**Python intro notes**  (hopefully helpful for people coming from **R** )

Notes from [open.hpi.de/courses/pythonjunior2020](https://open.hpi.de/courses/pythonjunior2020) and personal learning. RefCards [search](https://www.ecosia.org/search?q=Python+Refcard)

By Berry Boessenkool, [berry-b@gmx.de](mailto:berry-b@gmx.de), [brry.github.io](https://brry.github.io/), January 2021 RefCard [example](https://michaelgoerz.net/refcards/python_refcard.pdf)

!!! means this will fry your brain if you're used to R. Especially subsetting with positions will be horrible.

Most python interpreters don't print unless excplicitely stated. print() is mostly left away here for brevity.

[Download & install](https://www.python.org/downloads/), hints for [Windows users](https://docs.python.org/3.9/using/windows.html).

Official documentation: [tutorial](https://docs.python.org/3.9/tutorial/index.html), standard [libraries](https://docs.python.org/3/library/index.html), language [reference](https://docs.python.org/3/reference/index.html#reference-index), general [documentation](https://docs.python.org/3.9/index.html).

**IDEs**

|  |  |
| --- | --- |
| [PyCharm](https://www.jetbrains.com/pycharm/download) | Good for scientific development, but slow in startup |
| [VScode](https://code.visualstudio.com/Download) | (Visual studio code) increasingly popular, supports multiple languages, e.g. R |
| IDLE | Installed by default, not suitable for large projects |
| More: | [www.programiz.com/python-programming/ide](http://www.programiz.com/python-programming/ide) , colab.research.google.com |

**Syntax**

function(arg, "txt", 'single quotes', 77.86) # comment

""" multi-line comment

with line breaks """

7\*6 ; 21+21 # semicolon possible, but not good practice.  *here for effective space use*

9 // 2 ; 20 % 7 ; 3 \*\* 2 # ≈ 9 %/% 2 ; 20 %% 7 ; 3^2 in R (int.div, modulo)

a = 5 ; a += 1 # short for a = a + 1 ; a \*= a+2 # short for a = a \* (a+2)

variable\_name = "value" # naming convention: lowercase, underscore

NameError: non-existing objects - [List of errors](https://docs.python.org/3/library/exceptions.html)

Variable names cannot start with numbers, Python is case sensitive

Reserved statements like else cannot be used as variable name

SyntaxError: often forgotten brackets or colons (e.g. in loops)

Method = function for an object class, e.g. listobject.append

Linter: program to analyze code style and determine structural problems (pointless lines of code, potentially overwriting variable names, etc)

**Collections (Arrays)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **type** | **example** | **changeable** | **ordered** | **indexed** | **notes** |
| list | [1,3] | yes | yes | yes | - |
| tuple | (1,2) | no | yes | yes | - |
| set | {1,4} | no, but add | no | no | no duplicates |
| dictionary | {"a":7, "b":3} | yes | no\* | by key | no duplicates |

**Data types**

integer, float, string, boolean (True/False), complex (2+1j) !!!

isinstance(7.5, int) # check for class

isinstance("Hello", (float, int, str, list, dict, tuple)) # one of types

value = input("Enter number: ") # interactive input ≈ readline("Enter num: ")

print("Type is: ", type(value)) ; type(int(value)) ; type(float(value))

value + 7 # ValueError if keyboard input was charstring

value = float(input("Give a number: ")) # read keyboard input and convert

**Lists**

list = [7, -4, 9, 1, 2, 3, 9] ; len(list)

list[0] # first element !!! ; list[1] # second element !!!

list[-1] # last element

list[-2] # second-to-last element list[5:-2] # range from left + right

list[2] = "newvalue" # overwrite third element, mixed data types possible !!!

list[2:5] # elements 3,4,5 exclusive at the right end !!!

list[4:] ; list[:6] # slicing: fifth till last element ; first till sixth

: is not an operator outside subsetting !!!

bar = list

list.append(66) # mutable object: changed even without re-assigning !!!

id(list) == id(bar) # both with 7,-4, "new value",1,2,3,9,66 !!!

