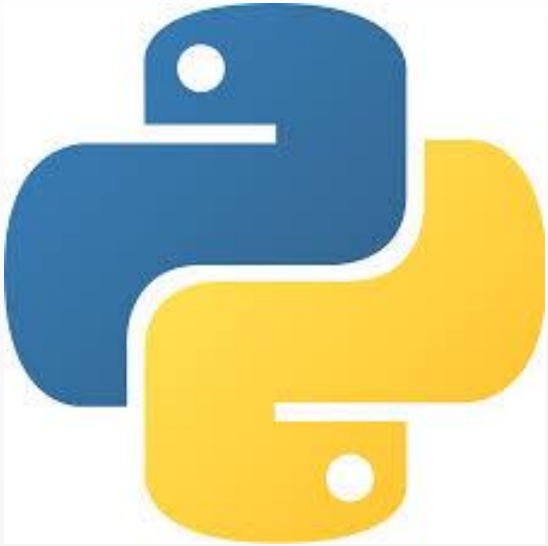

Hello Spam!

PYTHON BASICS - SLIGHTLY ADVANCED

Jui Pann

Eggs

- Python Sets & String Formats
- More about Modules
- Some useful toys
- More about Functions
- More about Classes
- Script Appreciation & Exercise
- Remarks

Python Sets

- >>> x = set('abcde')
- >>> y = set('wonanguo')
- >>> x & y
- >>> 'o' in y
- >>> nederlands = set(['Kuyt', 'Robben', 'Blind'])
- >>> fcbayern = set(['Neuer', 'Dante', 'Robben'])
- >>> nederlands | fcbayern

String Formats

- >>> excla = 'their'
- >>> 'All %s fault!' % excla
- >>> '%d %s you, pray %d you' % (1, 'love', 4)

mark	type
%s	string
%c	character
%d	decimal
%i	integer
%u	unsigned
%o	octal
%x ; %X	hexadecimal
%e ; %E	scientific expression
%f	float

Eggs

- Python Sets & String Formats
- More about Modules
- Some useful toys
- More about Functions
- More about Classes
- Script Appreciation & Exercise
- Remarks

Module **string** - Constants

- >>> dir(string)
- >>> import string
- >>> string.digits
- >>> string.ascii_lowercase
- >>> string.hexdigits
- >>> string.whitespace

- >>> ord('a')
- >>> chr(113)

Module **string** - Functions

- >>> x = 'spam'
- >>> x.capitalize()
- >>> x = x * 2
- >>> x.find('a')
- >>> x.rfind('a')
- >>> x.split('a')
- >>> help(x.rindex)

Module **time**

- >>> import time
- >>> time.time()
  - Epoch(or UNIX) time
  - The number of seconds that have elapsed since 00:00:00 UTC, Tuesday, 1st Jan, 1970.
- >>> time.daylight
- >>> time.timezone
- >>> zeit = time.localtime()
- >>> zeit[3:6]

Module **random**

- Random number generator of different distributions with different attributes.
- >>> import random
- >>> random.random()
- >>> random.randint()
- >>> random.gammavariate(1,2)
- >>> random.gauss(0,1)
- >>> x = range(1,53)
- >>> random.shuffle(x)

_foo & __all__ for Mysophobia

- >>> from moduadv1 import *
- >>> dir()

```
 1
 2 __all__ = ['fa', 'fc']
 3
 4 def fa():
 5     pass
 6
 7 def fb():
 8     pass
 9
10 def fc():
11     pass
12
13 def fd():
14     pass
15
```

```
 1
 2 # __all__ = ['fa', 'fc']
 3
 4 def _fa():
 5     pass
 6
 7 def _fb():
 8     pass
 9
10 def fc():
11     pass
12
13 def fd():
14     pass
15
```

Eggs

- Python Sets & String Formats
- More about Modules
- Some useful toys
- More about Functions
- More about Classes
- Script Appreciation & Exercise
- Remarks

zip

- Used to combine lists or tuples into a list of tuples, frequently seen in loops.

- >>> x = [(1,6), (2,7), (3,8)]
- >>> for a, b in x:

  ... print a, '+', b, '=', a+b
- >>> zip(range(1,4), range(6,9))
- >>> zip(range(6),range(6,9),range(7,3,-1))

enumerate

- Used to generate tuples of offset as well as its corresponding value.

- >>> s = 'spam!'
- >>> en = enumerate(s)
- >>> en.next()
- >>> for item, offs in enumerate(s):
- ... print item, 'appears at offset', offs

filter

- Use Boolean value to filter a list.
- Usage: filter(<boolean>, <list>)

- >>> filter((lambda a: a> 0), range(-5,5))
- >>> filter((lambda a: a%3==0), range(-10,10))

List Comprehension

- To construct a list with an expression instead of a statement. (Simple is better than complex)

- >>> L = range(1,6)
- >>> for i in range(len(L)):
  ... L[i] += 10
- >>> L

- >>> L = [x + 10 for x in L]

List Comprehension

- >>> [x + y for x in ['f7', 'seabox', 'hateOnas', 'boyo'] for y in ['5566', '7788', 'ininder']]
- >>> [ord(x) for x in 'spam']
- >>> [x**2 for x in range(10)]
- >>> map((lambda x: x**2), range(10))
- >>> [x**2 for x in range(10) if x%2 == 0]
- >>> [(x, y) for x in range(5) if x % 2 == 0 for y in range(5) if y % 2 == 1]

Matrix by List

- Quite similar to that in Matlab, although the commas cannot be omitted.
- >>> M = [[1,2,3],[4,5,6],[7,8,9]]
- >>> N = [[2,2,2],[3,3,3],[4,4,4]]
- >>> M[1], M[1][2]
- >>> [row[1] for row in M]
- >>> [M[i][i] for i in range(len(M))]
- >>> [M[row][col] * N[row][col] for row in range(3) for col in range(3)]

Eggs

- Python Sets & String Formats
- More about Modules
- Some useful toys
- More about Functions
- More about Classes
- Script Appreciation & Exercise
- Remarks

`global` statement

lambda

- The anonymous function constructor lambda is an expression rather than a statement.
- It expresses the result of execution, instead of returning it to the caller.

