Python 101

Rick @ teil

The purpose

- Help you with the homework (the Pac-Man project, which is written in python)

- I will assume you have basic programming knowledge (C/C++)
- The homework uses python 2 (,which is not fully compatible with python 3)

Outlines

- Introduction to python
  - From C++ to python
  - Things you might want to know
- Setup python
- Homework 1

The first thing I want to say

- Python is easy to learn

- Python wiki http://wiki.python.org/moin/
  - Beginners' Guide
  - Beginners' Errors
  - Documentation

Python

- Need not compile
- Fully dynamic type
- Automatic memory management
- Use indentation for scope determination

  (do not mix spaces and tabs)

C++ vs. Python

C++

```
int getMax(size_t size, int const* array){
    // Find the maximum element in array
    /*
      It is just an example.
      Python has nice build-in function
      called "max()".
   */
    int max = -1 * INT_MAX;
    for (size_t i=0; i<size; ++i) {
        if (array[i] > max)
            max = array[i];
    }
    return max;
}
```

Python

```
def getMax(array):
    # Find the maximum element in array
    """
      It is just an example.
      Python has nice build-in function
      called "max()".
   """
    max = -1* sys.maxint -1
    for element in array:
       if element > max:
          max = element
    return max
```

Boolean Operations

- True, False
- and, or, not

```
a = False
b = True
c = a and b
d = a or b
e = not a
print a
print b
print c
print d
print e
```

```
False
True
False
True
True
```

Numeric Types

- int , float, long, complex

`x // y`	**(floored) quotient of *x* and *y***
`int(x)`	*x* converted to integer
`complex(re,im)`	real part *re*, imaginary part *im*. *im* defaults to zero.
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*

- no `++x, x++, --x, x--`
- math (module)

help() → math

Sequence Types

- □ str, list, tuple, …

```
# str is string in python.  See String module ( help() →
str )
str1 = 'If you have any questions, please ask.'
str2 = "If you have any questions, please ask."
str3 = str('If you have any questions, please ask.')

# a list is like an array. ( help(list) ,  dir(list) )
list1 = []
list2 = [ 'a',  3,  list1 ]
list2.append(5)
len(list2)
list3 = range(0,10)  #return [0,1,2,…,9]


# a tuple is an immutable list
tuple1 = (2, 6)
tuple2 = (3, tuple1)
```

Sequence Types

```
# for loop
list1 = [2, 3, 4, 5]
for element in list1 :
  print element
for (index,element) in enumerate(list1):
  print 'index {0} is {1}'.format(index, element)

# if loop
if 6 in list1:
  print 'Oopsy!'
if 6 not in list1:
  print 'Correct'
```

Set Types

- *Unordered* collection of distinct *hashable* objects
- Hashable: immutable types(str, tuple, numbers)
- Not *hashable*: mutable types (lists, dictionaries)
- Not indexed
- membership testing, removing duplicates

see http://wiki.python.org/moin/TimeComplexity for time complexity

Set Types

```
# set, see help(set)
set1 = set()
set2 = {'rick', 'jacky'}
if 'john' not in set2:
  print 'John is not in the set'
set2.add('John')
if 'john' in set2:
  print 'John is in the set'
set2.remove('jacky')
```

Mapping Types — dict

- dict: A *mapping* object maps *hashable* values to arbitrary objects.

```
# dict, see help(dict)
dict1 = {'one': 1, 'two': 2}
dict1['one']
1
dict1['three']   #Raises a KeyError because 'three' is not in dict1
dict1.get('three', default) #OK, will return default
dict1['three'] = 3 #OK
if 'three' in dict1:
  print 'Correct'

for key in dict1:
  print dict1[key]

for key,value in dict1.items():
  print key,value
```

Function

```
"""
def function_name ( arg1, arg2, … ) :
 ……
  return return_var1, return_var2, …
"""
def fibonacci(n):
  # write Fibonacci series up to n
  a, b = 0, 1
  while b < n:
    print b,
    a, b = b, a+b

fibonacci(50)
1 1 2 3 5 8 13 21 34
```

Function arguments

```
def testArgument(arg1, arg2 = 20, arg3 = 30):
  print 'arg1 is ', arg1
  print 'arg2 is ', arg2
  print 'arg3 is ', arg3

testArgument(10, 40)
arg1 is 10
arg2 is 40
arg3 is 30

testArgument(10, arg3 = 40)
arg1 is 10
arg2 is 20
arg3 is 40
```

