Python Cheatsheet

Contents

[Python Cheatsheet 1](#_Toc392568696)

[Control Statements 3](#_Toc392568697)

[for-next loops 3](#_Toc392568698)

[CSV Files 4](#_Toc392568699)

[Read from a CSV file 4](#_Toc392568700)

[Write to a CSV file 4](#_Toc392568701)

[Database Functions 4](#_Toc392568702)

[Create a SQLite3 Database 4](#_Toc392568703)

[Insert Values into Database 4](#_Toc392568704)

[Read from a Database Table 4](#_Toc392568705)

[Parameterized Queries 5](#_Toc392568706)

[Date Functions 6](#_Toc392568707)

[Convert string to date 6](#_Toc392568708)

[Directories 7](#_Toc392568709)

[Get the Current Working Directory 7](#_Toc392568710)

[Read the files in a directory. 7](#_Toc392568711)

[Read the files in a directory with a specific extension 7](#_Toc392568712)

[Files 8](#_Toc392568713)

[Extract the file name from a path 8](#_Toc392568714)

[Read a CSV file 8](#_Toc392568715)

[Write to a Text File 8](#_Toc392568716)

[Libraries 8](#_Toc392568717)

[Find the Function Available in a Library 8](#_Toc392568718)

[Lists 9](#_Toc392568719)

[Creating and Appending to a List 9](#_Toc392568720)

[Math Functions 9](#_Toc392568721)

[Exponentiation 9](#_Toc392568722)

[pi 9](#_Toc392568723)

[Square Root 9](#_Toc392568724)

[Matrices 10](#_Toc392568725)

[Number of rows in a matrix 10](#_Toc392568726)

[Read a Matrix from a file 10](#_Toc392568727)

[Scale matrix columns 10](#_Toc392568728)

[Object Serialization 10](#_Toc392568729)

[Create an object from a stored serialization 10](#_Toc392568730)

[Strings 11](#_Toc392568731)

[Concatenate strings 11](#_Toc392568732)

[String Literals 11](#_Toc392568733)

[Sub-strings 11](#_Toc392568734)

[Program Execution 12](#_Toc392568735)

[Stopping program execution 12](#_Toc392568736)

[Timers 12](#_Toc392568737)

[Timing Code Execution 12](#_Toc392568738)

# Control Statements

## for-next loops

>>> for x in range(0,3):

print(x)

0

1

2

>>>

# CSV Files

## Read from a CSV file

import csv

with open(‘test\_tag\_speed\_km\_mins\_pace\_hour\_day\_6mos\_end\_2012\_05\_22.csv', 'rb') as csvfile:

myreader = csv.reader(csvfile, delimiter=',')

for row in myreader:

print(row) # each row is a list containing the fields

## Write to a CSV file

import csv

with open('eggs.csv', 'wb') as csvfile:

spamwriter = csv.writer(csvfile, delimiter=' ',

quotechar='|', quoting=csv.QUOTE\_MINIMAL)

spamwriter.writerow(['Spam'] \* 5 + ['Baked Beans'])

spamwriter.writerow(['Spam', 'Lovely Spam', 'Wonderful Spam'])

# Database Functions

## Create a SQLite3 Database

import sqlite3

conn = sqlite3.connect("lat\_lon\_db")

cursor = conn.cursor()

cursor.execute("create table address(name text PRIMARY\_KEY, latitude real, longitude real)")

## Insert Values into Database

cursor.execute("insert into address (name, latitude, longitude) values (?,?,?)",

("4704 Nelson Brogdon Blvd Ne Sugar Hill GA 30518", 34.1025827, -84.02279469999999))

conn.commit()

conn.close()

## Read from a Database Table

import sqlite3

conn = sqlite3.connect("lat\_lon\_db")

cursor = conn.cursor()

result = cursor.execute("select \* from address")

print(result.fetchall())

## Parameterized Queries

import sqlite3

conn = sqlite3.connect("lat\_lon\_db")

cursor = conn.cursor()

result = cursor.execute("select \* from address where street\_address = ?", ['1439 Buckeye Court Auburn CA 95603'])

x = result.fetchall()

conn.close()

# Date Functions

## Convert string to date

>>> datetime.datetime.strptime("02/05/2014", "%m/%d/%Y").date()

datetime.date(2014, 2, 5)

>>> datetime.datetime.strptime("2/5/2014", "%m/%d/%Y").date()

datetime.date(2014, 2, 5)

| **Directive** | **Meaning** | **Notes** |
| --- | --- | --- |
| %a | Locale’s abbreviated weekday name. |  |
| %A | Locale’s full weekday name. |  |
| %b | Locale’s abbreviated month name. |  |
| %B | Locale’s full month name. |  |
| %c | Locale’s appropriate date and time representation. |  |
| %d | Day of the month as a decimal number [01,31]. |  |
| %H | Hour (24-hour clock) as a decimal number [00,23]. |  |
| %I | Hour (12-hour clock) as a decimal number [01,12]. |  |
| %j | Day of the year as a decimal number [001,366]. |  |
| %m | Month as a decimal number [01,12]. |  |
| %M | Minute as a decimal number [00,59]. |  |
| %p | Locale’s equivalent of either AM or PM. | (1) |
| %S | Second as a decimal number [00,61]. | (2) |
| %U | Week number of the year (Sunday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday are considered to be in week 0. | (3) |
| %w | Weekday as a decimal number [0(Sunday),6]. |  |
| %W | Week number of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Monday are considered to be in week 0. | (3) |
| %x | Locale’s appropriate date representation. |  |
| %X | Locale’s appropriate time representation. |  |
| %y | Year without century as a decimal number [00,99]. |  |
| %Y | Year with century as a decimal number. |  |
| %Z | Time zone name (no characters if no time zone exists). |  |
| %% | A literal '%' character. |  |

