**Reserve Words**

**Comparison / 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, ex: 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

while 1:
Reading Keystrokes
Not needed within the [], {}, or ()
if a == 1

def if_example(a)
if     elif     else
break
for
alist=[
for
Looping
assert
finally:
return statement
function is good in the outer function
non local
results are triggered by
it returns a generator whose sequential

**Functions**

with

import

class

in

Comparsion / Conjunction

if c == '\n':
text = text + c
c = sys.stdin.read(1)

**String Handling**

(`` = converts/returns
str(object) string value of object
repr(object) printable string
ascii(str) printable string
eval(expression) 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) character value of Unicode character
slice() or slice(start, stop, [step])
an object selected by slice (start, stop, and step) word = “Python”; word[2:] = “tho
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(‘’, format(11.234567890)) yields ‘’11.23457%’’

**Number Handling**

abs(x) absolute value of x
bin(x) binary to integer: ex: bin(5) ‘0b101’

(one 4, no 2’s, one 1)
divmod(x, y) takes two (non complex)
numbers as arguments, returns a pair of numbers
– quotient and remainder using integer division
divmod(x, y) a floating point number from a number or string
hex(x) an integer to hexadecimal string

“0x10000”
int(x, y) an integer from a number or string
pow(x, y [, z]) x to y, if z is present
returns x to y, modulo
round(number [,digits]) floating point number rounded to digits; Without digits it returns 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)

for i in mydict.items():
print(‘key = ‘, key, ‘value = ‘, value)

if id(object) == id(object):
return True
else:
return False

tuple()
list()
dict()

**Data Container Functions**

list/tuple/dict
all(iterable) TRUE if all elements are true
any(iterable) TRUE if any element is true
enumerate(iterable, start = 0) list

**Functions**

def def_example(a):
def a == 1:
print(‘One’)

def if_example(a):
def a == 1:
print(‘Two’)
else:
print(‘Some other’)

**Decision Making**

if if_example(a):
if a == 1:
print(‘One’)
elif a == 2:
print(‘Two’)
elif:
print(‘Some other’)

**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: ‘, text)

**String Handling**

(`` = converts/returns
str(object) string value of object
repr(object) printable string
ascii(str) printable string
eval(expression) 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) character value of Unicode character
slice() or slice(start, stop, [step])
an object selected by slice (start, stop, and step) word = “Python”; word[2:] = “tho
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(‘’, format(11.234567890)) yields ‘’11.23457%’’

**Number Handling**

abs(x) absolute value of x
bin(x) binary to integer: ex: bin(5) ‘0b101’

(one 4, no 2’s, one 1)
divmod(x, y) takes two (non complex)
numbers as arguments, returns a pair of numbers
– quotient and remainder using integer division
divmod(x, y) a floating point number from a number or string
hex(x) an integer to hexadecimal string

“0x10000”
int(x, y) an integer from a number or string
pow(x, y [, z]) x to y, if z is present
returns x to y, modulo
round(number [,digits]) floating point number rounded to digits; Without digits it returns 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)

for i in mydict.items():
print(‘key = ‘, key, ‘value = ‘, value)

if id(object) == id(object):
return True
else:
return False

tuple()
list()
dict()

**Data Container Functions**

list/tuple/dict
all(iterable) TRUE if all elements are true
any(iterable) TRUE if any element is true
enumerate(iterable, start = 0) list

**Functions**

def def_example(a):
def a == 1:
print(‘One’)
elif a == 2:
print(‘Two’)
elif
print(‘Some other’)

**Decision Making**

if if_example(a):
if a == 1:
print(‘One’)
elif a == 2:
print(‘Two’)
elif:
print(‘Some other’)

**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: ‘, text)
Escape Characters
Non-printable characters represented with backslash notation:
\a bell or alert, \b backspace, \d {C-x} or \C-x \e Escape, \f Formfeed, \M-\C-x Meta-Control-x, \n Newline, \r Carriage return, \s Start, \t Vertical tab, \v \x Character x, \X Character \x, \y Linefeed, \l \n Hexadecimal notation, \n in the range 0-9, \Oxxx Hexadecimal notation, \n in the range 0-9, a-f, or A-F

String Format Operator: %
Depreciated: use str.format(), however:
% 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, %d signed integer, %u unsigned decimal, %e exponential notation, %f floating point real number, %g the shorter of %e and %f, %s the short string %E also: *
 specifies width, - left justification, + show sign, 0 pad from left with zero. (much more)

Data Containers Methods / Operations
Tuples fixed, immutable sets of data that cannot be changed mytp=tuple([7,yes,6,no])
A element tuple requires a comma xtp=tuple(testlist)
Indexing and slicing the same as for strings.
=tp=tuple(testlist) - converts list to tuples, ntp=tuple(1ylist):
len(tup), max(tup), min (tup)

Dict (key:val) - "mapped" unordered pairs.
d = { 'a':animal, '2':house, 'car':Ford, 'num':68}
k in d retrieves keys() a value of d, d.values()
d.items() - pairs list, len(d), d.keys() = value, del(d)[key], d.clear() remove all
key in d; key not in d; keys(); d.copy()
(makes a shallow copy, 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:
Key Wanted = key for key, val in mydict.items(): if value==TheValueYouHave[0] all one line

Lists
lst[i] = x item i is replaced by x
lst[i:j] = slice of s from i to j is replaced by the contents of the iterable t

del lst[i:j] same as lst[i:j] = []
del(lst[i:j]) = t the elements of s[i:j] are replaced by those of t
del(lst[i:j]) removes the elements of s[i:j] from the list
lst.append(x) adds 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() shallow copy (same as lst)
lst.extend(t) or s += t extends lst with the contents of t (for the part most the same as s[len(s):len(s)] = t)
lst *= n updates list 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) removes the first item from lst where lst[i] == x
lst.reverse() reverses the items of s in place

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:
set = set(“Banannas are nice”); print(set)
set.append(”Bannanas are nice”)  # error
set.add(”Bannanas are nice”)
set.discard(”Bannanas are nice”)
set.difference(”Bannanas are nice”)  # empty
set.intersection(”Bannanas are nice”)  # empty
set.isdisjoint(”Bannanas are nice”)
s.setnone() sort ascending, return None

Useful Modules
Good 3rd Party Index:
https://pymotw.com/2/py-modindex.html
Python Standard Library Module Index with links:
https://docs.python.org/3/library/
pip is normally installed with Python but if skipped the ensurepip PACKAGE will bootstrap into an existing installation.

Python -pip install SomePackage - command line
sys stdin standard input, stdout stdout output, exit("some error message")
open(open(name, mode, buffering )

Operators
+ - * / (floor or truncated division), ** (exponent), % (mod or modulus returns the remainder) x = 8%3; print(x) % 2
Assignment: = += -= *= /= //= **= Boolean/Logical: and, or, not
Comparison: < <= >= == != etc.

Comments:

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