Modules

functions.py

```
def fibonacci(n):
  # write Fibonacci series up to n
  a, b = 0, 1
  while b < n:
    print b,
    a, b = b, a+b
```

caller1.py

```
import functions

functions.fibonacci(50)
```

caller2.py

```
from functions import fibonacci

fibonacci(50)
```

Modules

- Just like *#include "header.h"* in C++, python uses *import module*
- Module : a file containing Python definitions and statements (ex: functions.py)
- The module's name (as a string) is available as the value of the global variable __name__

  ```
  ex:
  functions.__name__

  'functions'
  ```

- Each module is only imported once per interpreter session (to reload a module, use `reload(modulename)`)

Executing modules as scripts

- When running `python functions.py <arguments>`, the code in the module '`functions`' will be executed, just as if you imported it, but with the __name__ set to "__main__".

functions.py

```
def fibonacci(n):
  # write Fibonacci series up to n
  a, b = 0, 1
  while b < n:
    print b,
    a, b = b, a+b

if __name__ == "__main__":
   import sys
   fibonacci(int(sys.argv[1]))
```

command line

```
$ python functions.py 50
1 1 2 3 5 8 13 21 34
```

Class

```
class ClassName (ParentClass) :
 def __init__ (self, data2):
  self.dataMember1 = 'data1'
  self.dataMember2 = data2
 def printDataMembers(self):
  print 'dataMember1 is ', self.dataMember1
  print 'dataMember2 is ', self.dataMember2
 def fun(self, arg1):
  self.printDataMember()
  print 'arg1 is ', arg1

data2 = 'argument2'
object1 = ClassName(data2)
object1.printDataMembers()
object1.fun('hello')
```

Class

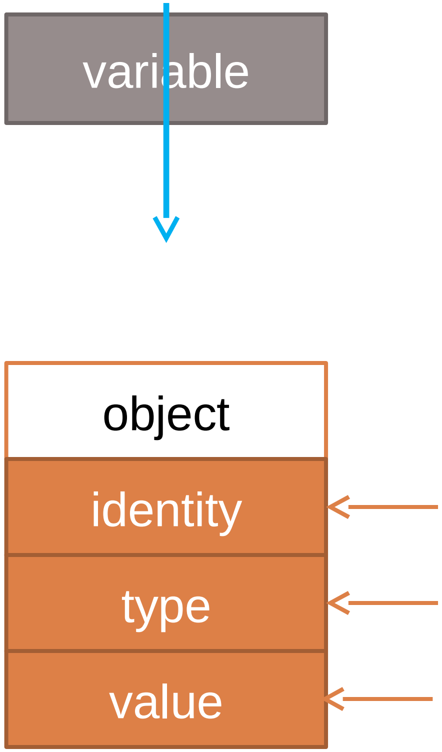
- Private members
  - does not exist in python

```
class ClassName:
  def __init__ (self):
    self._pri1 = 10
    self.__pri2 = 20
    self.__pri3__ = 30

object1 = ClassName()
object1._pri1
10
object1.__pri2
AttributeError: ClassName instance has no attribute '__pri2'
object1._ClassName__pri2
20
object1.__pri3__
30
```

Variable & Object

- Be careful. All variables are references!



variable

object

identity

type

value

Memory address. Use `id()` to check, `is` to compare

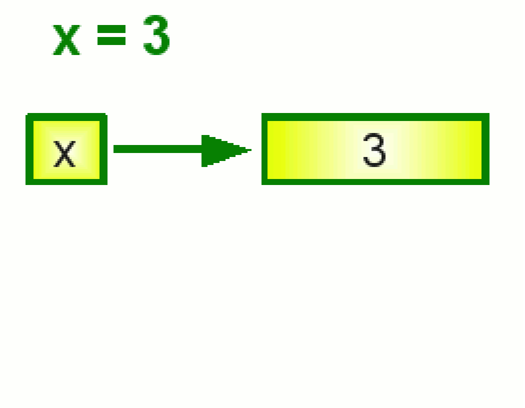
Determine supported operations. Use `type()` to check

