Python Training

A basic overview

Functions, Modules & Packages

Functions

- Built-in functions
- Lambda functions

Modules

- What are modules?
- Import statements

Packages

Libraries

- Math
- Numpy

Introduction to Functions

- The syntax for function definition is

```
    def NAME( PARAMETERS ):
        """Docstring"""
         STATEMENTS
      [return]
```

- A function must be defined before its first use
- The return statement is used to return a value from function.
- If a function does not have return statement , it is considered as a Procedure
- If a function has to return multiple values , tuples are preferred
- Sample function call

```
  name = my_func(arg1, arg2, arg=‘Default’)
```

Functions…cont.

Call by value for primitive data types

- Call by reference for derived data types
  - Q: Why?
  - A: Reference Semantics

Functions: Parameter passing

`def hello(greeting='Hello', name='world'):`
`    print ('%s, %s!' % (greeting, name))`
 	
`hello('Greetings')`
	Adding default values to parameters

`def hello_1(greeting, name):`
`    print ('%s, %s!' % (greeting, name))`
`# The order here doesn't matter at all:`
`hello_1(name='world', greeting='Hello')`
	Using named parameters. In this case the order of the arguments does not matter.

`def print_params(*params):`
`    print (params)`
 	
`print_params('Testing')`
`print_params(1, 2, 3)`
	The variable length function parameters allow us to create a function which can accept any number of parameters.

`def print_params_3(**params):`
`    print (params)`
 	
`print_params_3(x=1, y=2, z=3)`
	Variable named parameters

`def print_params_4(x, y, z=3, *pospar, **keypar):`
`    print (x, y, z)`
`    print (pospar)`
`    print (keypar)`
 	
`print_params_4(1, 2, 3, 5, 6, 7, foo=1, bar=2)`
`print_params_4(1, 2)`
	A combination of all of above cases

Built-in functions

abs()	divmod()	***input()***	open()	staticmethod()
all()	***enumerate()***	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	***filter()***	len()	***range()***	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	***reduce()***	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	***map()***	repr()	xrange()
cmp()	globals()	max()	reversed()	***zip()***
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

Lambda functions

- Unnamed functions
- Mechanism to handle function objects
- To write inline simple functions
- Generally used along with maps, filters on lists, sets etc.
- Not as powerful as in C++11, Haskell etc. e.g. no looping etc.
- Example: lambda x,y : x+y to add two values

Modules

- A module is a file containing Python definitions and statements intended for use in other Python programs.
- It is just like any other python program file with extension .py
- Use the “import <module>” statement to make the definitions in <module> available for use in current program.
- A new file appears in this case \path\<module>.pyc. The file with the .pyc extension is a compiled Python file for fast loading.
- Python will look for modules in its system path. So either put the modules in the right place or tell python where to look!

```
import sys
sys.path.append('c:/python')
```

Modules

- Three import statement variants

`import math`
`x = math.sqrt(10)`
 	
`import math as m`
`print  m.pi`
	Here just the single identifier math is added to the current namespace. If you want to access one of the functions in the module, you need to use the dot notation to get to it.

`from math import cos, sin, sqrt`
`x = sqrt(10)`
	The names are added directly to the current namespace, and can be used without qualification.

`from math import *`
`x = sqrt(10)`
	This will import all the identifiers from module into the current namespace, and can be used without qualification.

Packages

- Packages are used to organize modules. While a module is stored in a file with the file name extension .py, a package is a directory.

- To make Python treat it as a package, the folder must contain a file (module) named __init__.py

File/Directory	Description
~/python/	Directory in PYTHONPATH
~/python/drawing/	Package directory (drawing package)
~/python/drawing/__init__.py	Package code ("drawing module")
~/python/drawing/colors.py	colors module
~/python/drawing/shapes.py	shapes module
~/python/drawing/gradient.py	gradient module
~/python/drawing/text.py	text module
~/python/drawing/image.py	image module

Classes & Objects

- Python is an object-oriented programming language, which means that it provides features that support object-oriented programming (OOP).

- Sample class definition

```
class Point:
        """ Point class represents and manipulates x,y coords. """
        def __init__(self):
            """ Create a new point at the origin """
            self.x = 0
            self.y = 0
p = Point()
print p.x, p.y
```

- Constructor: In Python we use __init__ as the constructor name

```
        def __init__(self):                  # a = Point()
        def __init__(self, x=0, y=0):        # a = Point(5, 6)
```

Classes & Objects

- Methods

```
class Point:
    """ Point class represents and manipulates x,y coords. """
    def __init__(self, x=0): self.x = x
    def x_square(self): return self.x ** 2

p = Point(2)
print p.x_square()
```

- Objects are mutable.