Notes:

1. When used with the [**strptime()**](http://docs.python.org/2/library/time.html#time.strptime) function, the %p directive only affects the output hour field if the %I directive is used to parse the hour.
2. The range really is 0 to 61; this accounts for leap seconds and the (very rare) double leap seconds.
3. When used with the [**strptime()**](http://docs.python.org/2/library/time.html#time.strptime) function, %U and %W are only used in calculations when the day of the week and the year are specified.

# Directories

## 

## Get the Current Working Directory

>>> import os

>>> os.getcwd()

'C:\\Python27'

## Read the files in a directory.

path=r"C:\Users\bbeauchamp\Documents\Data Analytics\Customers\_and\_Projects\SGS\eta\_raw\parser"

>>> import os

>>> files = os.listdir(path)

File names are returned as elements in a list. Note that this will also read in subdirectories.

## Read the files in a directory with a specific extension

>>> import glob

>>> glob.glob(r"C:\Users\bbeauchamp\Documents\Data Analytics\Customers\_and\_Projects\SGS\eta\_raw\parser\\*.arff")

# Files

## Extract the file name from a path

>>> pathname = "C:\Users\bbeauchamp\Documents\Data Analytics\Customers\_and\_Projects\SGS\eta\_raw\parser\F000008\_km\_min.arff"

>>> from os.path import basename

>>> print basename(pathname)

F000008\_km\_min.arff

## Read a CSV file

import csv

with open('C:\\Users\\bbeauchamp\\Documents\\Data Analytics\\Customers\_and\_Projects\\SGS\\' +

'GSM\_fail\_2\_or\_more\_devices\_with\_conn\_pct.csv', 'rb') as csvfile:

myReader = csv.reader(csvfile, delimiter = ',')

print( 'test')

for myRow in myReader:

print', '.join(myRow)

## Write to a Text File

points\_file = open('C:\\Users\\bbeauchamp\\Documents\\Data Analytics\\Customers\_and\_Projects\\SGS\\test\_output', 'w')

points\_file.write('This is a test\n')

points\_file.close()

# Libraries

## Find the Function Available in a Library

import math

dir(math)

# Lists

## Creating and Appending to a List

>>> adds = []

>>> adds

[]

>>> adds.append("a")

>>> adds

['a']

>>> adds.append("b")

>>> adds

['a', 'b']

>>> 'a' in adds

True

>>> 'c' in adds

False

# Math Functions

## Exponentiation

>>> 2\*\*3

8

## pi

math.pi

## Square Root

import math

math.sqrt(25)

# Matrices

## Number of rows in a matrix

row\_count = X\_train.shape[0]

## Read a Matrix from a file

>>> import numpy as np

>>> my\_data = np.genfromtxt(filepath, delimiter=',', skip\_header=8)

(my\_data will be a 2d numpy array)

## Read the contents of a matrix column into an array

>>> mydata

array([[ 1. , 19.91142191, 16. , 74. ],

[ 2. , 17.99404762, 15. , 48. ],

[ 3. , 18.94845361, 16. , 89. ],

[ 4. , 29.55978261, 21.5 , 120. ],

[ 5. , 25.80927835, 18. , 129. ],

[ 6. , 20.21631206, 16. , 16. ],

[ 7. , 18.47900763, 15. , 15. ],

[ 8. , 18.82753165, 15. , 30. ],

[ 9. , 16.14227642, 15. , 21. ],

[ 10. , 18.10933941, 15. , 37. ],

[ 11. , 18.24694377, 14. , 151. ],

[ 12. , 17.70260223, 12. , 79. ]])

>>> rainfall\_mm = mydata[:,3]

>>> rainfall\_mm

array([ 74., 48., 89., 120., 129., 16., 15., 30., 21.,

37., 151., 79.])

## Scale matrix columns

from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(X)

X\_scaled = scaler.transform(X)

# Object Serialization

## Create an object from a stored serialization

# Load the dictionary back from the pickle file.

import pickle

favorite\_color = pickle.load( open( "latlong.p", "rb" ) )

print(favorite\_color["330 Lee Industrial Blvd Austell, Ga 30168"])

print(favorite\_color["330 LEE INDUSTRIAL BLVD AUSTELL, GA 30168"])

# Strings

## Concatenate strings

>>> 'is' + 'test'

'istest'

## String Literals

>>> x = r"\n"

>>> x

'\\n'

>>>

## Sub-strings

>>> x = "Hello World!"

>>> x[2:]

'llo World!'

>>> x[:2]

'He'

>>> x[:-2]

'Hello Worl'

>>> x[-2:]

'd!'

>>> x[2:-2]

'llo Worl'

# Program Execution

## Stopping program execution

sys.exit('exiting....')

# Timers

## Timing Code Execution

>>> import time

>>> start\_time = time.time()

>>> print (time.time() - start\_time)

18.8489999771