Immutable: numbers, strings, tuples

Mutable: dictionaries, lists, sets

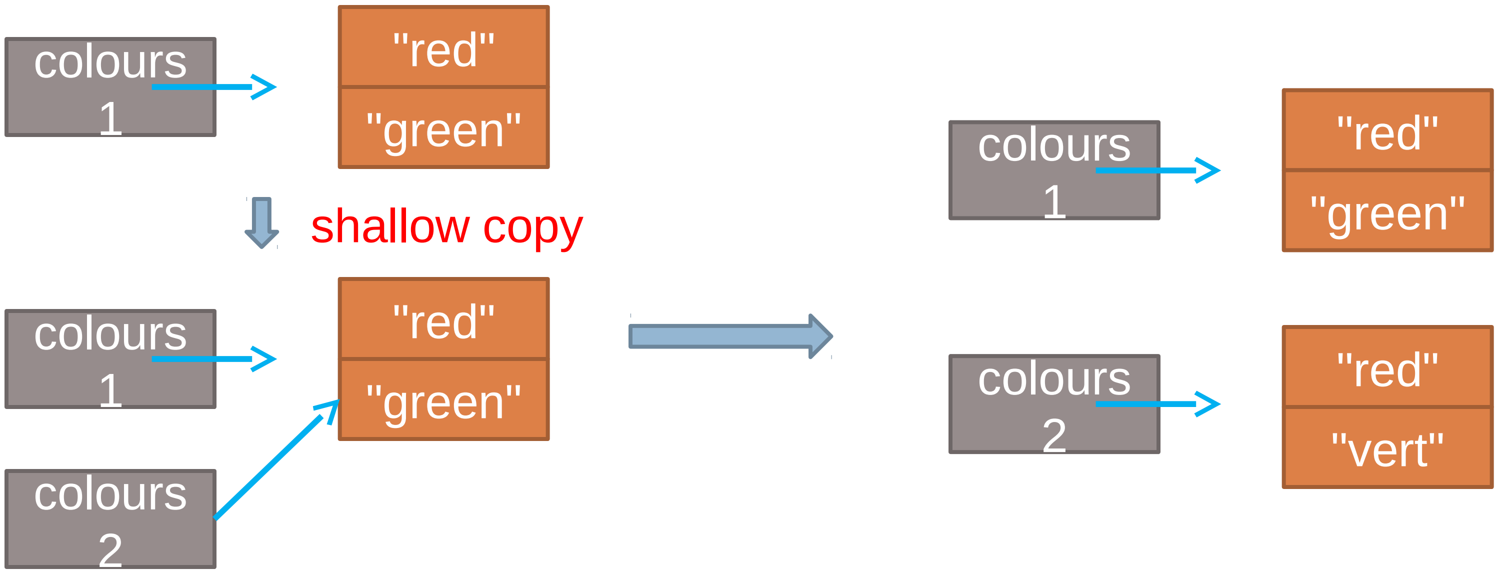
Variable & Object

- Be careful. All variables are references!
- For immutable types, operations that compute new values may actually return a reference to any existing object with the same type and value, while for mutable objects this is not allowed



