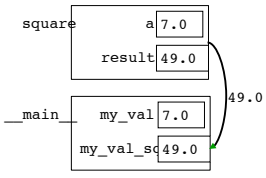
Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



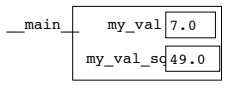
Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



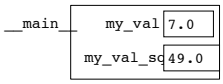
Review of function calling

```
def square(a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



Review of function calling

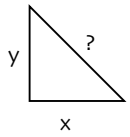
```
def square(double a):
    result = a * a
    return result

my_val = 7.0
my_val_sq = square(my_val)
print my_val_sq
```



49

Stack diagram exercise…

Write function **hypotenuse(x,y)** to find the hypotenuse of a right triangle, where x & y are the sides of the right angle.
Remember to make use of the **square(a)** function you saw earlier, and assume there is a function called **square_root(a)** defined for you.



Draw the stack diagram for hypotenuse(3,4)

Stack diagram exercise…

Code Organization

```
import MODULENAME

def method1():
   BODY1

def method2(a, b):
   BODY2

...
def methodn(a):
   BODYN

# start of the program
MAINBODY
```

import modules like math

Function definitions

your "main" program

Tiramisu

```
print 'Tiramisu Recipe'
def ladyfingers():
   BODY1
ladyfingers()
def mascarpone_cream():
   BODY2
mascarpone_cream()
def espresso_syrup():
   BODYN
espresso_syrup()
```


Tiramisu

```
def ladyfingers():
   BODY1

def mascarpone_cream():
   BODY2

def espresso_syrup():
   BODYN

# start of main program
print 'Tiramisu Recipe'
ladyfingers()
mascarpone_cream()
espresso_syrup()
```

Tiramisu

```
def tiramisu():
   def ladyfingers():
    BODY1

   def mascarpone_cream():
    BODY2

   def espresso_syrup():
    BODYN

   # start of main program
   print 'Tiramisu Recipe'
   ladyfingers()
   mascarpone_cream()
   espresso_syrup()

tiramisu()
```


Tiramisu

```
def ladyfingers():
   BODY1

def mascarpone_cream():
   BODY2

def espresso_syrup():
   BODYN

def tiramisu():
   print 'Tiramisu Recipe'
   ladyfingers()
   mascarpone_cream()
   espresso_syrup()

tiramisu()
```


Power Table

Global variables

```
import math
n = 6
max_digits = int(math.floor(math.log10(n**n))) + 1

def power_table():
    for i in range(1, n+1):
        for j in range(1,n+1):
            prod = i**j
            print '%*d' % (max_digits, prod),
        print

power_table()
```

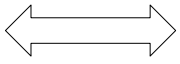
Local variable

Power Table

```
import math

def power_table(n):
    max_digits = int(math.floor(math.log10(n**n))) + 1

    for i in range(1, n+1):
        for j in range(1,n+1):
            prod = i**j
            print '%*d' % (max_digits, prod),
        print

# start of our main program
power_table(6)
```


Operator Shortcuts

```
res = 5
b = 10
res = res + 10
res = res - b
res = res * 2
res = res / 2
```



```
res = 5
b = 10
res += 10
res -= b
res *= 2
res /= 2
```


Strings and List revisited

The len function

- Returns the length of a string or list

```
drink_list = ['coke', 'sprite', 'fanta']
print len(drink_list)

for drink in drink_list:
      print drink, 'has', len(drink), 'characters'
```

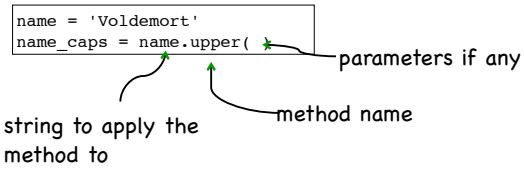

The `in` operator

- Boolean operator that takes two strings and returns True if the first is a substring of the second.