list.pop() # remove last element (+ return it invisibly)

list.pop(index) # remove (+ return) selected element

del(list[index]) # only remove element at given index

list.insert(5, "new\_val") # insert at given position

list.remove(9) # remove first instance of 9

9 not in list # check for non-presence, returns a boolean ≈ ! 9 %in% vec in R

list.reverse() ; list.sort() ; list.sort(reverse=True)

list = [1,2,3,[31,32,33],4] # Nesting possible

list\_with\_charstrings[3][7] # eighth letter in fourth element

one\_list.extend(another\_list) # ≈ one\_list <- c(one\_list, another\_list) in R

one\_list + another\_list # ≈ c(one\_list, another\_list) in R

one\_list \* 2 # ≈ rep(one\_list, 2) in R

list = [] # empty list

**Dictionaries**

\*: since python version 3.6/3.7, dictionaries *are* ordered

dict = {'name': "Berry", 'age': 31} # keys (name, age) must be unique

len(dict)

dict ['name'] = "new\_value" # key + value = 'pair' dict['new\_key'] = 42

f"Hi, {dict['name']}" # fstring double and single quotes cannot be mixed

print("Hello, {name}" .format(name=dict['name']) )

dict.get('NAME', "value\_if\_key\_not\_present")

del(dict['age']) # delete pair (entire entry)

dict.keys() ; dict.values() ; dict.items()

list(dict.items()) # -> list with tuples -> very high memory usage!

the\_age = dict.pop('age') # KeyError: key no longer in dict

other\_dict = dict.copy() ; dict.clear() ; dict.update(another\_dictionary)

**Charstrings**

"Hey" + "You there" # + operator to concatenate (chain) strings

3\*"Hi" # -> "HiHiHi" # \* operator to repeat strings

len("char string") # ≈ nchar in R. Not the same as len(some\_list) !!!

print("Hey", "You there", sep=" ", end="--\n—")

charstring = "Hi this is a text. with words"

"this" in charstring # ≈ grepl("this", charstring) in R

charstring[0] # ≈ substr(cs, 1,1) in R. Not the same as some\_list[0] !!!

~~charstring[1] = "b"~~ # not possible: unlike lists, strings are immutable

charstring[5:-2] # subset region

charstring[300] # IndexError: subsetting outside of existing range

charstring.split() # split at spaces. immutable - does not change object !!!

charstring.split(".") # split at periods. The default includes \n as space.

"\_".join(["list", "of", "words"]) # ≈ paste(wordvec, collapse="\_") in R

" char string ".strip() # strip white space (or given symbols) on both sides

"CharString".lower() # ≈ tolower("CharString") in R

"CharString".startswith("Ch") # ≈ startsWith("CharString", "Ch") in R

"Chars".replace("Ch", "K") # ≈ gsub("Ch", "K", "Chars", fixed=TRUE) in R

re module for regular expressions aka. wildcards (see section Packages):

import re ; re.sub('[xyz]', 'K', "abycd") # ≈ gsub("[xyz]", "K", "abycd")

F-string placeholder (since Python 3.6). Inline arithmetics posible:

person="Berry" ; f"{person} is a nice guy with {5+5} fingers"

print("%d %s cost $%.2f" % (6, "bananas", 1.74)) # -> 6 bananas cost $1.74

print("{0} {1} cost ${2}" .format(6, "bananas", 1.74))

**Packages**

[pip](https://pip.pypa.io/en/stable/) to install packages e.g from [pypi.org](https://pypi.org) (PYthon Package Index) ≈ CRAN for R

[Anaconda](https://www.anaconda.com/products/individual) to install binary packages (also R) from their [cloud](https://anaconda.org/anaconda/repo). Anaconda Prompt: conda install pandas ; conda list

Popular packages: Data science: pandas, numpy, Machine learning: tensorflow, pytorch, Statistical analysis: scipy, Web application: django, Plotting: matplotlib, seaborn

