**Lyhyt ja ytimekäs Python-opas, Pauli Tikka, 17.1.17-21.2.17**

Sisällys

[Perusteet 2](#_Toc475357516)

[Atomissa 2](#_Toc475357517)

[Powershellissä tai Geany:ssä 2](#_Toc475357518)

[Peruskomentoja 2](#_Toc475357519)

[Jos ohjelma sisältää ”argv:n” niin lisä-arvot laitetaan seuraavasti 3](#_Toc475357520)

[Näytä tiedosta vähän 3](#_Toc475357521)

[Python -ajo-ohjelman löytäminen ja sulkeminen Powershellissä: 3](#_Toc475357522)

[Koodaus, Tekstinkäsittely- ja ajo-ohjelmassa 4](#_Toc475357523)

[Koodi-1, sanakirjoista 4](#_Toc475357524)

[Koodin 1 ajo 4](#_Toc475357525)

[Koodi-2, sanakirjan sanat for loopille 4](#_Toc475357526)

[Koodin 2 ajo: 4](#_Toc475357527)

[Koodi-3, while looppi laskussa ja if lause 4](#_Toc475357528)

[Koodin-3 ajo 4](#_Toc475357529)

[Koodi-4, funktio ja positionaaliset argumentit 5](#_Toc475357530)

[Koodin-4 ajo 5](#_Toc475357531)

[Koodi-5, moduli-tiedoston funktiot ja ohjelmatiedoston funktiohaut arvoineen 5](#_Toc475357532)

[Koodin-5 ajo 5](#_Toc475357533)

[Koodi-6, luokan teko 6](#_Toc475357534)

[Koodin 6 ajo 6](#_Toc475357535)

[Koodi 7, tiedoston kanssa leikkiminen 6](#_Toc475357536)

[Koodin 7 ajo 6](#_Toc475357537)

[Koodi 8, tiedoston kanssa leikkiminen, osa2 7](#_Toc475357538)

[Koodin 8 ajo 7](#_Toc475357539)

[Koodi 9, tiedoston luonti ja kirjoitus 7](#_Toc475357540)

[Koodin 9 ajo 7](#_Toc475357541)

[Koodi 10, usean tiedoston analysointi 7](#_Toc475357542)

[Koodin 10 ajo 7](#_Toc475357543)

[Koodi 11, plottaus ja looppi 8](#_Toc475357544)

[Koodin 11 ajo 8](#_Toc475357545)

[Koodi 12, random-kävelyn (rmk) plottaus 8](#_Toc475357546)

[Koodin 12 ajo 9](#_Toc475357547)

[Koodi 13, nopat tiedostot: C:\Python34\harkat 10](#_Toc475357548)

[Koodin 13 ajo, käytetään geany –ohjelmaa, valmiina run 11](#_Toc475357549)

[Koodi 14, datan tuonti ulkosesta lähteestä 11](#_Toc475357550)

[Koodin 14 ajo (geniessä) 12](#_Toc475357551)

[Koodi 15, lisää datan tuontia ja analysointia 12](#_Toc475357552)

[Koodin 15 ajo (geanyssä) 13](#_Toc475357553)

[Python filosofointia 13](#_Toc475357554)

# Perusteet

Mitä tarvitaan python koodailuun?

Koodin kirjoitusohjelma, joku kehittynyt tekstinkäsittelyohjelma, kuten Atom. Koodin ajo-ohjelma, voi tehdä ’komentokehote’, PowerShell alias DOS, –ohjelmissa, mutta graaffinen ajo-ohjelma on selkeämpi, esim. Geany. Pythonin perusasennuksessa toimii parhaiten Powershell ja järjestelmä muuttujien lisäys windowsisa. Googletuksen jälkeen lataa ja asenna graaffinen ajo-ohjelma sekä tekstinkäsittelyohjelma. ”Python crash course” (Eric Matthes, 2016) -kirjassa hyvät ohjeet asennuksiin ja peruskäyttöön. Kannattaa olla uusin python versio ellei joku spesifinen juttusi vaadi muuta, jos et varma, niin uusin.

