THE PYTHON CHEATSHEET

STRUCTURES

STRING

s=’’ or s=””

Action	Method	Comments
replace	string.replace(s,’search’, ‘replace’)	
split	s.split(s,’sep’)	
find	string.find(s,’search’)	requires ‘**import** string’ Index of the first occurrence
count	string.count(s, ‘search’)	Number of occurrences
find (regexp)	[m.start() for m in re.finditer('regexp', s)] [m.start() for m in re.finditer('(?=regexp)', s)]	requires ‘**import** re’ for overlapping occurrences
upper/lower	s.upper()/s.lower()	returns the string in upper/lowercase

LIST

a=[]

Action	Method	Comments
access	a[i]	
slice	a[i:j]	
length	len(a)	
remove	del a[i]	
add	f.append(v)	
sort	f.sort or sorted(f)	more here: https://wiki.python.org/moin/HowTo/Sorting
merge	‘glue’.join(a)	returns ‘a[0]gluea[1]gluea[2]…’
deep copy	a2=copy.deepcopy(f)	requires ‘import copy’
pop	a.pop()	returns and removes the last element of the list
range	range([s],e)	returns [s,s+1,s+2,…, e-1]
	range(e,s,-1)	returns [s-1,s-2,…,e+1,e]
xrange	as in range	returns an iterator instead (better for loops with >10^6 iterations)
unique	list(set(a))	
difference	list(set(a)-set(b))	returns elements in a that are not in b
index	a.index(v)	returns the position of the first occurence of v in a

DICTIONARY

d={}

Action	Method	Comments
keys	d.keys()	
values	d.values()	
access	d[k]	
set	d[k]=v	

COMMENTS

```
''' single line comment
# single line comment too
''' multiple
line comment '''
```

I/O

PRINT

```
print v     #can be a single value or any structure (e.g. string, list, dictionary)
```

FORMAT

```
'{0} any text {1} any text {2} …'.format(v0,v1,v2…)
      #returns a string formed by the values of the variables instead of  {n}
```

FILE

```
f=open(path, 'access')#access is usually 'r' or 'w'
```

Action	**Method**	**Comments**
read	f.readlines()	returns an array of strings
write	f.write(string)	use '\n' for newline
save	f.close()	

CONTROL

LOOP

```
for index in list:
      do_lines                #indentation marks what's inside the loop

one-line form: [do_line for index in list]     #results are returned in a new list
      #this is equivalent to a flexible map (see http://www.bogotobogo.com/python/python_fncs_map_filter_reduce.php)

while(condition):
      do_lines
```

METHOD

```
def method(arguments):
      method_lines
      return value            #optional

      yield: returns at this point, but for the next call to the method, it will resume from this point
              (see http://www.prasannatech.net/2009/07/introduction-python-generators.html)
```

STATISTICS

import numpy as np

Action	**Method**	**Comments**
mean	np.mean(a)	a is a list of numbers. nanmean to ignore NaNs
standard dev.	np.std(a)	nanstd to ignore NaNs
min/max	np.amin(a) / np.amax(a)	nanmin/nanmax to ignore NaNs
percentile	np.percentile(a,g)	computes the qth percentile more at: http://docs.scipy.org/doc/numpy/reference/routines.statistics.html
floor/ceil	np.floor(x)/np.ceil(x)	nearest above/below integer
round	np.fix(a[,decimals])	rounds array to the nearest integer (or given number of decimals)
sum/prod	np.sum(a)/np.prod(a)	sum/prod of all the elements in the array more at: http://docs.scipy.org/doc/numpy/reference/routines.math.html

NUMPY.ARRAY

import numpy as np

matrix	m=np.array([[1,2,3],[4,5,6]])	more at: http://docs.scipy.org/doc/numpy/reference/arrays.ndarray.html
dimension	m.shape()	(2,3)
access	m[1,2]	element at **second** row, **third** column
slicing	m[:,1]	whole first column as an array
append	m=np.append(m,[34])	appends at the end of matrix
table	t=np.empty(#rows,dtype=[(“name”,”type”),…]	**dtype** is a list of as many pairs as columns. Each pair contains the **name** of the column and the **type** (a-character, f-float, i-integer) and **size** (in bytes) of data in it: np.empty([53,dtype=[("pos", "i4"),("text", "a10”)])
init	t=np.zeroes(#rows,dtype=[(“name”,”type”),…]	As empty, but fills each element in the table with zeroes
access	t[pos] for rows; t[“name”] for cols	
sort	t=np.sort(t,order=(”name”…))	Sorts t rows by column “name” (additional columns can be set)
search	np.where(t[“name”]==”pattern”)	
	np.where(m>5)	
	np.seachsorted(t[“name”], “pattern”)	Search on a sorted column (faster than where)

numpy.array is a direct C array wrapper, and is recommended with long (>10^6 elements) arrays for better memory usage

TIME

```
import time
t0=time.clock()
operation_lines
print ‘it took {0}s to make the operation’.format(time.clock()-t0)
```

LAMBDA FUNCTION

http://www.secnetix.de/olli/Python/lambda_functions.hawk

Action	Method	Comments
lambda	lambda x:x+3	equivalent to def f(x): return x+3
	foo=[2, 18, 9, 22, 17, 24, 8, 12, 27]	
filter	filter(lambda x:x>=10, foo)	gets a list only with the values >10 in foo
map	map(lambda x:x*2+10, foo)	Applies lambda to the values in foo
reduce	reduce(lambda x,y:x+y,foo)	Applies lambda to the first two values in foo and then aggregates each following value