

Python Sheet (v. 4/7/14)

Language Elements

Some datatypes:

Numbers: 23, 3.14
Imaginary, complex: 1j, 0.5+4.3j
String: 'hi there'
String: "hi there"
String: '''first line
Second line'''
List: [12, 'Blah', 3+2j]
Nested lists: [12, [2, 3], 'a']
Tuple: (1, -4, 2)
Dictionary: {'one': 'pb', 2:8}
Booleans: True, False

Assignment:

```
a = 12
beef = 'a dead cow'
c = [1, 'a list of 3', "things"]
a += 3 # means a = a + 3
a *= 2 # a = a * 2
a <op>= b # a = a <op> b where
# <op> is some binary operator
```

Multiple assignment:

```
a,b = 1,8 # a=1 and b=8
a,b = b,a # now a=8 and b=1
a,b,c = 1+5,math.sqrt(3),f(17)
```

Operators:

```
2+3 # 5
'flu'+ "gle" # "flugle"
[1,8]+[3,-1] # [1,8,3,-1]
5-2 # 3
5*2 # 10
[2,5]*3 # [2,5,2,5,2,5]
3/4 # 0 (integer division)
3.0/4 # 0.75
17%5 # 2 (remainder)
7.5%5 # 2.5 (remainder)
2**3 # 8 (exponentiation)
< <= == != >= > # numeric
comparisons
and or not # logical connectives
```

Functions:

```
def <function-name>(<args,...>):
    <body>

def Fred(n, Harry):
    a = n * Harry
    return a

print Fred(2,6) # 12
print Fred(2,Fred(3,4)) # 24
```

Multiple Return:

```
def TryToDivide(a,b):
    if b != 0:
        return "Yes", a/b
    else:
        return "No", 0

msg,answer = TryToDivide(2,0)
# msg = "No", answer = 0
```

if: (decision-making)

```
if <test 1>:
    <body 1>
elif <test 2>: # optional
    <body 2>
elif <test 3>: # optional
    <body 3>
else: # optional
    <body 4>

if n == 0:
    C = 12
elif n < 0:
    C = 13
    D = 1
elif 1 <= n <= 10:
    C = 14
else:
    C = 15
```

while: (looping)

```
while <test>:
    <body>

i = 1
while i <= 5:
    print i
    i += 1
```

for: (looping)

```
for <variable> in <list or string>:
    <body>

sum=0
for t in [1, 2, 18, -2]:
    sum += t
print sum # 19
```

break: (breaking out of a loop)

```
# print the elements of a list L
i = 0
while True:
    if i >= len(L):
        break
    print L[i]
    i = i + 1
```

in: (is an element of?)

```
2 in [1, 2, 5] # True
'ab' in "fabulous" # True
'abc' in "fabulous" # False
```

Python Sheet (v. 4/7/14)

try/except: (dealing with runtime errors)

```
try:
    <risky code>
except:
    <code to handle problems>
    <that arose in the risky code>

try:
    f = open("fred.txt","r")
    s = f.read()
    f.close()
    return s, 'OK'
except:
    return '', 'Cannot open file'
```

Selected useful functions/methods

String methods: (there are many others)

```
s='abcdbcA'
len(s) # 7
s.find('c') # 2
s.find('c',3) # 5
s.count('bc') # 2
s.replace('bc','PQ-') # 'aPQ-dPQ-A'
s.isalpha() # True
'123'.isdigit() # True
'123.45'.isdigit() # False
s.endswith('cA') # True
" ab ".strip() # 'ab'
"ab c def".split() #
['ab','c','def']
'abTTcTTdef'.split('TT') #
['ab','c','def']
```

List methods: (...and others...)

```
L=[23,-2.5,1.8e9,0]
len(L) # 4
L[1]='17' # changes L to
[23,17,1.8e9,0]
L[1:3]=[] # changes L to [23,0]
L.append(12) # changes L to
[23,-2.5,1.8e9,0,12]
L.index(-2.5) # 1 (same as s.find)
L.count(1.8e9) # 1
L.sort() # will modify L to:
[-2.5,0,23,1.8e9], but return
nothing
```

Dictionary methods (...and others...)

```
d={'last':'Brooks', 'age':100,
'charm':True, 'IQ':6.02e23}
print d['age'] # 100
d['GPA'] # error (no such key)
d.has_key('GPA') # False
d.has_key('last') # True
```

d.keys() # ['last', 'age', ..., 'IQ'] Slicing Strings (works with lists, tuples)

```
s[N] # char at position N
s[A:B] # substring starting at A
and stopping before B
s[A:B:C] # substring starting at A
and stopping before B, every C
steps
```

```
Pos:0123456789
s='abcdefghij'
```

```
s[0] # 'a'
s[-1] # 'j'
s[2:5] # 'cde'
s[:3] # 'abc' (the front)
s[3:] # 'defghij' (the end)
s[:n]+s[n:] == s # for any n
s[:] == s
s[3:-2] # 'defgh'
s[1:8:2] # 'bdfh'
s[8:1:-2] # 'igec'
s[::-1] # 'jihgfedcba' (reverse)
```

```
s[4] = 'B' # ILLEGAL! Cannot assign
to part of a string, but works with
lists
```

Input/Output: (files and keyboard/screen)

```
f = open('fred.txt','r') # open for
reading
f = open('fred.txt','w') # open for
writing (over-writing)
s = f.read() # read entire file
into s
f.write('whatever\n') # write a
string ('\n' is newline char)
f.close() # close file
```

```
s = raw_input('Your name? ') #
request string from keyboard
n = input('Your age? ') #
request number
```

Libraries (modules): (importing/using)

```
import math
a = math.sqrt(5.6) # call function
with library name as prefix

from math import *
b = sqrt(5.6) # call function
without library name as prefix
```

Many (hundreds of) libraries: math, random, os, time, sys, ...