## Atomissa

*Kaksoisklikkaa Windows-ikonia päästäksesi ohjelmaan.*

Kirjoita ohjelmasi täydessä loistossaan python-kielellä tässä ohjelmassa teksti-tyyppisesti ja tallenna nimellä ja muodossa xyz.py.

## Powershellissä tai Geany:ssä

*Kaksoisklikkaa Windows-ikonia päästäksesi ohjelmaan.*

Aja Atomissa tekemäsi ohjelmasi suoraan Geany:ssä klikkaamalla ’suorita’ tai Powershellissä kirjoittamalla python hakemistossasi, esim. C:\python27\harkat python xyz.py (ja paina enter). Varmista, että kaikki tarvittavat tiedostot ovat samassa kansiossa ja ohjelmien näkyvissä.

### Peruskomentoja

mkdir hei -> tekee tiedoston hei (tästä lähtien oletan, että komennot tulee vahvistaa enterillä)

cd hei -> siirtää sinut tiedostoon hei

Ajaaksesi ohjelman ex1.py kirjoita Poweshellin komentokehotteessa:

**C:\python27\harkat> python ex1.py**

### Jos ohjelma sisältää ”argv:n” niin lisä-arvot laitetaan seuraavasti

C:\python27\harkat> python ex13.py pauli ratkaisi kiitos

The script is called: ex13.py

Your first variable is: pauli

Your second variable is: ratkaisi

Your third variable is: kiitos

Muuta ajoa:

python ex17.py test.txt new\_file.txt

python -m pydoc raw\_input # pydoc - -m = manual, pydoc=the Python documentation tool

### Näytä tiedosta vähän

PS C:\python27\harkat> cat test.txt

Tama on elamaa parhaimmillaan, eiko?

Kylla, tasta vaan sitten eteenpain!

Jee!!!

tee pikkutiedosto:

echo "This is a test file." > test.txt

### Python -ajo-ohjelman löytäminen ja sulkeminen Powershellissä:

Pikku python ohjelmat voi ajaa python -ajo-ohjelmassa, jonne pääset kirjoittamalla Powerhsellin oikeassa hakemistossa ”python” (+enter) ja sieltä pois kirjoittamalla quit():

**PS C:\python27\harkat> python**

Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win3

Type "help", "copyright", "credits" or "license" for more information.

>>>

, esim.:

>>> cars=['bmw','audi','toyota','subaru']

>>> len(cars)

ja

>>> quit()

PS C:\python27\harkat>

ja edelleen

>>> print 7/4

1

>>> print "test2, tarkka", 7.0/4.0

test2, tarkka 1.75

# Koodaus, Tekstinkäsittely- ja ajo-ohjelmassa

## Koodi-1, sanakirjoista

alien\_0 = {'color': 'green', 'points': 5} **#’ color’ on avain (key) ja ‘green’ on arvo (value)**

print(alien\_0['color'])

print(alien\_0['points'])

### Koodin 1 ajo

PS C:\python27\harkat> python alien.py

green

5

## Koodi-2, sanakirjan sanat for loopille

favorite\_languages = {

'jen': 'python',

'sarah': 'c',

'edward': 'ruby',

'phil': 'python'

}

for name, language in favorite\_languages.items():

print (name.title() + "'s favorite language is " +

language.title() + ".")

### Koodin 2 ajo:

PS C:\python27\harkat> python favorite\_languages.py

Sarah's favorite language is C.

Edward's favorite language is Ruby.

Jen's favorite language is Python.

