PYTHON 3 CHEATSHEET

BY
ENG. MOHAMED J.

DIY LAB
www.diylab.io

DATA TYPES

Integers	-2, -1, 0, 1, 2, 3, 4, 5
Floats	-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25
Strings	'a', 'aa', 'aaa', 'Hello!', '11 cats'

MATH OPERATORS

**	Exponent	2 ** 3 = 8
%	Remainder	22 % 8 = 6
//	Integer division	22 // 8 = 2
/	Division	22 / 8 = 2.75
*	Multiplication	3 * 3 = 9
-	Subtraction	5 - 2 = 3
+	Addition	2 + 2 = 4

COMPARASION OPTERATORS

<	less than	a < b
>	greater than	a > b
<=	less than or equal	a <= b
=>	greater than or equal	a => b
==	equal to	a == b
!=	not equal to	a != b
in	Exists in iterator?	a in ['a', 'b']
not in	Not Exists in iterator?	a not in ['a', 'b']

LOGICAL OPERATORS

and	evaluates if both sides are true
or	evaluates if at least one side is true
not	inverse a boolean type

ASSIGNMENT OPTERATORS

=	x = 2
+=	x += 2 (x = x + 2)
-=	x -= 2 (x = x - 2)
*=	x *= 2 (x = x * 2)
/=	x /= 2 (x = x / 2)
%=	x %= 2 (x = x % 2)
a, b, c = 1, 2, 3	a = 1 b = 2 c = 3
a, b = b, a	swap values

PYTHON BUILT IN FUNCTIONS

abs()	Returns the absolute value of a number	abs(-20) >> 20
any()	if any item in an iterable object is true	any([0, 1, 0]) >> True
dict()	Returns a dictionary (Array)	x = dict(name = “ALi”, age = 30) >> {”name” : “Ali”, “Age” : 30}
format()	Formats a specified value	”Ali is {} years old”.**format(30)** >> Ali is 30 years old
input()	Allowing user input	name = input('What is your name?') >> Ali
int()	Returns an integer number	x = int(3.5) >> 3
len()	Returns the length of an object	len('hello') >> 5
list()	Returns a list	x = list(('apple', 'banana', 'cherry')) >> ['apple', banana', 'cherry']
max()	Returns the largest item in an iterable	x = max(1,3, 44, 5, 10) >> 44
min()	Returns the smallest item in an iterable	x = min(1,3, 44, 5, 10) >> 1
pow()	Returns the value of x to the power of y	x = pow(4, 3) >> 64
print()	Prints to the standard output device	print("Hello World") >> Hello World
range()	Returns a sequence of numbers	for n in range(4): print(n) >> 0 >> 1 >> 2 >> 3
round()	Rounds a numbers	x = round(5.76543, 2) >> 5.77
set()	Returns a new set object	x = set(("apple", "banana", "cherry")) >> {'cherry', 'apple', 'banana'}
sorted()	Returns a sorted list	a = ("b", "g", "a", "d", "f", "c", "h", "e") x = sorted(a) >> ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
str()	Returns a string object	a = str(3.5) >> ‘3.5’
sum()	Sums the items of an iterator	a = (1, 2, 3, 4, 5) x = sum(a) >> 15
tuple()	Returns a tuple	x = tuple(("apple", "banana", "cherry")) >> ('banana', 'cherry', 'apple')
type()	Returns the type of an object	a = ('apple', 'banana', 'cherry') b = "Hello World" c = 33 print(type(a)) print(type(b)) print(type(c)) >> tuple >> str >> int

Lists

Ordered sequence of objects

EXAMPLE: [40,“python",400.2]

