Programming with Python

80 **Tasks & solutions**

By Adam Higherstein

Table of Contents

Start: tool first

Start by installing python too.
Steps are here:

Install Python
https://www.python.org/downloads/

Open editor
IDLE (Python ..)

Try to print something to IDLE tool (type the code and push enter)
print ("Hello")

Start a new source file
Choose File - New File
Put there some code

print("let's start")

Save the file
code1.py

Choose Run - Run module

Good!

Task set 1: variables, operators

1. Define suitable variables that can store these values:

a) 999999

b) 5.55555555555

c) 'x'

e) 2.33

f) 10

g) 300

h) 9 billions

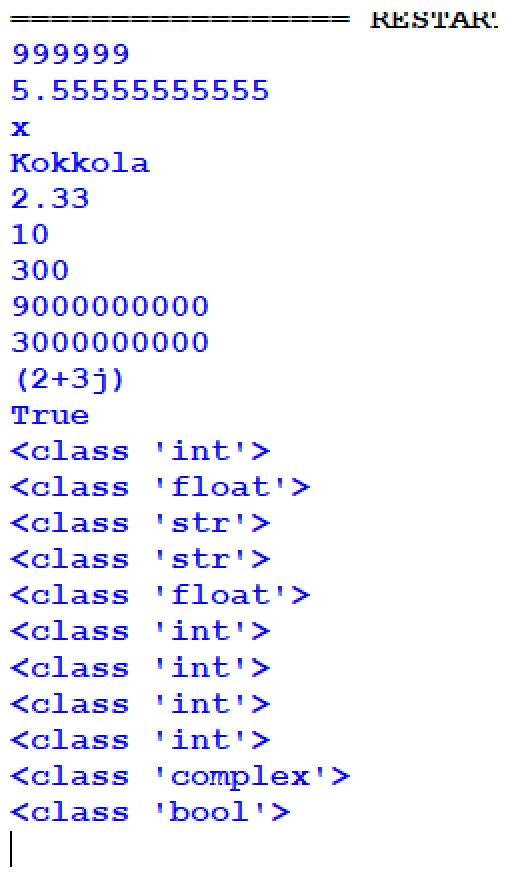
i) 3 billions

2

j) j = 2 + 3j

k) True

```
a = 999999
b = 5.55555555555
c = 'x'
d = "Kokkola"
e = 2.33
f = 10
g = 300
h = 9000000000
i = 3000000000
j = 2 + 3j
k = True


print(a)
print(b)
print(c)
print(d)
print(e)
print(f)
print(g)
print(h)
print(i)
print(j)
print(k)
print(type(a))

print(type(b))

print(type(c))

print(type(d))

print(type(e))

print(type(f))

print(type(g))

print(type(h))

print(type(i))

print(type(j))

print(type(k))
```



```
================= RESTAR
999999
5.55555555555
x
Kokkola
2.33
10
300
9000000000
3000000000
(2+3j)
True
<class 'int'>
<class 'float'>
<class 'str'>
<class 'str'>
<class 'float'>
<class 'int'>
<class 'int'>
<class 'int'>
<class 'int'>
<class 'complex'>
<class 'bool'>
```

2
Our programs uses Ohm's law to calculate the resistance.

User gives voltage and current.

```
print("Ohm Law calculator")
print()
voltage = float(input("Give Voltage: "))
current  = float(input("Give Current: "))

resistance = voltage/current
print(f"Resistance is: {resistance}")
```

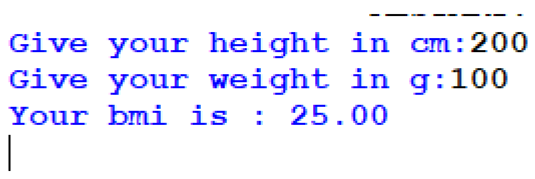
Task 3
User gives the speed of the car (km/h) and the distance (km). Program calculates amount of time.
a) in hours
b) in whole hours and minutes

```
speed = int(input('Enter speed in kmh: '))
distance = int(input('Enter distance in km: '))
dur_hour = distance//speed
dur_mins = int((distance/speed - distance // speed) * 60)
print(f'{dur_hour} hours and {dur_mins} minutes')
whole_hour = distance / speed
print(f'{whole_hour} hours')
```

Task 4
Our program calculates BMI.

```
height = float(input("Give your height in cm:"))
weight = float(input("Give your weight in g:"))
bmi = weight/(height/100)**2
print("Your bmi is : ""{:.2f}".format(bmi))
```

5



```
Give your height in cm:200
Give your weight in g:100
Your bmi is : 25.00
```

Task 5
Create a euro converter: dollars to euros.

```
dollars = int(input('Enter amout of dollars you have: '))

euros = dollars * 0.86
print(f'{dollars} dollars is {euros} euros')
```

Task 6
Convert seconds to hours, minutes, seconds.

```
seconds = int(input('Enter seconds: '))

# First we take seconds
sec = seconds % 60
# Then find out how many minutes there are then we use modulo 60 to
take out minutes thats go over an hour.
minutes = (seconds // 60) % 60
hours = seconds // 3600
print(f'{seconds} is \n{hours} hours {minutes} mins and {sec}
seconds.')
```