Working with Files

- Python supports both free form and fixed form files – text and binary
- open() returns a file object, and is most commonly used with two arguments: open(filename, mode)
- Modes:

Value	Description
'r'	Read mode
'w'	Write mode
'a'	Append mode
'b'	Binary mode (added to other mode)
'+'	Read/write mode (added to other mode)

- f = open(r'C:\text\somefile.txt')
- For Input/Output: read(), readline(), write() and writeline()

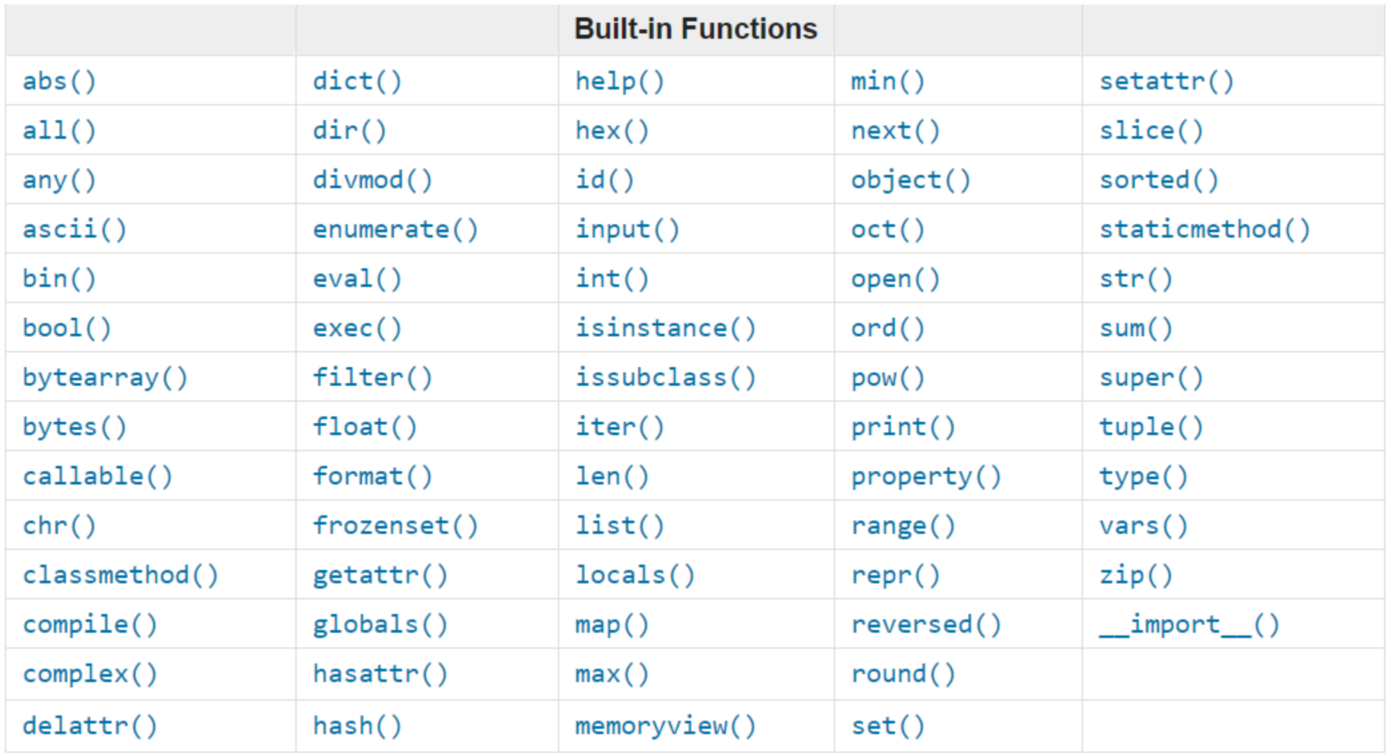
Working with Files

- File Object attributes

Attribute	Description
file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.
file.softspace	Returns false if space explicitly required with print, true otherwise.

Libraries

- The “Python library” contains several different kinds of components.
- Libraries contain built-in function and exceptions.
- The bulk of library consist of collection of modules.



		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Math Librarie

- It provides access to the mathematical functions defined by the C standard
- The following some of the functions provided by this module
  - Number-theoretic and representation functions
    - ceil(x):
    - copysign(x,y)
    - fcactorial(x)
  - Power and logarithmic functions
    - exp(x)
    - pow(x,y)
    - sqrt(x)

Demo of Math Library

Numpy Library

- Numpy stands for Numerical Python
- NumPy is the fundamental package for scientific computing
- It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

Demo of Numpy Library