Phil's favorite language is Python. **#huom. kielten järjestys…**

## 

## Koodi-3, while looppi laskussa ja if lause

current\_number = 0

while current\_number < 10:

current\_number += 1

if current\_number % 2 == 0:

continue

print(current\_number)

### Koodin-3 ajo

PS C:\python27\harkat> python counting.p

1

3

5

7

9

## Koodi-4, funktio ja positionaaliset argumentit

def describe\_pet(animal\_type, pet\_name): **#def:n jälkeen tulee funktio nimeltään “describe\_pet”**

"""Display information about a pet."""

print("\nI have a " + animal\_type + ".") **#koodin tehtävä on printata infoa eläimistä**

print("My " + animal\_type + "'s name is " + pet\_name.title() + ".")

describe\_pet('hamster','harry') **#“eläin” ja ”nimi” ovat positionaaliset argumentit funktiossa**

describe\_pet('dog', 'willie')

### Koodin-4 ajo

PS C:\python27\harkat> python pets.py

I have a hamster.

My hamster's name is Harry.

I have a dog.

My dog's name is Willie.

## Koodi-5, moduli-tiedoston funktiot ja ohjelmatiedoston funktiohaut arvoineen

***Moduli –tiedosto, abc.py, esim. pizza.py***

def make\_pizza(\*toppings): **#funktio on nyt nimeltään make\_pizza**

"""Summarize the pizza with the following toppings."""

print("\nMaking a pizza with the following toppings:")

for topping in toppings: **#tässä se for looppi**

print("- " + topping)

make\_pizza('pepperoni') **# näitä kutsutaan ”arbitary” argumenteiksi**

make\_pizza('mushrooms', 'green peppers', 'extra cheese')

***Ohjelma/ajo –tiedosto, xyz.py, esim. making\_pizza.py***

from pizza import make\_pizza **#funktion make\_pizza haku moduli –tiedostosta pizza.py**

make\_pizza(16, 'pepperoni')

make\_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')

### Koodin-5 ajo

PS C:\python27\harkat> python making\_pizza.py **#huom.! ajetaan vain ohjelma/ajo –tiedostolla!!**

Making a 16-inch pizza with the following toppings:

- pepperoni

Making a 12-inch pizza with the following toppings:

- mushrooms

- green peppers

- extra cheese

## Koodi-6, luokan teko

***Luokka –tiedosto (class file)***

class Dog(object):

"""A simple attempt to model a dog."""

def \_\_init\_\_(self, name, age) ***#funktiota nimitetään luokan alla metodiksi, method; \_,\_ kaksi kpl!***

"""Initialize name and age attributes."""

self.name = name ***# self.name:a nimitetään atribuutiksi (attribute)***

self.age = age

def sit(self):

"""Simulate a dog sitting in response to a command."""

print(self.name.title() + " is now sitting.")

def roll\_over(self):

"""Simulate rolling over in response to a command."""

print(self.name.title() + " rolled over!")

my\_dog = Dog('willie', 3) ***#my\_dog:in ja your\_dogi:in nimitys on luokan instanssi ‘instance’***

your\_dog = Dog('lucy', 3)

print("My dog's name is " + my\_dog.name.title() + ".") **#tässä alla tehtävät ko. instansseille**

print("My dog is " + str(my\_dog.age) + " years old.")

my\_dog.sit()

my\_dog.roll\_over()

print("My dog's name is " + your\_dog.name.title() + ".")

print("My dog is " + str(your\_dog.age) + " years old.")

your\_dog.sit()

your\_dog.roll\_over()

### Koodin 6 ajo

PS C:\python27\harkat> python dog.py **#luokka –tiedostossa voi olla sekä metodit, että ’ajokoodi’**

My dog's name is Willie.

My dog is 3 years old.

Willie is now sitting.

Willie rolled over!

My dog's name is Lucy.

My dog is 3 years old.

Lucy is now sitting.

Lucy rolled over!