Task 7
Convert euros to 5, 10, 20, 50, 100, 200, 500 euros bills.

```
money = int(input('Enter amount of euros you have: '))

# first check amount of 500 bills
bill_500 = money // 500
# extra_money variable is the money left after we have taken the
bills out.
# example we have 900 then we get one 500 bill and extra money is
400 and so on.
extra_money = money % 500
bill_200 = extra_money // 200
extra_money = extra_money % 200
```

```
bill_100 = extra_money // 100
extra_money = extra_money % 100
bill_50 = extra_money // 50
extra_money = extra_money % 50
bill_20 = extra_money // 20
extra_money = extra_money % 20
bill_10 = extra_money // 10
extra_money = extra_money % 10
bill_5 = extra_money // 5

print(f'{money} in euro bills is {bill_500} 500bills, {bill_200}
200bills, {bill_100} 100bills, {bill_50} 50bills, {bill_20} 20bills,
{bill_10} 10bills and {bill_5} 5bills')
```

Task set 2: decision making

Task 8
User gives a value and our program tells if the value is > 100 or not.

```
num = input("Please insert your value: ")

if float(num) > 100:

    print("You value is greater than 100!", "\n")

else:

    print("Your value is NOT greater than 100!", "\n")
```

Task 9
User enters a weekday number and the program tells the name of the day.

```
day = input("Please insert your weekday number, 1 to 7: ")

if int(day) == 1:

    print("Sunday","\n")

elif int(day) == 2:

    print("Monday","\n")

elif int(day) == 3:
    print("Tuesday","\n")
elif int(day) == 4:
    print("Wednesday","\n")
elif int(day) == 5:
    print("Thursday","\n")
elif int(day) == 6:
    print("Friday","\n")
elif int(day) == 7:
    print("Saturday","\n")
else:
  print("Error!, please try again!","\n")
```

Task 10
Program calculates BMI and gives also a textual description.

```
weight = float(input("Insert weight in kg: "))
height = float(input("Insert height in m: "))
BMI = float(weight/(height * height))
format_BMI = "{:.2f}".format(BMI)
FBMI = float(format_BMI)
print("Your BMI value: ", FBMI, " kg/m2", "\n")
if float(FBMI) < 18.5:
    print("You are underweight!", "\n")
elif float(FBMI) >= 18.5 and float(FBMI) < 24.9:
    print("You are in healthy weight range!", "\n")
```

```
elif float(FBMI) >= 24.9 and float(FBMI) < 29.9:

    print("You are overweight!", "\n")

elif float(FBMI) >= 29.9:

    print("You are obese!!!", "\n")

else:

    print("ERROR!!!", "\n")
```

Task 11
User gives a month number and our program tells the number of days in that month.

```
month = input("Please insert your month number, 1 to 12: ")

if int(month) == 1 or int(month) == 3 or int(month) == 5 or
int(month) == 7 or int(month) == 8 or int(month) == 10 or int(month)
== 12:

    print("This month has 31 days!", "\n")

elif int(month) == 4 or int(month) == 6 or int(month) == 9 or
int(month) == 11:

    print("This month has 30 days!", "\n")

elif int(month) == 2:

    print("This month has 28 days in normal year, 29 days in leap
year!", "\n")

else:

    print("ERROR!, please try again!","\n")
```

Task 12
User gives the lengths of the triangle's sides. Program tells what is the triangle like (e.g. is it right angled, isosceles…)

```
a1 = float(input("1. side?"))
```

```
a2 = float(input("2. side?"))

a3 = float(input("3. side?"))


if (a1 == a2 and a1 == a3):

    print("Equilateral\n")

elif (a1 == a2 or a1 == a3 or a2 == a3):

    print("Isosceles\n")

elif (a1**2 + a2**2 == a3**2 or a1**2 + a3**2 == a2**2 or a2**2 +
a3**2 == a1**2):

    print("Right angled\n")

else:

    print("regular")
```

Task 13
Variables a, b and c have different values. Create a program that finds the biggest one.
Show 3 different ways to solve the problem.

```
a1 = float(input("1. value?"))

a2 = float(input("2. value?"))

a3 = float(input("3. value?"))


if (a1 >= a2 and a1 >= a3):

    print("a1\n")

elif (a2 <= a1 and a2 == a3):

    print("a2\n")

else:

    print("a3\n")




if a1 >= a2:
    if a1 >= a3:
        print("a1\n")
elif a2 >= a1:
    if a2 >= a3:
        print("a2\n")
else:
    print("a3\n")



max = a1
if a2 >= max:
    max = a2
if a3 >= max:
    max = a3
print(max)
```

Task 14
Check that given value is bigger than 100.

```
value_input = input("Please enter a value: ")

try:
    num = float(value_input)
    if num > 100:
        print("Value is greater than 100")
    else:
        print("Value is not greater than 100")
```

11

```
except ValueError:
    print("Value is not a number")
```

Task 15
User gives weekday number: program tells the name of that day.

```
weekday_number = input("Please enter a number corresponding to a
weekday (1-7): ")


weekdays = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"]


try:
    index = int(weekday_number) - 1
    if 0 < index < 7:
        day_name = weekdays[index]
        print(f"The number {weekday_number} returns {day_name}")
    else:
        print("Number not in range")
except ValueError:
    print("Input is not a valid number")
```