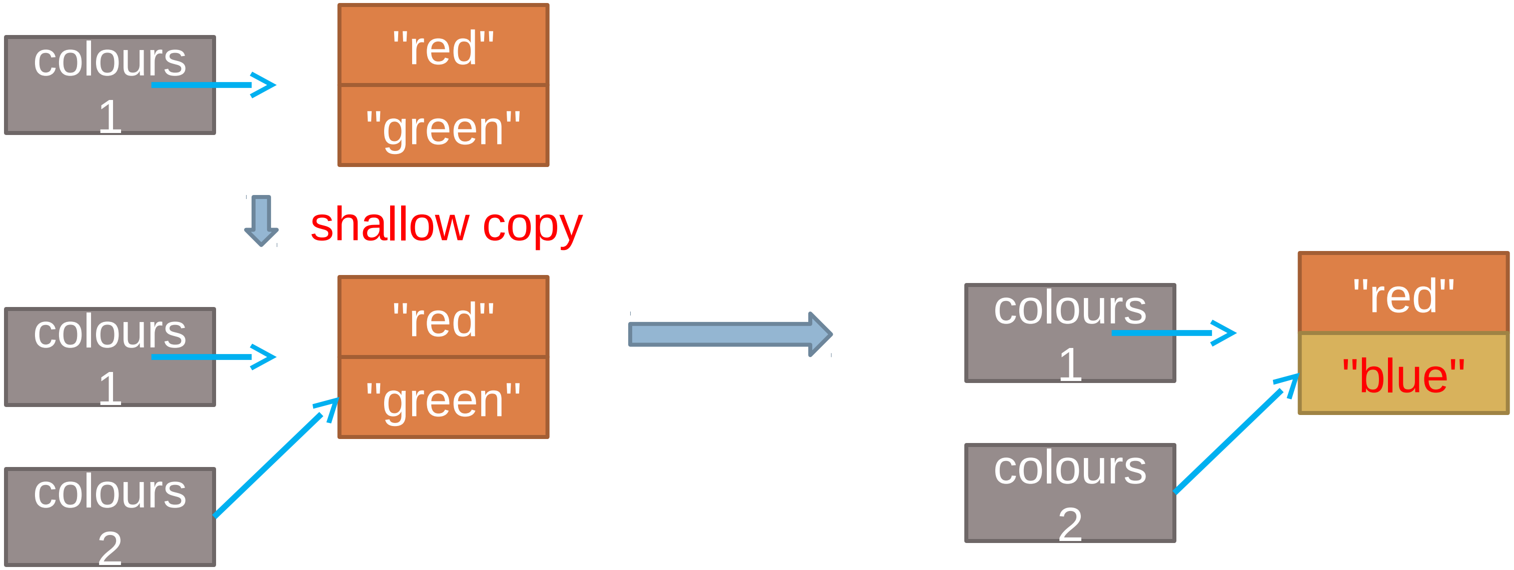
Variable & Object

```
colours1 = ["red", "green"]
colours2 = colours1  # shallow copy
colours2 = ["rouge", "vert"]  # create a new object
print colours1
['red', 'green']
```



Variable & Object

```
colours1 = ["red", "green"]
colours2 = colours1  # shallow copy
colours2[1] = "blue"
colours1
['red', 'blue']
```



Deep Copy

- Module "copy" method "deepcopy"

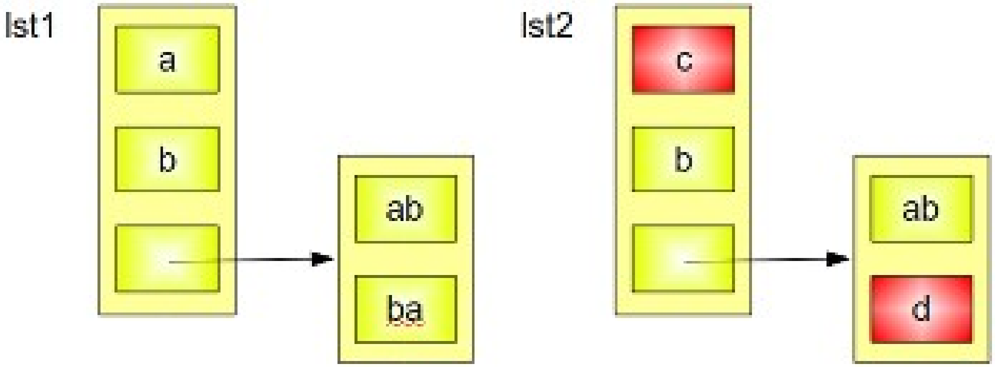
```
from copy import deepcopy

lst1 = ['a','b',['ab','ba']]
lst2 = deepcopy(lst1)

lst2[2][1] = "d"
lst2[0] = "c";



print lst2
['c', 'b', ['ab', 'd']]
print lst1
['a', 'b', ['ab', 'ba']]
```



Deep Copy

- For more information, check http://docs.python.org/reference/datamodel.html http://docs.python.org/library/copy.html http://www.python-course.eu/deep_copy.php

Setup python

Python

- Please use python 2 (2.7.2 is recommended)
- For BSD, Linux and Mac OS X,
  - python2 might already been installed
  - try python (python2.x) in your command line
  - Might need to install python-tk
    http://tkinter.unpythonic.net/wiki/How_to_install_Tkinter
- For windows,
  - Download python → install (with tcl/tk selected)
  - Add Environment Variable (ex: C://Python27)
    (My Computer ‣ Properties ‣ Advanced ‣ Environment Variables)
  - http://docs.python.org/using/windows.html

Homework 1 -- Search

Due: 2012. 03. 26

Homework 1 -- Search

- Download the package from Ceiba
- See search.html
- In start-up menu of Windows, type *cmd* and open cmd.exe.
- Change the directory to the folder containing the homework
- Try *python pacman.py*(or *python2 pacman.py*)