BIG DADDY'S PYTHON TOOLBOX For 3.5

 R4 **V060516B**

Reserve Words

Comparsion / Conjunction

true **==** (equal) **false** **none** (i.e., null) **and** **not** **or**
in list, tuple, string, dictionary
is true if **same** object

Definition

class create a class
def create a function
del items in lists (del mylist[2]), whole strings, whole tuples, whole dictionaries

Module Management

import connects mod, i.e, import math
from gets a function from math import cos
as creates an alias for a function

Miscellaneous

pass (placeholder – no action)
with wrapper ensures _exit_ method

Functions

def, return(obj), yield, next inside functions **yield** is like **return** except it returns a generator whose sequential results are triggered by **next**
global declares global inside a function
non local a variable inside a nested function is good in the outer function
lambda anonymous inline function with no return statement

```
a = lambda x: x*2
for i in range (1,6):
    print (a(i))
```

Error Management

raise forces a ZeroDivisionError
try except finally else return used in error handling blocks
try: code with error potential
except: do this if you get the error
else: otherwise do this code
finally: do this either way
assert condition=False raises **AssertionError**

Looping

while (some statement is true)
for example:

```
alist=['Be','my','love']
for wordnum in range(0,len(alist)):
    print(wordnum, alist[wordnum])
```

range range (1,10) iterates 123456789
break **continue**
break ends the smallest loop it is in;
continue ends current loop iteration

Decision Making

if elif else

```
def if_example(a):
if a == 1:
    print('One')
elif a == 2:
    print('Two')
else:
    print('Some other')
```

The Ternary if Statement
An inline **if** that works in formulas:

```
myval = (high if (high > low) else low) * 3
```

Multi-line Statements \
Not needed within the [], {}, or ()
Multiple Statements on a Line **;** not with statements starting blocks

Reading Keystrokes

```
text = ""
while 1:
  c = sys.stdin.read(1)
  text = text + c
  if c == '\n':
     break
print("Input: %s" % text)
```

You must import sys before you can use the standard input (sys.stdin.read) function.

Major Built-In Functions

String Handling (↳=converts / returns)

str(object) ↳string value of object
repr(object) ↳ printable string
ascii(str) ↳ printable string
eval(expresion) ↳ value after evaluation
chr(i) ↳ character of Unicode [chr(97) = 'a']
input(prompt) ↳ user input
len(—) ↳ length of str, items in list/dict/tuple
ord(str) ↳ value of Unicode character
slice(stop) or **slice(start, stop [,step])**
↳a slice object specified by slice (start, stop, and step) word = "Python"; word[2:5]='thon'
format(value [,format_spec]) ↳ value in a formatted string—**extensive and complex** - 2 examples (comma separator & % to 3 places)
print('{:,}'.format(1234567890)) yields '1,234,567,890'
print('{:.3%}'.format(11.23456789)) yields '1123.457%'

Number Handling

abs(x) ↳ absolute value of x
bin(x) ↳ integer to binary bin(5)='0b101' (one 4, no 2's, one 1)]
divmod(x,y) takes two (non complex) numbers as arguments, ↳a pair of numbers - quotient and remainder using integer division.
float(x) ↳ a floating point number from a number or string
hex(x) ↳ an integer to a hexadecimal string hex(65536) = ox10000
int(x) ↳ an integer from a number or string
pow(x,y [,z]) ↳ x to y, if z is present returns x to y, modulo z
round(number [,digits]) ↳ floating point number rounded to digits; Without digits it re-turns the nearest integer.

Miscellaneous Functions

bool(x) ↳ true/false, ↳ false if x is omitted
callable(object) ↳true if object callable
help(object) invokes built-in help system, (for interactive use)
id(object) ↳unique object integer identifier
print(*objects, sep=' , end='\n', file= sys.stdout, flush=False) prints objects separ -ated by sep, followed by end; **%** *see other side*

Data Container Functions type=list/tuple/dict

all(iterable) ↳ TRUE if all elements are true or it is empty
any(iterable) ↳ TRUE if any element is are true, FALSE if empty
type(enumerate(iterable, start = 0)
plist = ['to','of','and']
print(**list**(enumerate(plist)))
↳ [(0,'to'), (1,'of'), (2,'and')]