```
str1 = 'Chocolate cake is the best dessert.'
list1 = ['coke', 'sprite', 'fanta']
print 'cake' in str1
print 'truffle' in str1
print 'sprite' in list1
print '7up' in list1
```


String methods

- A method is similar to a function - it takes parameters and returns a value
- Different syntax



List methods

- `append` - add a new element to the end of the list
- `extend` - appends all of the elements in a list to the end of the list
- `insert` - inserts an element at a given index

List methods

```
1 char_names = ['Hiro', 'Sylar', 'Peter']
2 char_names.append('Nathan')
3 print char_names
4 char_names.extend(['Matt', 'Claire'])
5 print char_names
6 char_names.insert(3, 'Noah')
7 print char_names
```


Creating a new list

```
numbers = [1, 4, 8, 9, 10]

squares = []
for num in numbers:
    squares.append(num**2)
    print num, squares
```

Creating a new list

```
numbers = [1, 4, 8, 9, 10]

squares = []
for num in numbers:
    squares = squares + [num**2]
    print num, squares
```


Creating a new list

```
numbers = [1, 4, 8, 9, 10]

squares = []
for num in numbers:
    squares += [num**2]
    print num, squares
```


Deleting list elements

- Lists are mutable - we can add new elements, we can change elements, we can also remove them.

```
char_names = ['Hiro', 'Sylar', 'Peter']
name = char_names.pop(1)
print name
print char_names
```


Deleting list elements

- If you don't need the removed value

```
char_names = ['Hiro', 'Sylar', 'Peter']
del char_names[2]
print char_names
```

- Deleting multiple elements

```
char_names = ['Hiro', 'Sylar', 'Peter']
del char_names[1:]
print char_names
```


Deleting list elements

- If you know the element you want to remove

```
char_names = ['Hiro', 'Sylar', 'Peter']
char_names.remove('Sylar')
print char_names
```


Splitting string into words

```
book_title = 'Handbook of Do-It-Yourself Broom Care'
words = book_title.split()
print words
diy = words[2].split('-')
print diy
```

delimiter

Joining words into a string

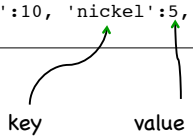
- Inverse of split

```
words = ['Handbook', 'of', 'Do-It-Yourself', 'Broom', 'Care']
diy = ['Do', 'It', 'Yourself']
delimiter = ' '
print delimiter.join(words)
delimiter = '-'
print delimiter.join(diy)
```


Dictionary

- Mapping between a set of indices (keys) to a set of values.
- The keys can be of almost any type.

```
eng2sp = {'one':'uno', 'two':'dos', 'three':'tres'}
coins = {'dime':10, 'nickel':5, 'quarter':25, 'penny':1}
empty = {}
```



Dictionary operators

- Accessing elements (lookup)

```
1 coins = {'dime':10, 'nickel':5, 'quarter':25, 'penny':1}
2 print coins
3 print coins['penny']
4 coins['penny'] = 2
5 print coins['penny']
6 print coins['dollar']
```


The `in` operator

- Returns true if something appears as a key in the dictionary

```
1 eng2sp = {'one':'uno', 'two':'dos', 'three':'tres'}
2 print 'one' in eng2sp
3 print 'uno' in eng2sp
4 print 'uno' in eng2sp.values()
```

Returns the list of values in the dictionary

Deleting elements

```
1 eng2sp = {'one':'uno', 'two':'dos', 'three':'tres'}
2 del eng2sp['one']
3 print eng2sp
4 res = eng2sp.pop['three']
5 print res
6 print eng2sp
```


Example

```
1  def histogram(s):
2      d = {}
3      for c in s:
4          if c not in d:
5              d[c] = 1
6          else:
7              d[c] += 1
8      return d
9
10 print histogram('mississippi')
```

Today's exercises

- Review of writing functions
- Practice working with lists
- More practice with strings
- Readings for Tuesday: Sections 15, 17.1-6