CSV Module

Table of Contents

[Pretty Printed 2](#_Toc13644749)

[A cool trick to show only first five rows 2](#_Toc13644750)

[writer(), DictWriter() 2](#_Toc13644751)

[writer() 2](#_Toc13644752)

[DictWriter() 2](#_Toc13644753)

[CoreyMS 3](#_Toc13644754)

[reader() 3](#_Toc13644755)

[delimiter 3](#_Toc13644756)

[reader(), writer() 3](#_Toc13644757)

[DictReader() 4](#_Toc13644758)

[Printing [emails] only from dictionary 4](#_Toc13644759)

[DictReader(), DictWriter() 4](#_Toc13644760)

[Parsing names from a CSV to an HTML list 5](#_Toc13644761)

[Sentdex 6](#_Toc13644762)

[index() 6](#_Toc13644763)

[Example: Asking user for [email] input and printing user’s associated firstname and lastname 6](#_Toc13644764)

[Socratica 7](#_Toc13644765)

[Spliting data without csv 7](#_Toc13644766)

[Using csv module, convert the content of this csv file to its proper data type and add it to a list (datas) 7](#_Toc13644767)

[Calculating stock return 8](#_Toc13644768)

# Pretty Printed

## A cool trick to show only first five rows

import csv

with open('C:/py/modules/csv/csv\_files/names.csv') as file:

reader = csv.DictReader(file)

count = 0

print(f"{reader.fieldnames[0]} {reader.fieldnames[1]}") # prints header

for row in reader:

if count > 5:

break

count += 1

print(f"{row['first\_name']} {row['last\_name']}")

# John Doe

# Mary Smith-Robinson

# Dave Smith

# Jane Stuart

# Tom Wright

# Steve Robinson

## writer(), DictWriter()

### writer()

with open('names\_new\_2.csv', 'w', newline='') as file:

writer = csv.writer(file)

# Error: writerow() takes exactly one argument (4 given)

writer.writerow(['John', 'Doe', 27, 'john.doe@gmail.com'])

for i in range(1, 100, 2):

writer.writerow([f'Marry{i}', f"Doe{i}", i, f"email{i}@hotmail.com"])

### DictWriter()

with open('names\_new.csv', 'w', newline='') as file:

writer = csv.DictWriter(file, fieldnames = ['firstname', 'lastname'])

writer.writeheader() # this will write the fieldnames at top of csv file

writer.writerow({"firstname":"John", "lastname":"Doe"})

# writer.writerow([{"firstname":"John", "lastname":"Doe"}]) # Caution: No Brackets

writer.writerow({"firstname":"Mary", "lastname": 'Doe'})

for i in range(1, 200, 3):

writer.writerow({"firstname":f"Mr.{i}", "lastname":f"Doe{i}"})

# CoreyMS

To pull data from a CSV file, you must use the **reader()** function to generate a reader object.

The **writer()** function will create an object suitable for writing. To write data you will need to use the **writerow()** function.

**Writerow()** and **append()** expects one argument, if you have several data to insert use list as single argument for **append()** and **writerow()**:

datas.append([the\_date, open\_price, high, low, close, volume, adj\_close])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## reader()

# reader()

with open('names.csv', 'r') as csv\_file:

csv\_reader = csv.reader(csv\_file)

next(csv\_reader)

for line in csv\_reader:

print(line)

## delimiter

# If a .csv file is separated by tab:

with open('names.csv', 'r') as csv\_file:

csv\_reader = csv.reader(csv\_file, delimiter='\t')

next(csv\_reader)

for line in csv\_reader:

print(line)

## reader(), writer()

# reader() & writer()

with open('coreyms/bio.csv', 'r') as csv\_file:

csv\_reader = csv.reader(csv\_file)

with open('names\_copy.csv', 'w') as names\_copy:

csv\_writer = csv.writer(names\_copy, delimiter='-')

for line in csv\_reader:

csv\_writer.writerow(line)

## DictReader()

# DictReader()

with open('coreyms/bio.csv', 'r') as bio\_file:

bio\_reader = csv.DictReader(bio\_file)

for line in bio\_reader:

print(line)

# OrderedDict(

[('first\_name', 'John'), ('last\_name', 'Doe'), ('email', 'john-doe@bogusemail.com')]

)

# OrderedDict(

[('first\_name', 'Mary'), ('last\_name', 'Smith-Robinson'), ('email', 'maryjacobs@bogusemail.com')]

)

...