Task 16

Calculate BMI

```
print("\nWelcome to the BMI-calculator!")


description = ""
weight = float(input("Please enter your weight in kg: "))
height_cm = float(input("Please enter your height in cm: "))
height = height_cm / 100
bmi = weight / (height * height)


if bmi < 18.5:
    description = "You are underweight. \nIt would be healthy to
increase your weight."
```

```
elif bmi < 25:
    description = "Great! Your weight is normal. \nKeep going like this!"
elif bmi < 30:
    description = "You are overweight. \nIt would be healthy to lose weight."
elif bmi < 35:
    description = "You are obese. \nIt would be very important for you to lose weight."
else:
    description = "You are very obese. \nIt is urgent that you lose weight to lower your risk of disease.\nIt is tough, but you can do it!"


print(f"Your calculated BMI is: {round(bmi, 1)}")
print(description)
```

Task 17

Get number of a month and return the number of days it has

```
month_input = input("\nPlease enter the number of a month (1-12): ")
month_day_amount = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
user_choice = ""
index = int(month_input) - 1
if index == 1:
    while user_choice not in ["y", "n"]:
        user_choice = input("You have chosen February, is the year in question a leap year? (y/n)")
    if user_choice == "y":
        number_of_days = month_day_amount[index] + 1
        print(f"The month in question has {number_of_days} days.")
    else:
        number_of_days = month_day_amount[index]
        print(f"The month in question has {number_of_days} days.")

elif 0 <= index < 12:
    number_of_days = month_day_amount[index]
    print(f"The month in question has {number_of_days} days.")
else:
    print("Not a valid month.")
```

Task set 3: loops

Task 18
Program calculates the sum of values 1 - 5.
Use: for and while

```
sum = 0
for i in range(1, 5):
    sum += i          # sum = sum + i
print(sum)


sum = 0


j = 1                              #initialize
while j < 5:                       #condition
    sum = sum + j
    j = j+ 1     #increment
print(sum)
```

Task 19
Program calculates the sum of even numbers between 2 - 40.
Use: for and while

```
sum = 0
for i in range(2, 40):
    if i%2 == 0:
        sum = sum + i
        #print("The even numbers are : ", i)
print("The sum of even numbers : ", sum)


#OR this way
```

```
sum = 0
for i in range(2, 40, 2):
    sum = sum + i

print("The sum of even numbers : ", sum)
```

Task 20
Program calculates sum: 5, 10, 15, .. 100.
Use: for and while

```
sum = 0
for i in range(5, 100, 5):
    sum = sum + i

print("The sum is : ", sum)
```

Task 21
Program throws dice 100 times and tells amounts of different values (1, 2, 3, 4, 5, and 6).
Hints:
from random import randint
# scaling example [0,10]
value = randint(0, 10)

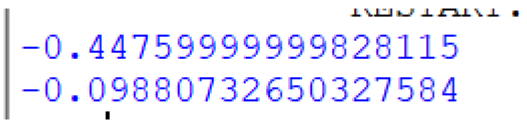
```
from random import randint
# scaling example [0,10]
# 100 times
n1=n2=n3=n4=n5=n6=0

for i in range(1,100):
    value = randint(1, 6)
    if value == 1:
        n1 += 1
    elif value == 2:
        n2 += 1
    elif value == 3:
        n3 += 1
    elif value == 4:
        n4 += 1
    elif value == 5:
        n5 += 1
    elif value == 6:
        n6 += 1
print (n1)
print (n2)
print (n3)
print (n4)
print (n5)
print (n6)
```

Task 22
Account manager with menu:
User can make deposits
Do withdrawal
Check the balance

```
balance = 2000
while True:
    print("1 = add money")
    print("2 = take money")
    print("3 = check balance")
    print("0 = exit")
    val = int(input("your choice?"))
    if val == 1:
        print("how much are you adding?")
        sum = int(input("amount?"))
        balance += sum
        print ("balance is now " + str(balance))
    elif val == 2:
        print("how much are you taking?")
        sum = int(input("amount?"))
        if balance >= sum:
            balance -= sum
        else:
            print ("you can not take so much")

        print ("balance is now " + str(balance))
    elif val == 0:
        break
```

Task 23
Try to solve this equation (try find 1 of roots)
3x^3 - 4x^2 + 9x +5 = 0
Here ^ means exponent