ORDERED: ✓ MUTABLE: ✓ UNIQUE VALUES: ✗

count()	IN: lst = [40,'python',400.2, 40]	IN: lst.count(40)		OUT: 2
append()	IN: lst = [40,'python',400.2]	IN: lst.append(”New”)	IN: print(lst)	OUT: [40,“python",400.2, “New”]
sort()	IN: lst = ["c", "f", "a", "b", "e", "d"]	IN: lst.sort()	IN: print(lst)	OUT: ['a', 'b', 'c', 'd', 'e', 'f']
pop()	IN: lst = [40,'python',400.2]	IN: lst.pop()	IN: print(lst)	OUT: [40,“python"]
sort()	IN: lst = [40,'python',400.2]	IN: lst.clear()	IN: print(lst)	OUT: [ ]
Indexing	IN: lst = [40,'python',400.2]	IN: lst[0]		OUT: 40
	IN: lst = [40,'python',400.2]	IN: lst[1] = 'hello' ✓ MUTABLE		OUT: [40,'hello',400.2]
	IN: lst = [40,'python',400.2]	IN: lst[-1] (or lst[2])		OUT: 400.2
Slicing	IN: lst = ['a', 'b', 'c', 'd', 'e', 'f']	IN: lst[1:3]		OUT: ['b', 'c']
	IN: lst = ['a', 'b', 'c', 'd', 'e', 'f']	IN: lst[:4]		OUT: ['a', 'b', 'c', 'd']
	IN: lst = ['a', 'b', 'c', 'd', 'e', 'f']	IN: lst[3:-1]		OUT: ['d', 'e']
	IN: lst = ['a', 'b', 'c', 'd', 'e', 'f']	IN: lst[: : 1] (= lst[:] = lst[: :] = lst)		OUT: ['a', 'b', 'c', 'd', 'e', 'f']
	IN: lst = ['a', 'b', 'c', 'd', 'e', 'f']	IN: lst[: : 2]		OUT: ['a', 'c', 'e']
	IN: lst = ['a', 'b', 'c', 'd', 'e', 'f']	IN: lst[: : -1]		OUT: ['f', 'e', 'd', 'c', 'b', 'a']

Tuples

Ordered immutable sequence of objects

EXAMPLE: (20,“python",300.5)

ORDERED: ✓ MUTABLE: ✗ UNIQUE VALUES: ✗

count()	IN: tpl = (40,'python',400.2, 40)	IN: tpl.count(40)	OUT: 2
Indexing	IN: tpl = (40,'python',400.2)	IN: tpl[0]	OUT: 40
	IN: tpl = (40,'python',400.2)	IN: tpl[1] = 'hello' ✗ ASSIGNMENT NOT ALLOWED FOR TUPLES	OUT: error!
	IN: tpl = (40,'python',400.2)	IN: tpl[-1] (or tpl[2])	OUT: 400.2
Slicing	IN: tpl = ('a', 'b', 'c', 'd', 'e', 'f')	IN: tpl[1:3]	OUT: ('b', 'c')
	IN: tpl = ('a', 'b', 'c', 'd', 'e', 'f')	IN: tpl[:4]	OUT: ('a', 'b', 'c', 'd')
	IN: tpl = ('a', 'b', 'c', 'd', 'e', 'f')	IN: tpl[3:-1]	OUT: ('d', 'e')
	IN: tpl = ('a', 'b', 'c', 'd', 'e', 'f')	IN: tpl[: : 1] (= tpl[:] = tpl[: :] = tpl)	OUT: ('a', 'b', 'c', 'd', 'e', 'f')
	IN: tpl = ('a', 'b', 'c', 'd', 'e', 'f')	IN: tpl[: : 2]	OUT: ('a', 'c', 'e')
	IN: tpl = ('a', 'b', 'c', 'd', 'e', 'f')	IN: tpl[: : -1]	OUT: ('f', 'e', 'd', 'c', 'b', 'a')