- Usage: lambda arg1, arg2, … : expression
- >>> f = lambda x, y, z: x + y + z
- >>> f(2,3,4)

lambda

- >>> L = [(lambda x: x**2), (lambda x: x*3), (lambda x: x%4), (lambda x: x+5)]
- >>> for f in L:

... print f(2)

- >>> print L[0](3)
- >>> lower = (lambda x, y: x if x < y else y)
- >>> lower(3,5)

try… except… in statements

```
 1
 2 while True:
 3     reply = raw_input('enter: ')
 4     if reply == 'stop':
 5         break
 6     elif not reply.isdigit():
 7         print reply.upper()
 8     else:
 9         print int(reply) ** 2
10 print 'au revoir!'
11
```

```
 1
 2 while True:
 3     reply = raw_input('enter: ')
 4     if reply == 'stop':
 5         break
 6     try:
 7         print int(reply) ** 2
 8     except:
 9         print reply.upper()
10 print 'au revoir!'
11
```

Arguments Passing

- >>> s1,s2,s3='spam', 'scam', 'slam'
- >>> intersect(s1,s2), union(s1,s2)
- >>> intersect([1,2,3], (1,4,5))
- >>> intersect(s1,s2,s3)

```
 1
 2 def intersect(*args):
 3     res = []
 4     for x in args[0]:
 5         for anders in args[1:]:
 6             if x not in anders:
 7                 break
 8             else:
 9                 res.append(x)
10     return res
```

```
11
12 def union(*args):
13     res = []
14     for seq in args:
15         for x in seq:
16             if not x in res:
17                 res.append(x)
18     return res
19
```

yield

- **return** returns return value and ends function.
- yield generates values without immediately ceasing the function (aka. generator function)
- >>> def reve(stri):

... for i in range(len(stri)-1, -1, -1):

... yield stri[i]

- >>> x = reve('reverse me, bitte')
- >>> next(x)

Eggs

- Python Sets & String Formats
- More about Modules
- Some useful toys
- More about Functions
- More about Classes
- Script Appreciation & Exercise
- Remarks

Operator Overloading

- Some operator performs differently in some class if specifically defined in the class.

```
1
2  class firstcla:
3
4      def __init__(self, value='data unset!'):
5          self.data = value
6          self.x = 2048
7      def setData(self, value):
8          self.data = value
9      def display(self):
10         print self.data
11
12     a = 1
13     b = 2
14
15
```

```
16 class secondcla(firstcla):
17
18     def display(self):
19         print 'current value = \'%s\'' % self.data
20     def dispsh(self,z):
21         print '*' * z
22
23     b = 3
24     c = 4
25
```

```
26 class thirdcla(secondcla):
27
28     def __init__(self, value):
29         self.data = value
30     def __add__(self, other):
31         return thirdcla(self.data + other)
32     def __mul__(self, other):
33         self.data = self.data * other
```

Operator Overloading

- >>> x = thirdcla('abc')
- >>> x.display()
- >>> y = x + 'xyz'
- >>> y.display()
- >>> x * 3
- >>> x.display()
- >>> x.a, x.b, x.c

Eggs

- Python Sets & String Formats
- More about Modules
- Some useful toys
- More about Functions
- More about Classes
- Script Appreciation & Exercise
- Remarks

Port Scanner

```
 1
 2 import socket
 3 import sys
 4 from time import time
 5
 6 remoteServer    = raw_input("Enter a remote host to scan: ")
 7 remoteServerIP  = socket.gethostbyname(remoteServer)
 8
 9 print "-" * 60
10 print 'Requested host: ', remoteServer
11 print "Please wait, scanning remote host", remoteServerIP
12 print "-" * 60
13
14 t1 = time()
15
16 try:
17     for port in range(1,65535):
18         sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
19         result = sock.connect_ex((remoteServerIP, port))
20         if result == 0:
21             print "Port {}: \t Open".format(port)
22         sock.close()
```

Port Scanner

```
23
24 except KeyboardInterrupt:
25     print "You pressed Ctrl+C"
26     sys.exit()
27
28 except socket.gaierror:
29     print 'Hostname could not be resolved. Exiting'
30     sys.exit()
31
32 except socket.error:
33     print "Couldn't connect to server"
34     sys.exit()
35
36 t2 = time()
37 telap = t2 - t1
38
39 print 'Scanning Completed in: ', telap, 'seconds'
40
```

Script Appreciation & Exercise

Eggs

- Python Sets & String Formats
- More about Modules
- Some useful toys
- More about Functions
- More about Classes
- Script Appreciation & Exercise
- Remarks

Remarks

- Useful modules:
  - http://www.catswhocode.com/blog/python-50-modules-for-all-needs
- Python debugger in Unix:
  - bugjar