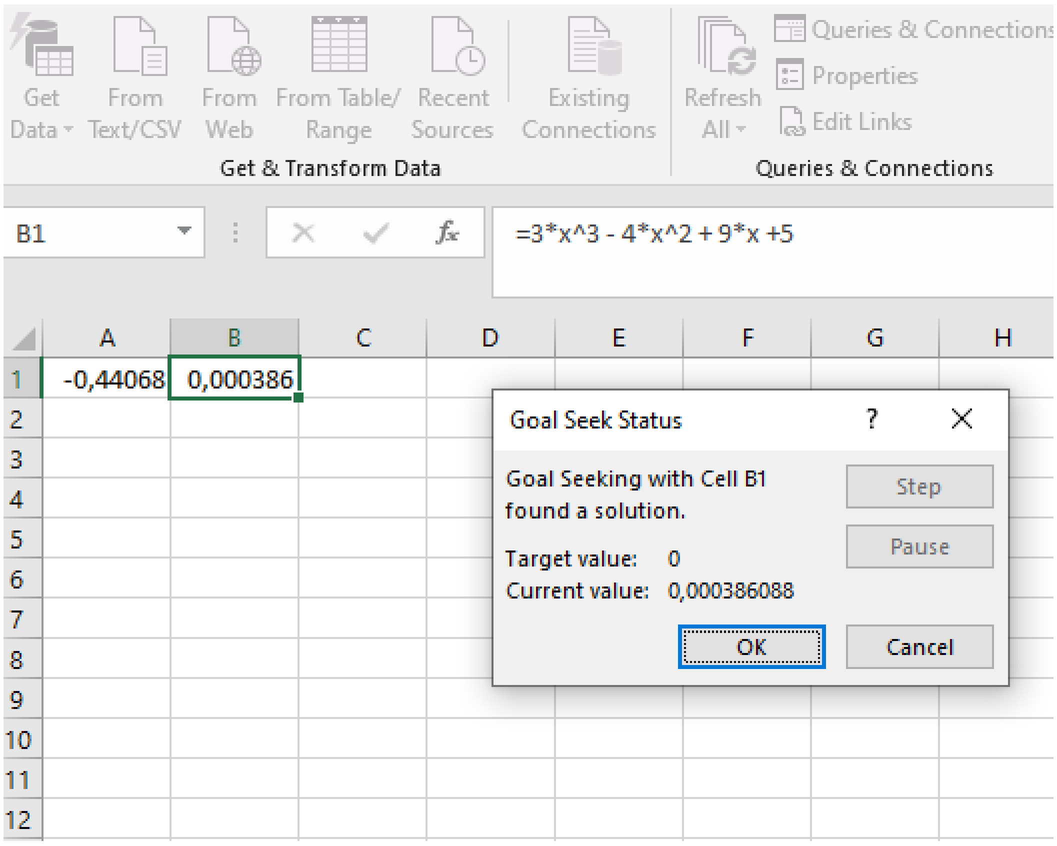
```
x = -100.0
y = 100

step = 0.001
margin = 0.1

while True:
    x += step
    y = 3*x**3 - 4*x**2 + 9*x + 5
    if  abs(y) < margin:
        break

print(x)
print(y)
```



```
-0.44759999999828115
-0.09880732650327584
```

20

Excel result is given here



=3*x^3 - 4*x^2 + 9*x +5

	A	B
1	-0,44068	0,000386

Goal Seek Status

Goal Seeking with Cell B1 found a solution.

Target value: 0

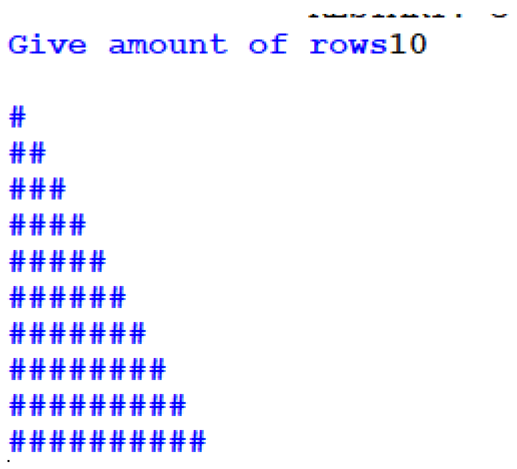
Current value: 0,000386088

Task 24

Print this kind of semipyramid (character amount of rows is given in a variable):

m
mm
mmm
mmmm
mmmmm

```
i = int(input("Give amount of rows"))

q = ""

for k in range(0,i+1):

    for s in range(0,k):

    q += "#"
    print(q)
    q = ""
```



```
Give amount of rows10

#
##
###
####
#####
######
#######
########
#########
##########
```

Task 25
Program calculates the factorial of n (given in a variable)

```
i = int(input("value?"))


s = 1
for k in range(1,i+1):
    s *= k


print(s)
```

Task 26
Program calculates the exponential value (base and exponent are given invariable). Base can be a real number, exponent is a whole number. Use a loop.

```
base = int(input("base?"))
exponent = int(input("exponent"))
```

```
s = 1
for k in range(1,exponent+1):
    s *= base

print(s)
```

Task set 4: arrays (lists)

Task 27

Do this array task.

1. Array contains 30 random values. Calculate the sum and average.

```
from random import randint

from math import sqrt

from time import sleep


def ArraySumAndAvg():

    print("Assign 30 random values to an array and calculate the sum and average.\n")

    rand_array = []

    for i in range(30):

        rand_array.append(randint(-1000, 1000))

    print("The array values are: " + str(rand_array))

    arr_sum = sum(rand_array)

    arr_avg = arr_sum/30

    print("The sum of the array is: " + str(arr_sum) + " and the average is " + str(arr_avg) + ".")

    sleep(3)
```

Task 28

2. Find the maximun of an array.

```
def ArrayMax():

    print("\n" + "-" * 25 + "\nProgram finds the maximum value in an array.\n")

    rand_array = []
    for i in range(30):
        rand_array.append(randint(-1000, 1000))
    print("The array values are: " + str(rand_array))
    arr_max = max(rand_array)
    print("The maximum value of the array is " + str(arr_max) + ".")
    sleep(3)
```

Task 29

3. Search a value from an array.

```
def ArraySearch():
    print("\n" + "-" * 25 + "\nProgram searches a value from an 
array.\n")
    rand_array = []
    search = -1001
    for i in range(30):
        rand_array.append(randint(-1000, 1000))
    print("The array values are: " + str(rand_array))
    while search not in rand_array:
        search = int(input("Give the value to be searched: "))
        if search not in rand_array:
            print("Could not find " + str(search) + " from the 
array.")
    print("Found the value " + str(search) + " from array at index " 
+ str(rand_array.index(search)))
    sleep(3)
```

