

ERRORS IN PYTHON

MS. RASHMI CHAUDHARY

AP

CSE DEPARTMENT

Python semantics

- Each statement has its own semantics, the `def` statement doesn't get executed immediately like other statements
- Python uses duck typing, or latent typing
 - Allows for polymorphism without inheritance
 - This means you can just declare
“`somevariable = 69`” don't actually have to declare a type
 - `print “somevariable = “ + tostring(somevariable)”`
strong typing , can't do operations on objects not defined without explicitly asking the operation to be done

Python Syntax

- Python uses indentation and/or whitespace to delimit statement blocks rather than keywords or braces
- `if __name__ == "__main__":`
 `print "Salve Mundo"`
if no comma (,) at end '\n' is auto-included

CONDITIONALS

- `if (i == 1): do_something1()`
`elif (i == 2): do_something2()`
`elif (i == 3): do_something3()`
`else: do_something4()`

Conditionals Cont.

- **if (value is not None) and (value == 1):**
print "value equals 1",
print " more can come in this block"
- **if (list1 <= list2) and (not age < 80):**
print "1 = 1, 2 = 2, but 3 <= 7 so its True"
- **if (job == "millionaire") or (state != "dead"):**
print "a suitable husband found"
else:
print "not suitable"
- **if ok: print "ok"**

Loops/Iterations

- sentence = ['Marry','had','a','little','lamb']
 for word in sentence:
 print word, len(word)
- for i in range(10):
 print i
 for i in xrange(1000):# does not allocate all initially
 print i
- while True:
 pass
- for i in xrange(10):
 if i == 3: continue
 if i == 5: break
 print i,

Functions

- ```
def print_hello():# returns nothing
 print "hello"
```
- ```
def has_args(arg1,arg2=['e', 0]):
    num = arg1 + 4
    mylist = arg2 + ['a',7]
    return [num, mylist]
has_args(5.16,[1,'b'])# returns [9.16,[[1, 'b'],[ 'a',7]]]
```
- ```
def duplicate_n_maker(n): #lambda on the fly func.
 return lambda arg1:arg1*n
dup3 = duplicate_n_maker(3)
dup_str = dup3('go') # dup_str == 'gogogo'
```

# Exception handling

- `try:`  
    `f = open("file.txt")`  
`except IOError:`  
        `print "Could not open"`  
`else:`  
    `f.close()`
  - `a = [1,2,3]`  
`try:`  
    `a[7] = 0`  
`except (IndexError,TypeError):`  
        `print "IndexError caught"`  
`except Exception, e:`  
    `print "Exception: ", e`  
`except: # catch everything`
- print "Unexpected:"  
print sys.exc\_info()[0]  
raise # re-throw caught exception
- `try:`  
    `a[7] = 0`  
`finally:`  
    `print "Will run regardless"`
- Easily make your own exceptions:  
`class myException(except)`  
    `def __init__(self,msg):`  
        `self.msg = msg`  
`def __str__(self):`  
        `return repr(self.msg)`

# Classes

```
class MyVector: """A simple vector class."""

num_created = 0

def __init__(self,x=0,y=0):
 self.__x = x
 self.__y = y
 MyVector.num_created += 1

def get_size(self):
 return self.__x+self.__y

@staticmethod
def get_num_created
 return MyVector.num_created
```

```
#USAGE OF CLASS MyVector
print MyVector.num_created
v = MyVector()
w = MyVector(0.23,0.98)
print w.get_size()
bool = isinstance(v, MyVector)
```

Output:  
0  
1.21

# I/O

```
import os
print os.getcwd() #get ""
os.chdir('..')
import glob #file globbing
lst = glob.glob('*.*txt') # get list of files
import shutil #mngmt tasks
shutil.copyfile('a.py','a.bak')

import pickle #serialization logic
ages = {"ron":18,"ted":21}
pickle.dump(ages,fout)
serialize the map into a writable file
ages = pickle.load(fin)
deserialize map from areadable file
```

```
read binary records from a file
from struct import *
fin = None
try:
 fin = open("input.bin","rb")
 s = f.read(8)#easy to read in
 while (len(s) == 8):
 x,y,z = unpack(">HH<L", s)
 print "Read record: " \
 "%04x %04x %08x"%(x,y,z)
 s = f.read(8)
except IOError:
 pass
if fin: fin.close()
```