## Koodi 7, tiedoston kanssa leikkiminen

with open('pi\_digits.txt') as file\_object: **# tiedoston pi:digits.txt avaus ja tallennus, open(x) as y**  contents = file\_object.read() **# sisällön uudelleentallennus ja luku, z=y\*read()**

print(contents.rstrip()) **# ohjelman tehtävä/pääfunktio on “printata z näytölle”**

### Koodin 7 ajo

PS C:\python27\harkat> python file\_reader.py

3.1415926535

8979323846

2643383279 **# z.rstrip():lla rivi pois lopusta**

## Koodi 8, tiedoston kanssa leikkiminen, osa2

filename = 'pi\_digits.txt' **# sama tiedosto tallennetaan muuttujaksi, filename** with open(filename) as file\_object: **# jos eri kansiossa kirjoita: open(filepath=‘C:\...\filename’)**

lines = file\_object.readlines() **# koko tiedosto luetaan rivikerrallaan, readlines()**

pi\_string = '' **# tehdään muuttuja pi\_string eli numerot tekstinä**

for line in lines: **# pi\_string:ille annetaan lines-muuttujan arvot**

pi\_string += line.strip() **# for-loopissa poistetaan rivien väliset aukot, line.strip()**

print(pi\_string) **# printataan koko rimpsu**

print(len(pi\_string)) **# printataan myös rimpsun pituus**

### Koodin 8 ajo

PS C:\python27\harkat> python pi\_string.py

3.141592653589793238462643383279 **# laita koodiin int(pi\_string) saadaksesi numerot arvoina**

32 **# rimspun pituus, 32**

## Koodi 9, tiedoston luonti ja kirjoitus

filename = 'abcd.txt' **# jos tätä tiedostoa ei ollut, niin se tehdään**

with open(filename, 'w') as file\_object: **# se avataan write –moodissa (‘w’), append (‘a’) lisää siihen**

file\_object.write("I love programming.") **# siihen kirjoitetaan… ☺**

### Koodin 9 ajo

PS C:\python27\harkat> python write\_message.py

PS C:\python27\harkat> cat abcd.txt

I love programming.

## Koodi 10, usean tiedoston analysointi

def count\_words(filename): **# sisältää laskufunktion ja suorituksen (alla)**

"""Count the approximate number of words in a file."""

try: **# jos ei poikkeuksia niin kokeillaan (try)**

with open(filename) as f\_obj:

contents = f\_obj.read()

except IOError: **# kirjassa tää poikkeus oli ‘FileNotFoundError’ (python3?)**

msg = "Sorry, the file " + filename + " does not exist."

print(msg)

else:

# Count approximate number of words in the file.

words = contents.split()

num\_words = len(words)

print("The file " + filename + " has about " + str(num\_words) + " words.")

filenames = ['alice.txt', 'siddhartha.txt', 'moby\_dick.txt', 'little\_women.txt']

for filename in filenames:

count\_words(filename) **# funktio suoritetaan looppina ‘filenames’ tiedostoille**

### Koodin 10 ajo

PS C:\python27\harkat> python word\_count.py **#ajoskripti ja funktio oli samassa tiedostossa**

The file alice.txt has about 29461 words.

Sorry, the file siddhartha.txt does not exist. **# ajo osasi myös todeta poikkeuksen**

The file moby\_dick.txt has about 215136 words.

The file little\_women.txt has about 189079 words.

## Koodi 11, plottaus ja looppi

import matplotlib.pyplot as plt **# matplotlib:iä varten käytän python3.4:sta, löyt foam.pif vääntäes**

x=list(range(1,1001)) **# 1000 arvoa ei käydä käsin läpi, tehdään autom. listalla**

y=[x\*\*2 for x in x] **# tässä looppi listan y-arvoille**

plt.scatter(x,y, s=100) **# pääfunktio on scatter, joka plottaa x&y:n piste kerrallaan**

# Set chart title and label axes. **# sitten kaikkee kuvateknistä yksityiskohtaa**

plt.title("Square Numbers", fontsize=24)

plt.xlabel("Value", fontsize=14)

plt.ylabel("Square of Value", fontsize=14)