Task 30

4. Fill 2 arrays with some values and calculate the sum array.

```
def TwoArrays():
    print("\n" + "-" * 25 + "\nProgram adds random values to two
arrays and then calulates their sum.\n")
    arr_one = []
    arr_two = []
    for i in range(randint(4, 8)):
        arr_one.append(randint(randint(-1000, 0) , randint(0,
1000)))
        arr_two.append(randint(randint(-1000, 0) , randint(0,
1000)))
    print("Array 1 is: " + str(arr_one) + ".")
    print("Array 2 is: " + str(arr_two) + ".")
    sum_arr = []
    for i in range(len(arr_one)):
        sum_arr.append(arr_one[i]+arr_two[i])
    print("The sum of these two arrays is " + str(sum_arr))
    sleep(3)
```

Task 31

Generate a lottorow (try to use an array here).

```
def Lottorow():
    print("\n" + "-" * 25 + "\nProgram generates a random lotto row.
\n")
    lottorow = []
    n = 0
    while n < 7:
        new_number = randint(0, 39) + 1
        if new_number not in lottorow:
            lottorow.append(new_number)
            n += 1
    lottorow.sort()
    print("Generated lotto row is: " + str(lottorow) + ".")
    sleep(3)
```

Task 32

Take a look at python.org site.
Array methods are presented here:
https://docs.python.org/3/tutorial/datastructures.html

Give your own examples of using those metods.

```
def ListExamples():
    print("\n" + "-" * 25 + "\nProgram gives examples of different 
python list methods.\n")
    arr_one = [3, 5, 12]
    arr_two = [6, 22, 4]
    print(str(arr_one) + "\n" + str(arr_two))
    #append
    arr_one.append(88)
    print("Append 88 to arr_one: " + str(arr_one))
    sleep(1)
    #extend
    arr_two.extend([2, 1, 0])
    print("Extend arr_one by adding 2, 1, 0: " + str(arr_two))
    sleep(1)
    #insert
    arr_one.insert(1, 9)
    print("Insert value 9 at index 1 to arr_one: " + str(arr_one))
    sleep(1)
    #remove
    arr_two.remove(0)
    print("Remove value 0 from arr_two: " + str(arr_two))
    sleep(1)
    #pop
    pop = arr_one.pop(0)
    print("Pop from arr_one item at index 0: " + str(pop))
    sleep(1)
    #index
    ind = arr_two.index(22)
    print("Return the index of value 22 at arr_two: " + str(ind))
    sleep(1)
    #count
    count = arr_one.count(9)
    print("Return the times value 9 appears in arr_one: " + 
str(count))
    sleep(1)
    #sort
    arr_two.sort()
    print("Sort arr_two: " + str(arr_two))
    sleep(1)
    #reverse
    arr_one.reverse()
    print("Reverse arr_one: " + str(arr_one))
    sleep(1)
    #copy
    arr_three = arr_two.copy()
    print("Copy arr_two to arr_three: " + str(arr_three))
    sleep(1)
    #clear
    arr_three.clear()
    print("Clear arr_three: " + str(arr_three))
    sleep(2)
```

Task 33

Create a short dictionary: e.g Finnish to English.
Add some wordpairs to a list.
Or as example, here are some English – Italian words:

treaty	**patto**
truck	**camion**
trust	**trust**

```
def Dictionary():

    print("\n" + "-" * 25 + "\nA tiny Finnish-English dictionary.
\n")

    fi_en_dict = {"aamu": "morning",

                  "auto": "car",

                  "arka": "shy",

                  "elokuva": "movie",

                  "hallitus": "government",

                  "hirviö": "monster",

                  "ilta": "evening",

                  "jää": "ice",

                  "maailma": "world",

                  "olut": "beer"}

    print(str(fi_en_dict))

    sleep(3)
```

Task 34

There are 20 values in an array:
calculate the standard deviation

```
def StandardDeviation():

    print("\n" + "-" * 25 + "\nProgram calculates the standard
deviation of 20 random values.\n")

    values = []

    for i in range(20):
        values.append(randint(1, 10))
    print("Array: " + str(values))
    avg_values = sum(values)/len(values)
    print("Average: " + str(avg_values))
    dev_values = []
    for i in range(len(values)):
        dev_values.append((values[i]-avg_values)**2)
    print("Value difference to mean: " + str(dev_values))
    sd = sqrt(sum(dev_values)/len(values))
    print("The standard deviation of the array is " + str(sd) + ".")
    sleep(4)
```

