Python 3 Reference Cheat Sheet for BIOS1100 Fall 2019

Version 2019.11

Main data types

integer	10
float	10.01
string	"abc123"
list	[value1, value2, ...]
dictionary	{key1: value1, key2: value2}
boolean	True/False

List operations

`L = []`	defines an empty list
`L = [x1, x2, ...]`	defines a list
`L[i]`	retrieves item with index i
`L[i] = x`	stores x with index i
`L[-1]`	retrieves last item
`L[i:j]`	retrieves items in range i to j
`L[i:j:m]`	retrieves items in range i to j with step m
`del L[i]`	removes item with index i

List methods

`L.append(x)`	adds x to the end of the list
`L.extend(L2)`	appends L2 to the end of the list
`L.insert(i, x)`	inserts x before index i
`L.remove(x)`	removes the first list item whose value is x
`L.index(x)`	find index of first occurrence of x
`L.count(x)`	count occurrences of x
`L.copy()`	returns a copy of the list

Numeric operators

`+`	addition
`-`	substraction
`*`	multiplication
`/`	division
`**`	exponent
`%`	modulus

Comparison operators

`==`	equal
`!=`	not equal
`>`	higher
`<`	lower
`>=`	higher or equal
`<=`	lower or equal

Dictionary operations

`D = {}`	defines an empty dictionary
`D = {k1:x1, k2:x2}`	defines a dictionary
`D[k] = x`	stores x associated to key k
`D[k]`	retrieves the item with key k
`del D[k]`	removes the item with key k

Dictionary methods

`D.keys()`	returns a list of keys
`D.values()`	returns a list of values
`D.items()`	returns a list of pairs (key,value)
`D.get(x)`	item with key k if k is in D, else None
`D.copy()`	returns a copy of the dictionary

Boolean operators

and	logical AND
or	logical OR
not	logical NOT

Special characters

`#`	comment
`\n`	new line

String operations

`S[i]`	retrieves character at position i
`S[-1]`	retrieves last character
`S[i:j]`	retrieves characters in range i to j
`S[i:j:m]`	retrieves characters in range i to j with step m

String methods

`S.upper()`	converts to uppercase
`S.lower()`	converts to lowercase
`S.count(x)`	counts how many times x appears
`S.find(x)`	position of the x first occurrence
`S.replace(x)`	replaces x for y
`S.strip(x)`	returns a list of values delimited by x

Short-hand syntax

`x += 1`	`x = x + 1`
`x -= 1`	`x = x - 1`
`x *= 1`	`x = x * 1`
`x /= 1`	`x = x / 1`

pylab

`from pylab import *`	Imports all functions from pylab
`from pylab import sqrt`	Imports `sqrt` function from pylab
`choice(L)`	returns a random element from L
`random()`	returns a random number between 0 and 1

Numpy arrays

`A = array([5, 6, 7])`	Defines an array
`arange(n1,n2,n)`	returns an array of numbers from n1 to n2 in steps of n
`linspace(n1,n2)`	returns an array of numbers from n1 to n2 (including) with 50 elements
`linspace(n1,n2,n)`	returns an array of numbers from n1 to n2 (including) with n elements

Tuples

a = tuple(x1, x2, x3)	defines a tuple
a[i]	retrieves item with index i

Legend	**n**: number	**D**: dictionary
x, y: any kind of data	**L**: list	**k**: dictionary key
S: string	**i,j**: list indexes	**A**: Numpy array

Python 3 Reference Cheat Sheet for BIOS1100 Fall 2019

Version 2019.11

Built-in functions

`print(x)`	prints x
`len(L)`	returns number of elements in L
`len(D)`	returns number of key, value pairs in D
`min(L)`	returns the minimum value in L
`max(L)`	returns the maximum value in L
`sum(L)`	returns the sum of the values in L
`range(n1,n2,n)`	returns a sequence of numbers from n1 to n2 in steps of n
`range(n1,n2)`	returns a sequence of numbers from n1 to n2
`range(n2)`	returns a sequence of numbers from 0 to n2
`round(n1,n)`	returns the n1 number rounded to n digits
`type(x)`	returns the type of x (string, float, list, dict …)
`int(x)`	return an integer from x
`float(x)`	return a floating point number from x
`str(x)`	return a string from x
`list(x)`	return a list from x
`help(s)`	prints help about x
`sorted(L)`	return sorted version of list L

Loops

```
while <condition> :
    <code>

x = 0
while x < 5:
    <code>
    x = x + 1

for <variable> in <list>:
    <code>

for x1, x2 in zip(L1, L2):
    <code>

for <variable> in range(n1):
    <code>

for key in D:
    print(key, D[key])

for key, value in D.items():
    <code>
```

Conditional statements

```
if <condition>:
    <code>

if <condition>:
    <code>
else:
    <code>

if <condition> :
    <code>
elif <condition>:
    <code>
...
else:
    <code>

if <...> and <...>:
    <code>

if <...> or <...>:
    <code>

if <value> in <list>:
    <code>

if <key> in <dict>:
    <code>
```

Functions

```
def function(<params>):
    """Helptext"""
    <code>
    return <...>

def function(x1, x2=3):
    """Helptext"""
    <code>
    return <...>
```

Working with files

```
f = open("filename", "r")
lines = f.readlines()
for line in lines:
    <code>
f.close()

f = open("filename", "w")
f.write("Some data\n")
f.close()

import pandas
data = pandas.read_csv(
"file.csv")
x1 = list(data["x1"])
x2 = list(data["x2"])
```

Plotting

`plot(x, y)`	Plot x versus y
`plot(x, y, 'g-', label = "label")`	Plot x versus y as a green line with a label for the legend
`xlabel("X label")`	Label for x-axis
`ylabel("Y label")`	Label for y-axis
`title("Title")`	Title of plot
`legend()`	Show the legend in the plot
`subplot(2, 1, 1)`	plot in 2 rows, 1 columns, first (top left) plot
`yscale("log")`	Use logarithmic axis on the y-axis
`axhline(3, color = "red")`	Add a red horizontal line at `y = 3`
`axvline(5, color = "blue")`	Add a blue vertical line at `x = 5`
`savefig("file.png")`	Save the plot as `file.png`
`show()`	Show the plot

Matplotlib

colors		**markers**		**linestyles**	
`"b"`	blue	`"."`	point	`"-"`	solid
`"r"`	red	`"o"`	circle	`"-.`	dash dot
`"g"`	green	`"*"`	star	`"--`	dashed
`"c"`	cyan	`"D"`	diamond	`":"`	dotted
`"k"`	black				

Legend		
x, y: any kind of data	**n**: number	**D**: dictionary
S: string	**L**: list	**k**: dictionary key
	i,j: list indexes	**A**: Numpy array