Dictionaries

Unordered Key:Value pairs

EXAMPLE: {“key" : "value" , "name" : “Sam"}

ORDERED: ✗ MUTABLE: ✓ UNIQUE VALUES: ✗

		IN:		IN:		OUT:
	IN:	d = {'name' : 'Ali', 'age' : 30}	d['age']			30
get()	IN:	d = {'name' : 'Ali', 'age' : 30}	d.get('name', 0)	Return value of key (”name”)		'Ali'
	IN:	d = {'name' : 'Ali', 'age' : 30}	d.get('address', 0)	Return 0 if key not exists		0
keys()	IN:	d = {'name' : 'Ali', 'age' : 30}	d.keys()			(['name', 'age'])
values()	IN:	d = {'name' : 'Ali', 'age' : 30}	d.values()			(['Ali', 30])
items()	IN:	d = {'name' : 'Ali', 'age' : 30}	d.items()			([('name', 'Ali'), ('age', 30)])
pop()	IN:	d = {'name' : 'Ali', 'age' : 30}	d.pop('age')	IN: d		{'name' : 'Ali'}
clear()	IN:	d = {'name' : 'Ali', 'age' : 30}	d.clear()	IN: d		{ }
Nested Dictionary	IN:	a = {'b' : {'c': ['d', 'e', 'f', {'g' : [5] }] } }	a['b']['c'][-1]['g'][0]	Getting No 5 from the dictionary		5

Sets

Unordered collection of unique objects

EXAMPLE: {“g”,“k"}

ORDERED: ✗ MUTABLE: ✗ UNIQUE VALUES: ✓

		IN:		IN:		OUT:
add()	IN:	s = {40,'python',400.2, 40}	s.add('New Item')	s		s = {40,'python',400.2, 40, 'New Item'}
clear()	IN:	s = {40,'python',400.2, 40}	s.clear()	s		{ }
difference()	IN:	a = {1, 2, 3, 4} b = {3, 4, 5, 6}	a.difference(b)	Different elements of a from b		{1, 2,}
pop() Remove element randomly	IN:	s = {'a', 'b', 'c', 'd', 'e', 'f'}	s.pop()	s		s = {'a', 'c', 'd', 'e', 'f'}
remove()	IN:	s = {40,'python',400.2, 40}	s.remove('python')	s		s = {40,400.2, 40}
✗ Indexing Slicing	IN:	s = {40,'python',400.2, 40}	s[0] s[0:3]	✗ INDEXING OR SLICING NOT ALLOWED FOR SETS		error!

FLOW CONTROL & FUNCTIONS

IF Statment

```
if some_condition :
    # execute some code
elif other_condition :
    # do something different
else :
    # do something else
```

All whitespace Indentations should be equal

Object	Operator		Object
int	<	==	int
float	>	!=	float
string	<=	in	string
boolean	=>	not in	boolean
list [ ]	**	*	list [ ]
tuple ( )	%	–	tuple ( )
dict { }	//	+	dict { }
set { }	/		set { }

Logical
and
or
not

Object	Operator		Object
int	<	==	int
float	>	!=	float
string	<=	in	string
boolean	=>	not in	boolean
list [ ]	**	*	list [ ]
tuple ( )	%	–	tuple ( )
dict { }	//	+	dict { }
set { }	/		set { }

For Loop

```
for item_name in my_iterable :
    # loop in some code
```

While Loop

```
while some_boolean_condition :
    # loop in some code
else :
    # do something else
```

Functions

```
def name_of_function ( ) :
    '''
    Docstring explains function.
    '''
    # execute some code
    return object to return
```

Calling the function

```
name_of_function()
```

parameters

```
def name_of_function ( a, b, c, d, e=1 ) :
    '''
    Docstring explains function.
    '''
    # execute some code
    return something to return
```

The function could return something or could not if not then this line could be removed

```
name_of_function(1, 2, 3, 4, 5)
```

Calling the function and passing arguments
When calling this function
a = 1, b = 2, c = 3, d = 4, e = 5 (if e not specified then e = 1 by default)