Task 35

2 arrays contain students grades in math and in English language. There are 10 students. Try to calculate the correlation.

```
def GradeCorrelation():
    print("\n" + "-" * 25 + "\nProgram calculates the correlation
between grades (1-5).\n")
    math_grades = []
    eng_grades = []
    for i in range(10):
        math_grades.append(randint(1, 5))
        eng_grades.append(randint(1, 5))
    print("Student math grades: " + str(math_grades))
    print("Student english grades: " + str(eng_grades))
    avg_math = sum(math_grades) / len(math_grades)
    avg_eng = sum(eng_grades) / len(eng_grades)
    print("Average grade for math: " + str(avg_math) + " and for
english: " + str(avg_eng))
    dev_math = []
    dev_eng = []
    for i in range(10):
        dev_math.append((math_grades[i]-avg_math)**2)
        dev_eng.append((eng_grades[i] - avg_eng)**2)
    print("Math grade difference to mean: " + str(dev_math) + ".
\nEnglish grade difference to mean: " + str(dev_eng) + ".")
    math_x_eng = []
    math_sqr = []
    eng_sqr = []
    for i in range(10):
        math_x_eng.append(dev_math[i]*dev_eng[i])
        math_sqr.append(dev_math[i]**2)
        eng_sqr.append(dev_eng[i]**2)
    math_eng_corr = sum(math_x_eng) /
sqrt(sum(math_sqr)*sum(eng_sqr))
    print("The correlation between math and english grades is: " +
str(math_eng_corr))


To previous array tasks function calls here
def main():
    ArraySumAndAvg()
    ArrayMax()
    ArraySearch()
    TwoArrays()
    Lottorow()
    ListExamples()
    Dictionary()
    StandardDeviation()
    GradeCorrelation()
main()
```

Task 36

Create function: Returns the average of 2 integers

```
def twoNumberAverage(a, b):
    return ((a + b)/2)
```

```
# testing

a, b = map(int, input("Please enter two numbers: ").split())
print(" Average of %d + %d is : %.1f" % (a, b, twoNumberAverage(a, b)))
```

Task 37

Create function: Returns the average of 4 floating point values.

```
def fourFloatAverage(a, b, c, d):
    return ((a + b + c + d)/4)
```

```
a, b, c, d = map(float, input("Enter four decimal numbers (2 decimal accuracy): ").split())

print("The average of %.2f + %.2f + %.2f + %.2f is %.2f" % (a, b, c, d, fourFloatAverage(a, b, c, d)))
```

Task 38

Create function: Returns the sum of an array.

```
def arraySum(array):
    sum = 0
    for x in array:
        sum+= x
    return sum
```

numbersA = list(map(int, input("Enter numbers into the array: ").split()))

#test

print("The sum for array ", numbersA, "is %d" % arraySum(numbersA))

Task 39

Create function: Returns the factorial.

```
def factorial(number):
    fact = 1
    for i in range(1,number+1):
        fact = fact * i
    return(fact)
```

#testing

number = int(input("Enter a number to find out its factorial: "))

print("The factorial of %d is %d" % (number, factorial(number)))

Task 40

Create function: Returns the biggest of 3 integers.

```
def largestNumber(array):
    maximum = -5000
    for i in array:
        if i > maximum:
            maximum=i
    return maximum
```

numbersA = list(map(int, input("Enter three numbers: ").split()[:3]))

#test

print ("The largest number of %s is %d" % (numbersA, largestNumber(numbersA)))

Task 41

Create function: Returns the BMI.

```
def BMI(weight, height):
    BMI = weight / (height**2)
    return BMI

weight = float(input("Enter your weight in Kilograms: "))
height = float(input("Enter your height in meters: "))
```

#test

print("Weight is %dKg and the height is %.2fm, so the BMI is : %.1f" % (weight, height, BMI(weight, height)))

Task 42

Function returns the biggest of 5 integers.

```
def largestNumber(array):
    maximum = -5000
    for i in array:
        if i > maximum:
            maximum=i
    return maximum
```

```
numbersA = list(map(int, input("Enter five numbers: ").split()[:5]))

#test

print ("The largest number of %s is %d" % (numbersA, largestNumber(numbersA)))
```

Task 43

Calculates amount of combinations (try to use also an own factorial function here).

```
#factorial function
def factorial(number):
    fact = 1
    for i in range(1,number+1):
        fact = fact * i
    return(fact)

# combinations function

def factorial(number):
    fact = 1
    for i in range(1,number+1):
        fact = fact * i
    return(fact)



def combination(objects, sample):
    combinations = factorial(objects) / (factorial(sample) *
factorial((objects - sample)))
    return combinations


people = int(input("How many people are there: "))
chairs = int(input("How many chairs are there: "))


print(combination(people, chairs))
#test
print(combination(people, chairs))
```

Task 44

Create function: Calculates the standard deviation.

```
def stddiv(array):
    mean = average(array)
    sd = 0.0
    for x in array:
        sd += (float(x) - mean)**2
    sd = (sd / float((len(array) - 1)))**0.5
    return sd


def average(array):
    sum = 0.0
    for x in array:
            sum+= x
    mean = sum / len(array)
    return mean
```

```
array1 = list(map(int, input("Enter numbers into the array: 
").split()))


#test

print("Standard deviation of ", array1, " is : %.3f" % 
stddiv(array1))
```

Task 45

Create function: Searches for a value from an array.

```
def searchValue(value, array):

    if value in array:

        return "Found"

    else:

        return "Not Found"


List = ["Black", "Blue", "Green", "Yellow", "Red", "White", 
"Orange", "Blue"]

search = input("Type a colour: ")


print("in array : ", List, "%s was %s" % (search, 
searchValue(search, List)))
```

Task 46

Create function: Calculates the square root of value 2 (create your own function).

```
def squroot(a, b):

    return (a + b)**(1/2)


a, b = map(int, input("Please enter two numbers: ").split())
```

```
#test
print("The squareroot of %d + %d is %.2f" % (a, b, squroot(a, b)))
```