# # Set size of tick labels.

plt.tick\_params(axis='both', which='major', labelsize=14)

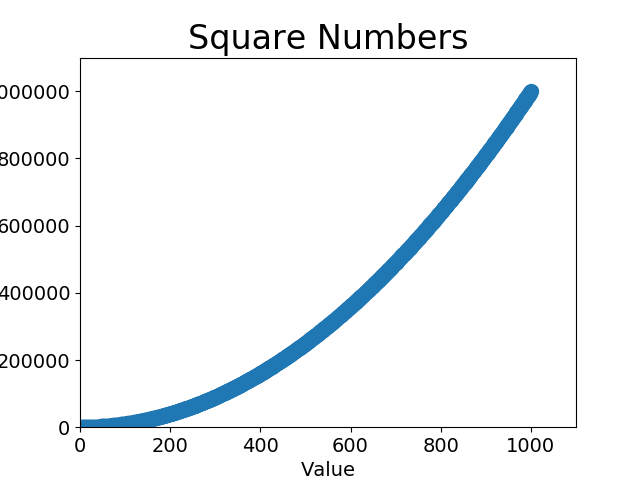
# Set the range for each axis

plt.axis([0, 1100, 0, 1100000])

plt.show()

### Koodin 11 ajo

C:\Python34\harkat>python scatter\_squares.py

 ☺

## Koodi 12, random-kävelyn (rmk) plottaus

from random import choice **# rmk:n funktiot luokka -tiedostossa**

class RandomWalk():

"""A class to generate random walks."""

def \_\_init\_\_(self, num\_points=5000): **# rw-pointtien määrän voi vaihtaa myös ajo -tiedostossa**

"""Initialize attributes of a walk."""

self.num\_points = num\_points

# All walks start at (0,0).

self.x\_values = [0]

self.y\_values = [0]

def fill\_walk(self):

"""Calculate all the points in the walk."""

# Keep taking steps until the walk reaches the desired length.

while len(self.x\_values) < self.num\_points:

# Decide which direction to go and how far to go in that direction.

x\_direction = choice([1, -1]) **# oikealla tai vasemmalle**

x\_distance = choice([0, 1, 2, 3, 4]) **# ilmeisesti voi edetä jopa neljä askelta kerrallaan!**

x\_step = x\_direction \* x\_distance

y\_direction = choice([1, -1]) **# ylös tai alas**

y\_distance = choice([0, 1, 2, 3, 4])

y\_step = y\_direction \* y\_distance

# Reject moves that go nowhere:

if x\_step == 0 and y\_step == 0: **# eli jos distance:n arvo on 0 (mitä se voi olla)**

continue

# Calculate the next x and y values.

next\_x = self.x\_values[-1] + x\_step **# -1 on edellinen “koordinaatti” esim. 57**

next\_y = self.y\_values[-1] + y\_step

self.x\_values.append(next\_x) **# x\_values:ksi annetaan next\_x edellisen kaavan mukaan**

self.y\_values.append(next\_y) **# tämä tehdään .append -funktiolla**

import matplotlib.pyplot as plt **# rmk:n plottaus ja ajo -tiedosto**

from random\_walk import RandomWalk

# Keep making new walks, as long as the program is asctive.

while True:

# Make a random walk, and plot the points.

rw = RandomWalk(100000) **# aika monta rw-pointtia laitettiin**

rw.fill\_walk()

# Set the size of the plotting window.

plt.figure(figsize=(10,6)) **# plottikuvan koko**

# Plot the points, and show the plot.

point\_numbers = list(range(rw.num\_points))

plt.scatter(rw.x\_values, rw.y\_values, c=point\_numbers,

cmap=plt.cm.Blues, edgecolor='none', s=1) **# rw-pointit värjättiin kasvavasti sinisellä**