Use enumerate to make a dictionary: ex 2:
mydict = {**tuple**(enumerate(mytup))}
For dictionaries it enumerates keys unless you specifiy values, ex 3:
print(**dict**(enumerate(mydict.values())))

type([iterable])
↳a mutable sequence; if a=[7,8,9] then list([a]) returns [[7, 8, 9]]
max(type) **min(type)** - **not** for ~~tuples~~
sum(iterable [, start]) must be all numeric, if a=[8,7,9] then sum(a) returns 24
sorted(iterable [,key=][,reversed]) reversed is Boolean with default False; strings without key sorted alphabetically, numbers high to low; key examples: print (sorted(strs, key=len)) sorts by length of each str value; 2 examp: key= strs.lower, or key = lambda tupsort: tupitem[1]
reversed(seq) - **reversed is tricky**, does **not** return a reversed list; if a=[4,5,6,7] then for i in reversed(a) yeilds 7/6/5/4; to get a reversed list for list mylist use:newlist = **list**(reversed(mylist))

range (stop) or
range (start, stop [, step])
tuple(iterable) not a function, an immutable sequence, mytuple=('dog',42,'x')
next(iterator [,default]) next item from iterator by calling next(iter). Default is returned if the iterator is exhausted, otherwise StopIteration raised.

```
>>> Mylist =[2,4,6,8];  MyItNum = iter(Mylist)
>>> next(MyItNum)  —> 2
>>> next(MyItNum)  —> 4        .....etcetera
```

File open (and methods)

fileobject=**open**(file [,mode],buffering])
The basic modes: **r, r+, w, w+, a** ..more
file object methods: **.read(size)**, **.readline**, **.readlines**, **list(fo)**, **.write(string)**, **.close**

```
with open("C:\Python351\Jack.txt","r+") as sprattfile:
  sprattlist=sprattfile.read().splitlines() *<- removes '/n'
print(sprattlist)
-->['Jack Spratt', 'could eat ', 'no fat.', 'His Wife', 'could eat', 'no lean.']   *The WITH structure auto closes the file.
```

Other Functions filter(), vars(), dir(), super(), globals(), map(), dict(), setattr(), bytearray(), oct(), set(), classmethod(), zip(), locals(), __import__(), object(), memoryview(), hasattr(), issubclass(), exec(), compile(), hash(), isinstance(), complex(), bytes(), iter(), delattr(), property(), type(), getattr(), frozenset(), staticmethod()

String Methods

.find(sub[, start[, end]])
↳1st char BEFORE sub is found or -1 if not found
.capitalize() ↳first character cap
.lower() ↳ a copy of the string with all t converted to lowercase.
.center(width[, fillchar])
string is centered in an area given by width using fill character 'fillchar'
.ljust(width [, fillchar]) or **.rjust()**
.count(sub[, start[, end]])
number of substrings in a string
.isalnum() .isnumeric() .isalpha .isdigit() .isspace() .islower() .isupper .isprintable() may be null
↳ true if all char meet condition & at least one char in length
.replace(old, new[, count])
↳ a copy of the string with substring old replaced by new. If opt argument count is given, only first count are replaced.
.rfind(sub[, start[, end]])
↳ the **highest index** in the string where substring sub is found, contained within slice [start:end]. Return -1 on failure.
.strip([chars]) ↳ a copy of the string with the leading and trailing characters removed. The chars argument is a string specifying the set of characters to be removed. If omitted or None, the chars argument removes whitespace.
.zfill(width) ↳ a copy of the string left filled with ASCII '0' digits to make a string of length width. A leading sign prefix ('+'/'-') is handled by inserting the padding after the sign character rather than before. The original string is returned if width is less than or equal to len(str).
str.split() - separates words by space

BIG DADDY'S PYTHON TOOLBOX For 3.5

CC BY NC SA R4 © 2016 John A. Oakey V060516B

Escape Characters

Non-printable characters represented with backslash notation:
\a Bell or alert, **\b** Backspace, **\cx** Control - x, **\C-x** Control-x, **\e** Escape, **\f** Formfeed, **\M-\C-x** Meta-Control-x, **\n** Newline, **\s** Space, **\t** Tab, **\v** Vertical tab, **\x** Character x, **\r** Carriage return, **\nnn** Octal notation, where range of n is 0-7 **\xnn** Hexadecimal notation, n is in the range 0.9, a.f, or A.F