Task 47

Create function: Calculates an approximation of Neper's value (e).

$$e = \sum_{k=0}^{\infty} (1/k!)$$

```
def factorial(number):
    fact = 1
    for i in range(1,number+1):
        fact = fact * i
    return(fact)

def Neper(k):
    sum = 1
    for x in range(int(k), 0, -1):
        sum += 1 / float(factorial(x))
    return sum

number = int(input("Enter the number of iterations to estimate
\"e\": "))

print(Neper(number))
```

Task 48

Create function: Calculates approximations of sin(x) and cos(x)

$$\cos(x) = 1 - x^2/2! + x^4/4! - \ldots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

```
import math
def factorial(number):
    fact = 1
    for i in range(1,number+1):
        fact = fact * i
    return(fact)

def cos(x):
    x = math.radians(x)
    sum = 0
    for i in range(10):
        coef = (-1)**i
        num = x**(2*i)
        denom = factorial(2*i)
        sum += ( coef ) * ( (num)/(denom) )
    return sum

def sine(x):
    x = math.radians(x)
    sum = 0
    for i in range(10):
        coef = (-1)**i
        num = x**(2*i + 1)

        denom = factorial(2*i + 1)

        sum += ( coef ) * ( (num)/(denom) )

    return sum


#Test

angle = float(input("Enter an angle in degrees: "))

print("Cos(%.2f) is %.3f" % (angle, cos(angle)))

print("Sine(%.2f) is %.3f" % (angle, sine(angle)))
```

Task 49

Create a function that sorts an array by using selection sort.

```
def sort(array):

    for i in range(len(array)):

        min = i

        for j in range(i+1, len(array)):

            if array[min] > array[j]:

                min = j

        array[i], array[min] = array[min], array[i]


numbers = list(map(int, input("Enter numbers into the array: 
").split()))


#test

print("Unsorted array is ", numbers)

sort(numbers)

print("The sorted array is ", numbers)
```

Task 50

Create a function that multiplies two arrays.

```
def multiply_arrays(array1, array2):
    answer = [0] * len(array1)
    if len(array1) != len(array2):
        return " Array lenghts do not match!"
    else:
        for i in range(len(array1)):
            answer[i] = array1[i] * array2[i]
        return answer


number1 = list(map(int, input("Enter five numbers into the first
array: ").split()[:5]))
number2 = list(map(int, input("Enter five numbers into the second
array: ").split()[:5]))


#test
print("Array 1 is ", number1)
print("Array 2 is ", number2)
print("The multipled array is : ", multiply_arrays(number1,
number2))
```

Task set 5: GUI

Task 50: math game

Create a kids math game (add prize: pic, sound or something else (negative/positive))
Two numbers are shown on labels
User adds the sum to a textbox
With a button sum is checked
Then new values are generated to labels
Right and wrong answers are shown

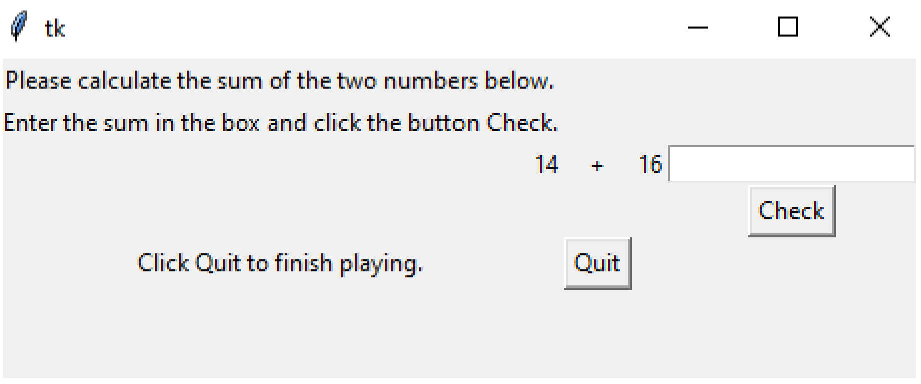
```
import tkinter
import random
def checkValue():
    sum = 0
    number = int(box_value.get()) #reads the user's input
    nrs1 = int(label_1["text"]) #reads the numbers in the labels
    nrs2 = int(label_2["text"])
    sum = nrs1 + nrs2
    if number == sum: #if the user's answer was correct
        result_label["text"] = "Correct. " +str(nrs1)+ " + "
+str(nrs2)+ " = " + str(sum) #show the calculation and answer
        label_1["text"] = "" #clear the numbers in the labels
        label_2["text"] = ""
        label_plus["text"] = "" #clear the plus sign
        path = "pic1.gif" #show the price photo
        photo["file"] = path
        label_3["image"] = photo

    else:
        result_label["text"] = "Sorry, incorrect. " +str(nrs1)+ " +
" +str(nrs2)+ " = " + str(sum)
        label_1["text"] = ""
        label_2["text"] = ""
        label_plus["text"] = ""
        path = "pic2.gif" #show the negative photo
        photo["file"] = path
        label_3["image"] = photo
    label_direction2["text"] = "Please press enter if you want to
try again."
```

```
def enter(event):
    nr1 = random.randint(1,21) #new random numbers
    nr2 = random.randint(1,21)
    label_1["text"] = nr1 #new numbers assigned to labels
    label_2["text"] = nr2
    label_plus["text"] = "+"
    box_value.delete(0, 100) #empties the entry box, characters btw
indeces 0-100
    result_label["text"] = "" #clears the result info
    label_direction2["text"] = "" #clears the "press enter for new
numbers" info

window = tkinter.Tk()

window.rowconfigure(7, minsize = 50, weight = 1)
window.columnconfigure([0,1,2,3,4,5], minsize = 5, weight = 1)

label_direction = tkinter.Label(master = window, text = "Please
calculate the sum of the two numbers below.")
label_direction.grid(row = 1, column = 1)

nr1 = random.randint(1,21)
nr2 = random.randint(1,21)

label_1 = tkinter.Label(master = window, text = nr1)
label_1.grid(row = 3, column = 1, sticky = "e")

label_2 = tkinter.Label(master = window, text = nr2)
label_2.grid(row = 3, column = 3, sticky = "w")
```

```
label_plus = tkinter.Label(master = window, text ="+")

label_plus.grid(row = 3, column = 2)


label_direction3 = tkinter.Label(master = window, text = "Enter the
sum in the box and click the button Check.")

label_direction3.grid(row = 2, column = 1)


box_value = tkinter.Entry(master = window, fg = "black", bg =
"white")

box_value.grid(row = 3, column = 4,sticky = "e")


label_direction2 = tkinter.Label(master = window, text = "")

label_direction2.grid(row = 4, column = 1)


result_label = tkinter.Label(master = window, text = "")

result_label.grid(row = 4, column = 2)


button1 = tkinter.Button(master = window, text = "Check", command =
checkValue)

button1.grid(row = 4, column = 4)


path = ""

photo = tkinter.PhotoImage(file = path)

label_3 = tkinter.Label(master = window, image=photo)

label_3.grid(row = 7, column = 1, columnspan = 6)


window.bind("<Return>", enter)#if user presses the enter key, go to
event "enter"
```

```
label_direction4 = tkinter.Label(master = window, text = "Click Quit
to finish playing.")

label_direction4.grid(row = 6, column = 1)


button3 = tkinter.Button(master = window, text = "Quit", command =
quit)

button3.grid(row = 6, column = 2)


window.mainloop() #runs the "window" loop
```