# Threading in Python

```
import threading

theVar = 1

class MyThread (threading.Thread):

 def run (self):

 global theVar

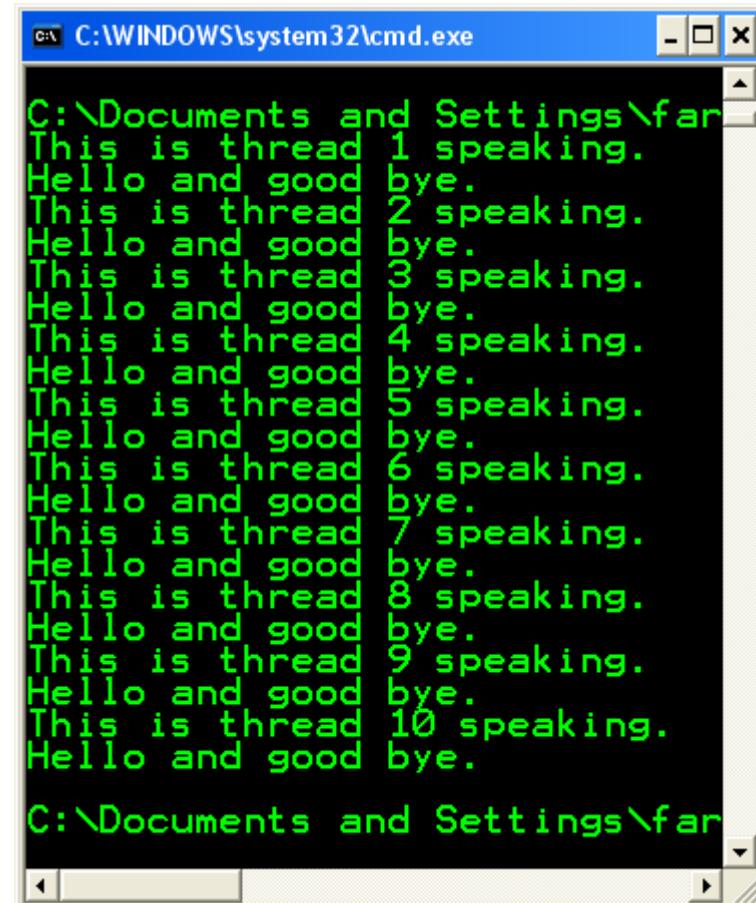
 print 'This is thread ' + \
 str (theVar) + ' speaking.'

 print 'Hello and good bye.'

 theVar = theVar + 1

for x in xrange (10):

 MyThread().start()
```