String Format Operator: %

% is used with print to build formatted strings

```
print ("My horse %s has starting slot %d!" % ('Arrow', 5))
```

Where the % character can format as:
%c character, **%s** string, **%i** signed decimal integer, **%d** signed decimal integer, **%u** unsigned decimal integer, **%e** exponential notation, **%E** exponential notation, **%f** floating point real number, **%g** the shorter of %f and %e, **%G** the shorter of %f and %E
also: * specifies width, **-** left justification, **+** show sign, **0** pad from left with zero, **(& more)**

Data Containers Methods / Operations

Tuples fixed, immutable sets of data that can not be changed mytup=(7,'yes',6,'no') a 1 element tuple requires a comma xtup=('test',)
Indexing and slicing the same as for strings.
tuple(sequence or list) - converts list to tuples: newtup = tuple(mylist) ;
len(tuple); **max(tuple)**; **min (tuple)**

Dict (dictionary) - a series of paired values.
d = { 'a':'animal', 2:'house', 'car':'Ford', 'num': 68}
d.keys() - value of d; **d.values()**;
d.items() - pairs list; **len(d)**;
d[key] = value; **del d[key]**; **d.clear()** remove all; **key in d**; **key not in d**; **keys ()**; **d.copy()** makes a shallow; **fromkeys (seq[, value])** from keys() is a class method - returns a new dictionary value defaults to None.
get(key[, default]) ; **items()**
~~**iteritems()**~~ ; ~~**itervalues()**~~ ; ~~**iterkeys()**~~
d.items(); **d.values()**; **d.keys()**
pop(key[, default]) remove and re-turn its value or default; **popitem()**;
setdefault(key[, default])
update([other])
To find a key if you know the value:

```
mykey=[key for key, value in mydict.items()if value==theval][0]
```

Lists
lst[i] = x item lst of s is replaced by x
lst[i:j] = t slice of s from i to j is replaced by the contents of iterable t
del lst[i:j] same as lst[i:j] = []
lst[i:j:k] = t the elements of s[i:j:k] are replaced by those of t
del lst[i:j:k] removes the elements of s [i:j:k] from the list
lst.append(x) appends x to the end of the sequence (same as lst[len(lst):len(lst)] = [x])
lst.clear() removes all items from s (same as del lst[:])
lst.copy() creates a shallow copy of s (same as lst[:])
lst.extend(t) or **s += t** extends lst with the contents of t (for the most part the same as **s**[len(s):len(s)] =t)
lst *= n updates lst with its contents repeated n times
lst.insert(i, x) inserts x into s at the index given by i (same as **lst[i:i] = [x]**)
lst.pop([i]) retrieves the item at i and also removes it from s
lst.remove(x) remove the first item from lst where **lst[i] == x**
lst.reverse() reverses the items of s in place
lst.sort() sort ascending, return None

Arrays - none, use **numpy** or **array** module or forget it.

Sets an unordered collection of unique immutable objects - **no multiple occurrences of the same element**

```
myset = set("Bannanas are nice");  print(myset)
```

↳: {'i', 'e', 's', 'a', 'B', ' ', 'c', 'r', 'n'}
add(), **clear()**, **pop()**, **discard()**, **copy** **difference()**, **remove()**, **isdisjoint()**, **issubset()**, **issuperset()**, **intersection()**
Example: Myset.add('x')