ImportError: wrong library/module name, non-existing objects

from library import \* # all functions -> bad practice: object origin unclear

from library import function1, function2 # specific function(s)

import library then you can use library.function(…)

import library as lib then you can use lib.function(…)

from random import random, randint

random() # float between 0 (inclusive) and 1 (exclusive!!!), ≈ runif() in R

randint(1, 6) # int between start and end, including these

import os # os is a module in the standard library, no installation needed

print(os.getcwd()) # ≈ getwd() in base R ; os.chdir() # ≈ setwd()

from math import pi

**Read files**

If at os.getcwd(), there is *mydataset.py* with age = 45, we can use:

from mydataset import age # to then use age + 2

from mydataset import \* # to import all ≈ source("mydataset.py") in R

import mydataset # to then use mydataset.age + 2

print(dir(mydataset)) # list the objects in the module

import os, sys ; fname = os.path.join(sys.path[0],"file.txt") # for wd

with open(fname) as f: # with closes the connection (even in case of error)

content = f.read()#.splitlines() # ≈ readLines("file.txt") in R

**Logicals**

< ; <= ; > ; >= ; == ; != ; and ; or ; not # comparison / logical operators

7 < 8 ; "9" < "A" ; "A" < "B" ; "A" < "a" ; "a" < "b" !!order in R: "a" < "A"

|  |  |
| --- | --- |
| **Conditional code execution** | **Loops** |
| IndentationError: wrong number of spaces at the beginning of a line | |
| if cond:  do(1)  do(2)  else:  do(3)  if cond1:  do(1)  elif cond2:  do(2)  else:  do(3)  if cond1 and (cond2 or cond3):  print("stuff") | for number in (0,1,2,3): # or in range(4)  print(number) # range(8, 0, -2)  # range stop exclusive!!!  # convention for unused index variable:  for \_ in range(8): # or \_var  print("stuff")  for a,b in ( (1,4), (5,7), (6,9) ):  print(f"a={a}, b={b}, a+b={a+b}")  while cond:  run\_things()  if(cond2):  break  # continue ≈ next in R |

result = []

for item in item\_list:

new\_item = do\_something\_with(item)

result.append(new\_item)

result = [do\_something\_with(item) for item in item\_list] # list comprehension

out = [] for word in charstring\_list if word[0] == "B": out.append(word)

out = [x for x in charstring\_list if x[0] == "B"]

≈ char\_vec[substr(char\_vec,1,1)=="B"] in R # not vectorizable in Python !!!

**Write custom functions**

def greet(name, time="morning"): # name+time are parameters

return f"Hello {name}! Good {time}." # return exits function execution

# explicit return is needed !!! else a function returns None  (≈ NULL in R)

greet("Berry") # Berry+evening are arguments

greet("Berry", "evening") # parameter=argument ≈ argument=value in R

**Multiple assignment**

def myfun(x, y): # related: swap two variables: a, b = b, a

return x\*2, y\*2

a, b = myfun(3, 4) # two int objects, each with a single value

c = myfun(3, 4) # tuple object with (6, 8)

**Error management**

import traceback

try:

7 + "2" # code that might fail. int("seven") would give ValueError

except TypeError: # TypeError: wrong data type for operator or function

print("That mixed charstrings and numbers")

except Exception: # print instead of error

print("another error occured: ", traceback.format\_exc() )

else:

do("stuff")

**Write custom class**

class Person:

"I define class for the people"

pass # Placeholder for future code. A class body may not be empty.

p1 = Person() # create object instance

p1.name = "Berry" ; p1.age = 31 # add attributes

class Person: # class attributes

def \_\_init\_\_(self, name, age): # initialize (assign values) to data members

self.name = name # of the object when Person() is called

self.age = age

if name=="forbidden":

raise Exception("Name cannot be 'forbidden'") # ≈ stop("msg") in R)

def can\_watch\_movie(self): # class methods

if self.age >= 18:

return "Sure, watch it" # self represents object of class Person,

else: # always first arg to \_\_init\_\_

return "Too young, sorry"

p2 = Person("John", 25) ; p2.name ; p2.can\_watch\_movie()

p2.\_\_dict\_\_ # dictionary of all given parameters and arguments

p2 = Person("forbidden", 25) ;

**turtle**

package to draw figures on plot range -200:200

forward(nsteps), right(degrees), goto(x,y), penup(), pendown(), shape("turtle"), register\_shape(), pencolor("yellow"), bgcolor(), fillcolor(), begin\_fill(), end\_fill()

**Misc**

colors = ['red', 'blue', 'blue', 'yellow', 'blue', 'red', 'green']

import collections

collections.Counter(colors).most\_common(6) ≈ sort(table(colors))[1:6] in R