# Emphasize the first and last points.

plt.scatter(0, 0, c='green', edgecolors='none', s=100) **# alkupiste vihreällä ja loppu punaisella**

plt.scatter(rw.x\_values[-1], rw.y\_values[-1], c='red',

edgecolors='none', s=100)

**# Remove the axes.**

plt.axes().get\_xaxis().set\_visible(False) **# kommentoi pois/True, jos haluat akselit näkyviin**

plt.axes().get\_yaxis().set\_visible(False)

plt.show()

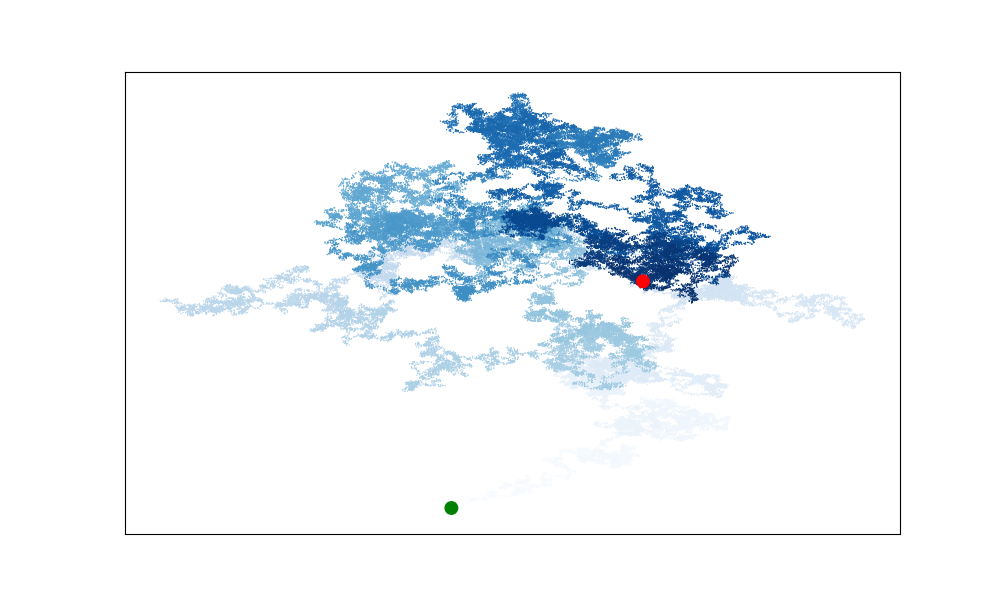
keep\_running = input("Make another walk? (y/n): ") **# kuvia tehdään kunnes toisin määrätään**

if keep\_running == 'n':

break

### Koodin 12 ajo

PS C:\python34\harkat> python rw\_visual.py

 ☺

Make another walk? (y/n): n

## Koodi 13, nopat tiedostot: C:\Python34\harkat

from random import randint **# nopat funktiot class -tiedostossa**

class Die:

**"""A class representing a single die."""**

def \_\_init\_\_(self, num\_sides=6):

**"""Assume a six-sided die."""**

self.num\_sides = num\_sides

def roll(self):

**"""Return a random value between 1 and number of sides."""**

return randint(1, self.num\_sides)

import pygal **# kahden noppan ajo –tiedosto, voi olla enemmänkin, looppaa x-labels**

from die import Die

**# Create a D6.**

die\_1 = Die()

die\_2 = Die(10)

**# Make some rolls, and store results in a list.**

results = []

for roll\_num in range(50000):

result = die\_1.roll()+die\_2.roll()

results.append(result)

**# Analyze the results.**

frequencies = []

max\_result = die\_1.num\_sides+die\_2.num\_sides

for value in range(2, max\_result+1):

frequency = results.count(value)

frequencies.append(frequency)

**# Visualize the results.**

hist = pygal.Bar()

hist.title = "Results of rolling a D6 and a D10 50000 times."

