The Quick Python Book

Types → Objects Not Variables
∴ less casting
Goal: higher level of abstraction
Simple syntax rules
Expressive: 1 line does a lot
Readable
indention
~~{ }~~
Complete
Not fastest
Not as many libraries
Variable types may err
Py3 more consistent ≠ Py2

Getting started
idle - Python shell
_ last result
help(x) ✓syntax
modules
help(math)
dir() list of objects

Overview
p18
Vars are not declared
Vars = labels
# 1. int
2. float
3. complex 2+3j
4. boolean True, False
Lists [ ]
x[0:2] = [0,1] not 2
Δ type
len, min, max, +, * append
index insert, pop, remove extend
reverse sort
Modifiable

Tuples ( )
immutable
Keys for dictionaries
Strings
immutable
Use " to ll java
""""" Documentation """""
re for regular expressions
Functions → new strings
Dictionaries
= associative arrays aka % hashes
clear, copy, get, has_key
items keys/values, update
Keys must be immutable
∴ tuples not list
Sets
unique list
frozenset immutable
File object
f = open("File", "r")
sys lib has stdin, stdout
pickle to save py. objects
if boolean:
do this
elif b2:
do that
else:
do nothing
while true:
keep going
if true:
continue (next) or break (quits loop)
Iterate over [ ], ( )
For x in item alist:
do x+3

Functions
def cfunc(self, x, y=1): default
z = x+y
return z for testing
Exceptions
try:
...
except IOError as e:
...
Run at command line
if __name__ == '__main__':
OO programing
class shape:
""""" Doc """""
def __init__(self, input)
self.in = input
def __str__(self):
output from print
subclass must explicitly call its base class

Basics
Indentation = blocks
no missing, extra { }
Visual structure = logical struct.
New assignment overides old
3/4 returns a float
3//4 " int
Strings "run on & on"
import math to get sin, cos
NumPy has tools
cmath to do complex algebra
d = input("data:")
gets a string
Reread Pep8, 309 on style

Lists, tuples, sets
more flexible than arrays
tuple() = list[], but immutable
array exists, no need for it
Have indizes [0, -1]
"between" the elements
alist.append('one')
alist.extend('a', 'bunch')
alist.insert(0, 'zero'):
alist.del(3)
alist.remove('value')
will raise an error (catch?)
alist.sort(key = my_rule)
sorts in place
tuples, dict use sorted
returns new tuple
3 in (1,2,3)
>>> True
z = (0)*4 → (0,0,0,0)
2 D lists [[ ], [ ]] or higher
copy.deepcopy for list of lists
Tuples ( )
Good for dictionaries
(a,)
initialize a, b, c = 1, 2, 3
Swap a, b = b, a
Any # of values w/ *
a, b* = 1, $\underbrace{2, 3, 4}$
set() or frozenset([ ])
unique
x|y x&y x^y
union int -int