### Printing [emails] only from dictionary

# DictReader()

with open('coreyms/bio.csv', 'r') as csv\_file:

csv\_reader = csv.DictReader(csv\_file)

for line in csv\_reader:

print(line['email'])

# john-doe@bogusemail.com

# maryjacobs@bogusemail.com

# davesmith@bogusemail.com

# janestuart@bogusemail.com

# tomwright@bogusemail.com

## DictReader(), DictWriter()

with open('coreyms/bio.csv', 'r') as bio\_file:

csv\_reader = csv.DictReader(bio\_file)

with open('coreyms/bio\_dict.csv', 'w') as bio\_file:

fieldnames = ['first\_name', 'last\_name', 'email']

bio\_writer = csv.DictWriter(bio\_file, fieldnames=fieldnames, delimiter='\t')

bio\_writer.writeheader()

for line in csv\_reader:

del line['email']

bio\_writer.writerow(line)

## Parsing names from a CSV to an HTML list

import csv

names = []

with open('C:/py/modules/csv/csv\_files/patreons.csv') as patreon\_contributor:

read\_manager = csv.DictReader(patreon\_contributor)

next(read\_manager)

next(read\_manager)

for row in read\_manager:

if row['FirstName'] == 'No Reward':

break

names.append(f"{row['FirstName']} {row['LastName']}")

output = "\n<ul>"

for name in names:

output += f"\n\t<li>{name}</li>"

output += "\n</ul>"

print(output)

"""

<ul>

    <li>Dave Smith</li>

    <li>Mary Jacobs</li>

    <li>Jane Stuart</li>

    <li>Tom Wright</li>

    <li>Steve Robinson</li>

    <li>Kurt Wright</li>

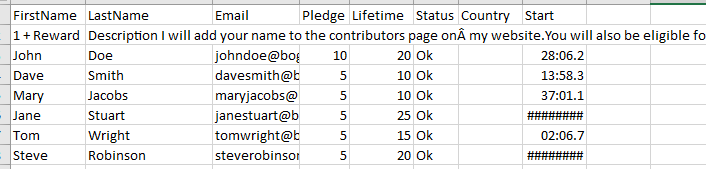
    ...

    <li>Sam Wilks</li>

    <li>Jane Stuart</li>

    <li>Maggie Jefferson</li>

</ul>

"""

# Sentdex

## index()

import csv

lists = ['John', 'Doe', '50000$', 'CEO']

res = lists.index('CEO')

# print(res) # 3

## Example: Asking user for [email] input and printing user’s associated firstname and lastname

# First off, we will store firstname, lastname and email within its separate list

# so that we can use index() function in order to find associated firstname and lastname based on index

path = 'C:/py/modules/csv/csv\_files/names.csv'

email = []

firstname = []

lastname = []

with open(path) as names\_file:

reader\_manager = csv.reader(names\_file)

next(reader\_manager)

for row in reader\_manager:

first\_name = row[0]

last\_name = row[1]

\_email = row[2]

firstname.append(first\_name)

lastname.append(last\_name)

email.append(\_email)

res = input('Enter your "email" to show your associated firstname and lastname?')

idx = email.index(res)

print('Your fullname is ', firstname[idx], " ", lastname[idx])

# Socratica

import csv

from datetime import datetime

path = 'C:/Users/Basir Payenda/Desktop\py/modules/csv/google\_stock\_data.csv'

## Spliting data without csv

file = open(path)

lines = [i.strip().split(',') for i in file]

for line in lines:

print(line)

# ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']

# ['8/19/2014', '585.002622', '587.342658', '584.002627', '586.862643', '978600', '586.862643']

# ['8/18/2014', '576.11258', '584.512631', '576.002598', '582.162619', '1284100', '582.162619']

## Using csv module, convert the content of this csv file to its proper data type and add it to a list (datas)

csv\_reader = csv.reader(file)

next(csv\_reader)

datas = []

for line in csv\_reader:

"""

By default the data of .csv file is string. In order to do mathematic calculations

we need to convert those strings to its associated data types and then calculate

"""

the\_date = datetime.strptime(line[0], '%m/%d/%Y')

open\_price = float(line[1])

high = float(line[2])

low = float(line[3])

close = float(line[4])

volume = float(line[5])

adj\_close = float(line[6])

datas.append([the\_date, open\_price, high, low, close, volume, adj\_close])

# now [datas] list holds data with proper data types, in example below we did mathematic operations on this list.

print(datas[0])

## Calculating stock return

print('stock\_return\_formula = (todays\_price - yesterdays\_price)/ yesterdays\_price')

path = 'C:/Users/Basir Payenda/Desktop\py/modules/csv/stock\_return.csv'

stock\_return = open(path, 'w')

writer = csv.writer(stock\_return)

# writerow() takes exactly one argument (2 given)

writer.writerow(['Date', 'Stock Return'])

for i in range(len(datas)-1):

# today

todays\_data = datas[i] # look above example we created [datas] list

todays\_date = todays\_data[0]

todays\_price = todays\_data[-1]

# yesterday

yesterdays\_data = datas[i+1]

yesterdays\_price = yesterdays\_data[-1]

"""

We converted datas of "google\_stock\_data.csv" in its associated data types then stored it in "datas" list (see above example). Now, we can do division and subtraction as below to calculated "stock\_return".

"""

daily\_return = (todays\_price - yesterdays\_price)/yesterdays\_price

todays\_date = datetime.strftime(todays\_date, '%m/%d/%Y')

writer.writerow([todays\_date, daily\_return])