hist.x\_labels = ['2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12',

'13','14','15','16']

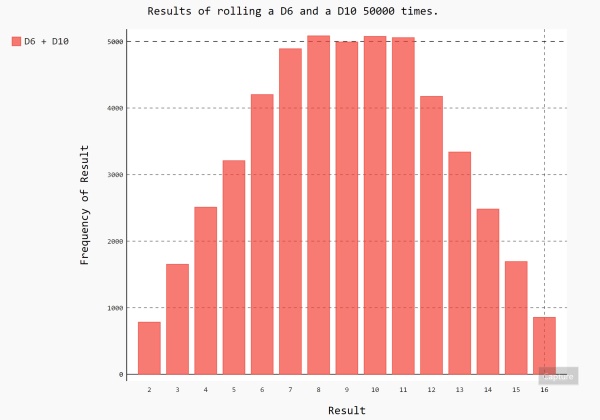
hist.x\_title = "Result"

hist.y\_title = "Frequency of Result"

hist.add('D6 + D10', frequencies)

hist.render\_to\_file('die\_visual3.svg') # <http://www.pygal.org/en/stable/>

## Koodin 13 ajo, käytetään geany –ohjelmaa, valmiina run



## Koodi 14, datan tuonti ulkosesta lähteestä

from pygal.maps.world import COUNTRIES **# modulitiedosto “country codes.py”**

from pygal\_maps\_world import i18n

def get\_country\_code(country\_name):

"""Return the Pygal 2-digit country code for the given country."""

for code, name in COUNTRIES.items():

if name == country\_name:

return code

**# If the country wasn't found, return None.**

return None

import json **# Ajo –tiedosto, world\_population.py, tarvitaan json moduli**

from country\_codes import get\_country\_code **# tässä kutsutaan funktiotiedostoa**

**# Load the date into a list.**

filename = 'population\_data.json' **# datan (\*.json) tulee löytyä samasta kansiosta kuin ajo -tiedosto**

with open(filename) as f:

pop\_data = json.load(f)

**# Print the 2010 population for each country.**

for pop\_dict in pop\_data:

if pop\_dict['Year'] == '2010':

country\_name = pop\_dict['Country Name']

population = int(float(pop\_dict['Value']))

code = get\_country\_code(country\_name)

if code:

print(code + ": " + str(population))

else:

**print('ERROR - ' + country\_name)**

### Koodin 14 ajo (geniessä)

ERROR - Arab World **# ohjelma ok, data:ssa puutteita, no problem**

ERROR - Caribbean small states

--snip—

zm: 12927000

zw: 12571000

## Koodi 15, lisää datan tuontia ja analysointia

import csv **# ajo –tiedostossa, high\_lows.py, tarvitaan csv funktio**

from datetime import datetime **# valmis moduli pythonissa**

from matplotlib import pyplot as plt **# pyplot funktion nimi vaihdettiin plt:ksi**

**# Get dates and high temperatures from file.**

filename = 'death\_valley\_2014.csv' **# tiedosto tuodaan \*.csv -muodossa**

with open(filename) as f: **# muutetaan tiedoston ‘ajonimeksi f’**

reader = csv.reader(f) **# lyhyt nimi, helppo laittaa funktion sisään**

header\_row = next(reader)

dates, highs, lows = [], [], []

for row in reader:**# jos data-tiedostossa on virheitä täytyy varautua huomattamalla ValueErrorista**

try:

current\_date=datetime.strptime(row[0], "%Y-%m-%d") **# pvm on \*.csv:n rivillä ‘0’**

high = int(row[1])

low = int(row[3])

except ValueError:

print(current\_date, 'missing data') # “current\_date” on tiedoston current\_date

else:

dates.append(current\_date)

highs.append(high)

lows.append(low)

print(highs)