OPENCV COMMON FUNCTIONS

load an image from a file	cv2.imread(image_path)
Save an image to a file	cv2.imwrite(filename, image)
Display an image in a window	cv2.imshow(window_name, image)
Video capturing from video, image sequences or cameras	cv2.VideoCapture(video_source)
Convert an image color space	cv2.cvtColor(image, color_space (ex. cv2.COLOR_BGR2RGB))
Merge images to multi-channel image.	cv2.merge((image, image, image))
Draw a line on an image	cv2.line(image, (x_start, y_start), (x_end, y_end), color (ex. (0, 255, 0)), thinkness)
Draw a rectangle on an image	cv2.rectangle(image, (x_start, y_start), (x_end, y_end), color (ex. (0, 255, 0)), thinkness)
Draw a circle on an image	cv2.circle(image, (x_center, y_center), radius, color (ex. (0, 255, 0)), thinkness)
Draw a text on an image	cv2.putText(image, text, (x_location, y_location), font (ex. cv2.FONT_HERSHEY_COMPLEX), font_scale, color, thinkness)
Finds edges in an image	cv2.Canny(image, threshold_1 (200), threshold_2 (200))
Apply Gaussian Smoothing (Blur) on image	cv2.GaussianBlur(image, (kernel_size_height (7), kernel_size_width (7)), Kernel standard deviation (Default = 0))
Dilates and image	cv2.dilate(image, kernel, iterations=1)
Erodes an image	cv2.erode(image, kernel, iterations=1)
Image Perspective Transformation	cv2.getPerspectiveTransform(source_points, destination_points)
Warp Perspective for image	cv2.warpPerspective(image, perspective_image, (output_image_width, outhput_image_height))
Haar feature-based cascade classifier (for Object Detection (face, body, eye...etc)	cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
cv2..detectMultiScale(imgGray, 1.1, 10)	cv2.detectMultiScale(grayscale_image, scaleFactor (1.1), minNeighbors (3))

CATEGORY: GENERAL PURPOSE

numpy.ndarray.dtype	Return the data-type of the elements of the array. Remember, arrays are homogeneous. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.dtype.html
numpy.ndarray.ndim	Return the number of array-dimensions (rank), e.g., it will return 2 for a 4x3 array. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.ndim.html
numpy.ndarray.shape	Return a tuple representing the array dimensions, e.g., it will return (rows,columns) for a rank 2 array. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.shape.html
numpy.ndarray.size	Return the number of elements present in the array. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.size.html
numpy.save	Save an array to .npy (numpy) format. https://numpy.org/doc/stable/reference/generated/numpy.save.html
numpy.load	Load array from the .npy files. https://numpy.org/doc/stable/reference/generated/numpy.load.html
numpy.random.random	Return random floats values from the interval [0.0, 1.0), in a specified shape. https://numpy.org/doc/stable/reference/random/generated/numpy.random.random.html
numpy.random.randint	Return random integers from the half-open interval [a, b), in a specified shape. https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html
numpy.random.normal	Return random samples from a Gaussian (normal) distribution. https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html
numpy.random.permutation	Return a randomly permuted sequence from the given list https://numpy.org/devdocs/reference/random/generated/numpy.random.permutation.html
numpy.reshape numpy.ndarray.reshape	Returns an array containing the same elements with a new shape, without affecting the the original array. https://numpy.org/doc/stable/reference/generated/numpy.reshape.html https://numpy.org/doc/stable/reference/generated/numpy.ndarray.reshape.html#numpy.ndarray.reshape

CATEGORY: ARRAY CREATION

numpy.ones	Return a new array of given shape and type, filled with 1s. https://numpy.org/devdocs/reference/generated/numpy.ones.html
numpy.zeros	Return a new array of given shape and type, filled with 0s. https://numpy.org/doc/stable/reference/generated/numpy.zeros.html#numpy.zeros
numpy.full	Return a new array of given shape and type, filled with a specific value. https://numpy.org/doc/stable/reference/generated/numpy.full.html#numpy.full
numpy.eye	Return a 2-D array with 1s on the diagonal and 0s elsewhere. https://numpy.org/doc/stable/reference/generated/numpy.eye.html#numpy-eye
numpy.diag	Extract the diagonal elements. https://numpy.org/doc/stable/reference/generated/numpy.diag.html
numpy.unique	Return the sorted unique elements of an array. https://numpy.org/doc/stable/reference/generated/numpy.unique.html
numpy.array	Create an n-dimensional array. https://numpy.org/doc/stable/reference/generated/numpy.array.html
numpy.arange	Return evenly spaced values within a given half-open interval [a, b). https://numpy.org/doc/stable/reference/generated/numpy.arange.html
numpy.linspace	Return evenly spaced numbers over a specified interval [a,b]. https://numpy.org/doc/stable/reference/generated/numpy.linspace.html#numpy.linspace
numpy.ndarray.copy	Returns a copy of the array. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.copy.html