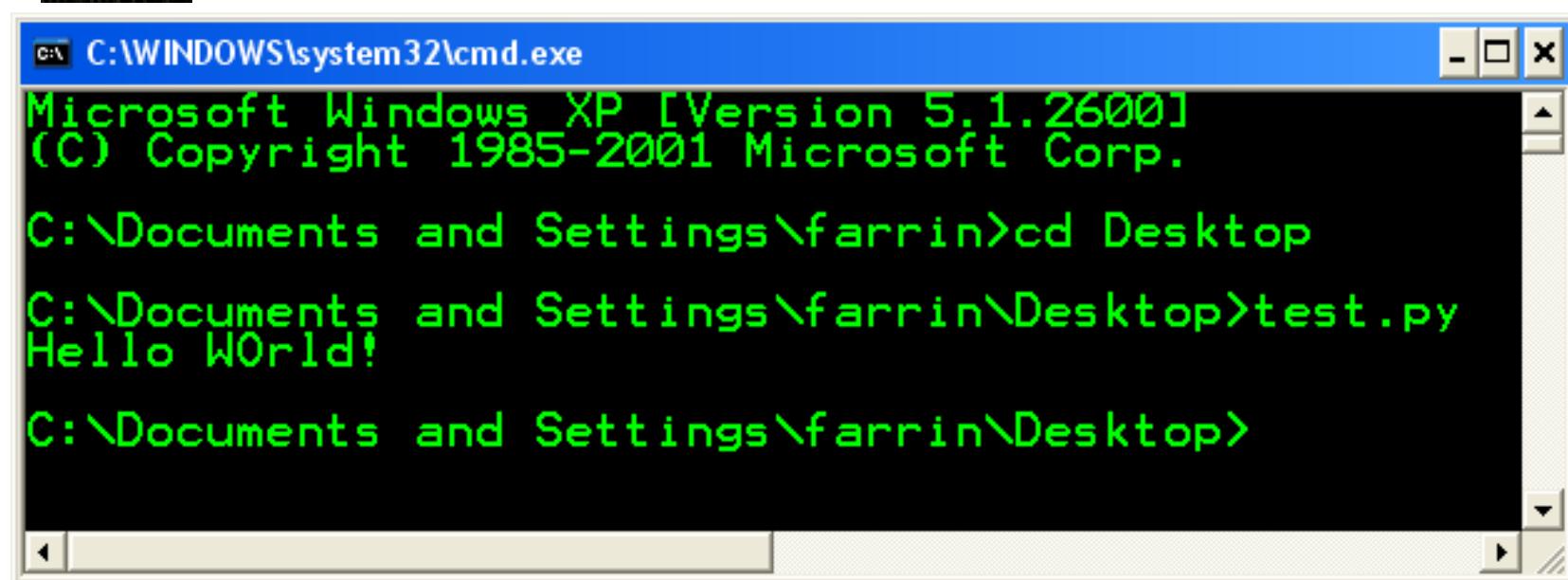


The image shows a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The window displays the output of a Python script. The output consists of ten lines, each starting with 'This is thread' followed by a number from 1 to 10, then 'speaking.', 'Hello and good bye.', and finally 'bye.' on a new line. The text is colored green.

```
C:\Documents and Settings\far
This is thread 1 speaking.
Hello and good bye.
This is thread 2 speaking.
Hello and good bye.
This is thread 3 speaking.
Hello and good bye.
This is thread 4 speaking.
Hello and good bye.
This is thread 5 speaking.
Hello and good bye.
This is thread 6 speaking.
Hello and good bye.
This is thread 7 speaking.
Hello and good bye.
This is thread 8 speaking.
Hello and good bye.
This is thread 9 speaking.
Hello and good bye.
This is thread 10 speaking.
Hello and good bye.
C:\Documents and Settings\far
```

# Running Python

- Windows XP – double click the icon or call it from the command line as such:



A screenshot of a Microsoft Windows XP Command Prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe". The window displays the following text:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\farrin>cd Desktop

C:\Documents and Settings\farrin\Desktop>test.py
Hello WOrld!

C:\Documents and Settings\farrin\Desktop>
```

# Python Interpreter

Python (command line)

```
Python 2.5.2 (r252:60911, Feb 21 2008, 13:11:45) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'hello world'
hello world
>>> x = 'roses'
>>> y = 12
>>> Roy_G_Biv = [('red', 'orange', 'yellow'), 'green', ('blue', ['indigo', 'violet'])]
>>> MyAwesomeVar = [x, y, Roy_G_Biv]
>>> print MyAwesomeVar
['roses', 12, [('red', 'orange', 'yellow'), 'green', ('blue', ['indigo', 'violet'])]]
>>> print MyAwesomeVar[0] + ' are ' + MyAwesomeVar[2:3][0][0][0]
roses are red
>>> print MyAwesomeVar[2:3][0][2][1][1] + 's are ' + MyAwesomeVar[2:3][0][2][0]
violets are blue
>>> print str(MyAwesomeVar[1]) + ' ' + MyAwesomeVar[0] + ' for my love.'
12 roses for my love.
>>> dict = {'place': 'mantle', 'where': 'above', 'myLove': True}
>>> if(dict["myLove"]): print 'the ' + dict["place"] + ' I put them ' + dict["where"]
...
the mantle I put them above
>>>
```