**# Plot data.**

fig = plt.figure(dpi=128, figsize=(10, 6))

plt.plot(dates, highs, c='red', alpha=0.5)

plt.plot(dates, lows, c='blue', alpha=0.5)

plt.fill\_between(dates, highs, lows, facecolor='blue', alpha=0.1)

**# Format plot.**

title="Daily high and low temperatures - 2014\nDeath Valley, CA"

plt.title(title, fontsize=24)

plt.xlabel('', fontsize=16)

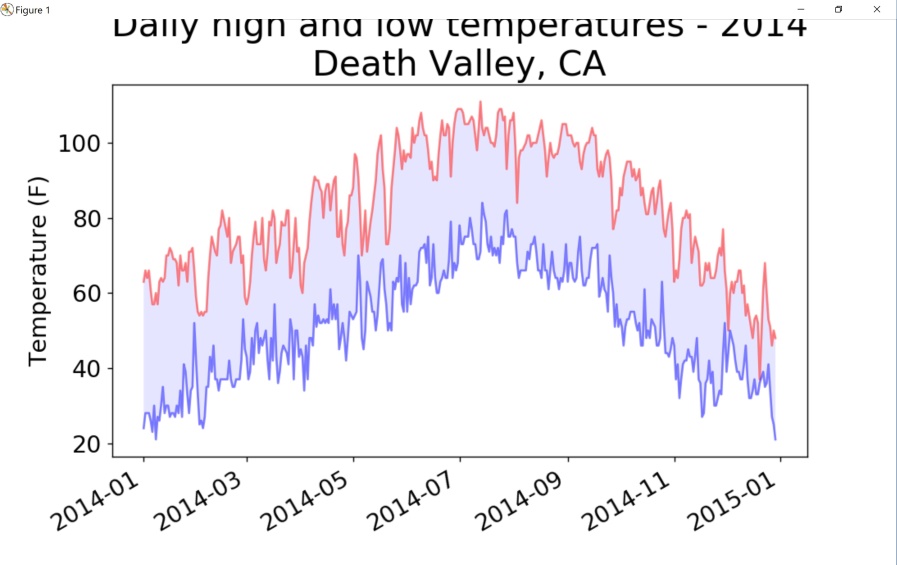
fig.autofmt\_xdate()

plt.ylabel("Temperature (F)", fontsize=16)

plt.tick\_params(axis='both', which='major', labelsize=16)

plt.show()

### Koodin 15 ajo (geanyssä)



# Python filosofointia

It does not really matter which calling style you use. As long as your function calls produce the output you want, just use the style you find easiest to understand.

Give your variables and functions descriptive names!!

A function doesn’t always have to display its output directly. ☺

Python ei toimi!? =>

1. Run ja ’Ei näy’: Laitoitko ”print” –komennon ajo –skrpitiisi? Error: Onko ajo –tiedostossasi uusin kirjasto/kirjotusvirhe, miltä data näyttää, puuttuvia rivejä tai muuta epämäärästä?

Ks. kirjastoversio win –powershellissä: ’pip show ”pygal”’ tai ’pip freeze’ tai paranna skriptiä.

Uudemmassa kirjastossa eri koodi kuin esim. oppikirjassa ja puuttuva data tulee skipata.

1. Ohjauspaneeli\Kaikki ohjauspaneelin…\Järjestelmä -> onko käyttämäsi python ’path’ ok?
2. Onko ohjelmasi tai kirjastosi samaa versiota kuin nykyinen käyttämäsi python (2/3.x)?
3. Youtubasitko/googletitko ongelmaasi? Huom. helpoin ratkaisu yleensä riittää!!
4. Koodauksen ajanhallinnasta. Käytitkö koodiasian selvittämiseen alle 10 min vai yli 30 min? Jos yli 30 min, ongelmasi on eri kuin miltä se nyt näyttää; jatka etsintääsi muualta! Jos alle 10 min; jatka alkuperäistä selvittelyä!