Useful Modules

Good 3rd Party Index:
https://pymotw.com/2/py-modindex.html
Python Standard Library Module Index with links:
https://docs.python.org/2/library/
pip is normally installed with Python but if skipped the **ensurepip** PACKAGE will bootstrap the installer into an existing installation.
python -m pip install SomePackage - command line
sys stdin standard input, stdout std output, exit("some error message")
os deep operating system access .open(name [,mode[, buffering]]) modes: '**r**' reading, '**w**' writing, '**a**' appending, binary append '**b**' like 'rb'
time .asctime(t) .clock() .sleep(secs)
datetime date.today() datetime.now()
random .seed([x]) .choice(seq) .randint (a,b) .randrange(start, stop [, step]) .random() - floating point [0.0 to 1.0]
csv import/export of comma separated values .reader .writer .excel
itertools advanced iteration functions
math like Excel math functions .ceil(x), .fsum (iterable), .factorial(x), .log(x[,base]), pi, e See also **cmath** for complex numbers
urllib for opening URLs, redirects, cookies, etc
pygame see http://www.pygame.org/hifi.html
tkInter Python's defacto std GUI - look it up
calendar—a world of date options

```
>>> import calendar
>>> c = calendar.TextCalendar(calendar.SUNDAY)
>>> c.prmonth(2016, 9)
   September 2016
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

This only works with a mono-spaced font like Consolas

output

curses - does not work in windows
picamera - Python access to your Raspberry Pi camera
RPi.GPIO - control Pi pins via Python
xml - to work with xml files UNSECURE
array or **numpy** work with arrays

```
Arrayname = array(typecode, [Initializers]
a = numpy.array([[1,2,3,4],[5,6,7,8]])
```

tarfile / **zipfile** - file compression
multiprocessing - take the course if you can handle it
wave - interface to wav format
yahoo-finance—to get stock data From PyPi **$ pip install yahoo-finance** use for historic data
googlefinance 0.7—real-time stock data **$ pip install googlefinance**

(re)Regular Expresions module
Searching for pattern matches: Top level functions (match, search,etc.) mirror arguments in corresponding compiled pattern method.
Compile ex: re.compile(pattern)

```
mypat=re.compile(r'\d..\w') then
myso=mypat.search(str) myso is search obj
Search object attributes:
group(), start(), end(), span()
```

Topline functions ex: myso=re.search (r'\d..\w', str)
search(pat,str)↳ True or None
match(pat,str) start of str ↳ **True or None**

```
match = re.search(pattern, string)
if match:
    process(match)
```

fullmatch, findall, escape, purge
Flags: S (DOTALL), A, I (IGNORECASE), M(MULTILINE ^$), X (VERBOSE), U,
Matching Characters: **use r' to match literally**; in v3 match is Unicode by default
\d any decimal digit \D non-decimal
\w any alphanumeric \W non-alphanum
\s any white space chr \S non-whtspace
any except newline * 0 or more + 1 or more ? 0 or 1 X{n} exactly n ,'X' chars X{m,n} between m & n X's **$** end of str
[ ] contains a set of chars to match
'-' – a range – [a-c] matches a,b,or c
special chars lose meaning inside [ ]
^ as 1st char starts complimentary match
| OR: a|b matches a OR b (...) what-ever re is in the parens (?abcdef) one or more letters in parens (?=...) a look ahead assertion, "only if" (?!=...) negated look-ahead assertion, "not if"
\A match only at start of string
\Z match only end of string \b empty string at the start/end of a word
Modifying Strings:
split() str into a list at re match
sub(pat,repl,str) - repl can be a function
subn() like sub but↳ the new str

Operators

Math: **+**, **-**, *, **/**, **//** (floor or truncated division), ** (expo-nent), **%** (mod or modulo returns the remainder) **x = 8%3; print(x)** ↳2
Assignment: (execute & assign) **=**, **+=**, **-=**, ***=**, **/=**, ****=**, **%=**
Boolean/Logical: **and, or, not**
Comparison: **<**, **<=**, **>**, **>=**, **is**, **is not**, **==** (equal), **!=**(not equal)
Special String: **+** concatenation (repetition), **[]** (slice), **[:]** (range slice), **in** (true if found, if "c" in "cat"), **not in**, **r** (r'*str*' – raw string suppresses ESC characters)
Identity: **is/is not** checks if variables point to the same object
Bitwise: **&**, **|** (or), **^** (xor), **~** (flips), **<<** (shift lft), **>>**(shift rt)
New Soon: **@** - a matrix multiplier
Note: operator module adds more.

comments and suggestions appreciated:
john@johnoakey.com

Basic Programming Examples: http://www.java2s.com/Tutorial/Python/CatalogPython.htm or https://wiki.python.org/moin/BeginnersGuide/programmers/SimpleExamples