CATEGORY: OPERATING WITH ELEMENTS AND INDICES

numpy.insert	Insert values along the given axis before the specified indices. https://numpy.org/doc/stable/reference/generated/numpy.insert.html
numpy.delete	Return a new array, after deleting sub-arrays along a specified axis. https://numpy.org/doc/stable/reference/generated/numpy.delete.html
numpy.append	Append values at the end of the specified array. https://numpy.org/doc/stable/reference/generated/numpy.append.html
numpy.hstack	Return a stacked array formed by stacking the given arrays in sequence horizontally (column-wise). https://numpy.org/doc/stable/reference/generated/numpy.hstack.html
numpy.vstack	Return a stacked array formed by stacking the given arrays, will be at least 2-D, in sequence vertically. https://numpy.org/doc/stable/reference/generated/numpy.vstack.html
numpy.sort	Return a sorted copy of an array. https://numpy.org/doc/stable/reference/generated/numpy.sort.html
numpy.ndarray.sort	Sort an array in-place. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.sort.html#numpy-ndarray-sort

CATEGORY: ARITHMETIC AND STATISTICAL OPERATIONS

numpy.add	Element-wise add given arrays https://numpy.org/doc/stable/reference/generated/numpy.add.html
numpy.subtract	Subtract arguments of given arrays, element-wise. https://numpy.org/doc/stable/reference/generated/numpy.subtract.html
numpy.multiply	Multiply arguments of given arrays, element-wise. https://numpy.org/doc/stable/reference/generated/numpy.multiply.html
numpy.divide	Returns a true division of the inputs, element-wise. https://numpy.org/doc/stable/reference/generated/numpy.divide.html
numpy.exp	Calculate the exponential of all elements in the input array. https://numpy.org/doc/stable/reference/generated/numpy.exp.html
numpy.power	First array elements raised to powers from second array, element-wise. https://numpy.org/doc/stable/reference/generated/numpy.power.html
numpy.sqrt	Return the non-negative square-root of an array, element-wise. https://numpy.org/doc/stable/reference/generated/numpy.sqrt.html
numpy.ndarray.min	Return the minimum along the specified axis. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.min.html
numpy.ndarray.max	Return the maximum along a given axis. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.max.html
numpy.mean numpy.ndarray.mean	Compute the arithmetic mean along the specified axis. https://numpy.org/doc/stable/reference/generated/numpy.mean.html
numpy.median	Compute the median along the specified axis. https://numpy.org/doc/stable/reference/generated/numpy.median.html

CATEGORY: SET OPERATIONS

numpy.intersect1d	Find the intersection of two arrays. https://numpy.org/doc/stable/reference/generated/numpy.intersect1d.html#numpy-intersect1d
numpy.setdiff1d	Find the set difference of two arrays. https://numpy.org/doc/stable/reference/generated/numpy.setdiff1d.html#numpy-setdiff1d
numpy.union1d	Return the unique, sorted array of values that are in either of the two input arrays. https://numpy.org/doc/stable/reference/generated/numpy.union1d.html

Python	Python 3 documentation	https://docs.python.org/3/
	Python Tutorial	https://docs.python.org/3/tutorial/
	Python Standard Library	https://docs.python.org/3.6/library/index.html
	Hacker Rank - Python	https://www.hackerrank.com/domains/python
	Code Wars	https://www.codewars.com/users/sign_in
NumPy	NumPy Manual	https://docs.scipy.org/doc/numpy-1.13.0/contents.html
	NumPy User Guide	https://numpy.org/devdocs/user/index.html
	NumPy Reference	https://numpy.org/devdocs/reference/index.html
	Scipy Lectures	http://www.scipy-lectures.org/intro/numpy/index.html
Jupyter Notebook	Github markdown cheatsheet	https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf
	Cheatsheet by Adam Pritchard	https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
	Magic Keywords	https://ipython.readthedocs.io/en/stable/interactive/magics.html
	Notebook Examples	https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/examples_index.html
OpenCV	OpenCV Tutorials	https://docs.opencv.org/3.3.0/d9/df8/tutorial_root.html
	OpenCV Books	https://opencv.org/books/
	Learn OpenCV	https://learnopencv.com/
	pyimagesearch website	https://www.pyimagesearch.com/