Test run



Please calculate the sum of the two numbers below.

Enter the sum in the box and click the button Check.

14 + 16

Click Quit to finish playing.

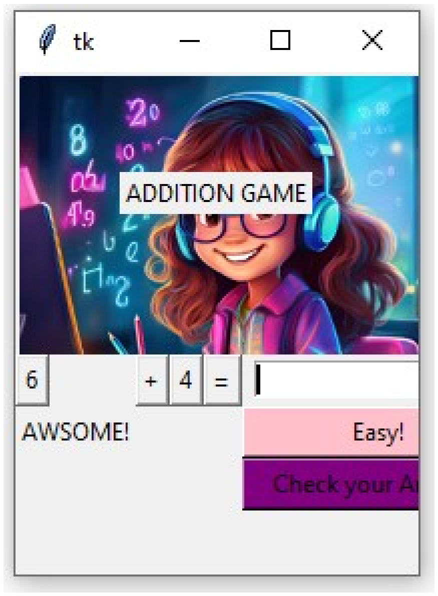
Task 52

Kids math game, simple version 2

```
import random

import os

from tkinter import *


img_dir = os.getcwd()

print(img_dir)
```

```
root = Tk()
root.geometry("201x251")
bg = PhotoImage(file = f'{img_dir}\image3.png')

# Show image using label
label1 = Label( root, image = bg)
label1.place(x = 0, y = 0)
label2 = Label( root, text = "ADDITION GAME")
label2.pack(pady = 50)

frameB= Frame(root)
frameB.pack(pady = 20)

#initiate two variables for calculation and operator(could be + and
- or others)
x = IntVar()
y = IntVar()
operator = StringVar()
#BUTTONS for variables
n1=Button(frameB,textvariable = x).grid(row = 0,column = 0,sticky=W)
o1=Button(frameB,textvariable = operator,width = 1).grid(row =
0,column = 1,sticky=W)
n2=Button(frameB,textvariable = y).grid(row = 0,column = 2,sticky=W)
o2=Button(frameB,text = '=').grid(row = 0,column = 3,sticky=W)
e = Entry(frameB)
e.grid(row = 0, column = 4)

def newGame():
    x.set(random.randint(5,10))
    y.set(random.randint(0,5))
    operator.set('+')


Button(frameB,text = 'Easy!', width = 18,height = 1,background =
'pink', command = newGame).grid(row = 1,column = 4,sticky=W)


result = StringVar()

def checkResult(event):

    c = str(x.get())+operator.get()+str(y.get())

    if len(e.get()) !=0:

        if int(e.get()) == eval(c):

            newGame()

            result.set("AWSOME!")

            e.delete(0,'end')

        else:

            result.set("OOOPS!")

            e.delete(0, 'end')

    else:

        result.set("STILL WAITING...")


root.bind('<Return>', checkResult)

button2 = Button(frameB,text="Check your Answer!",width = 18,height
= 1,background = 'purple')

button2.grid(row = 2,column = 4,sticky=W)

button2.bind('<Button-1>',checkResult)

Label(frameB,textvariable =result).grid(row = 1,column = 0,sticky=W)


root.mainloop()
```

Test run



Tasks 53

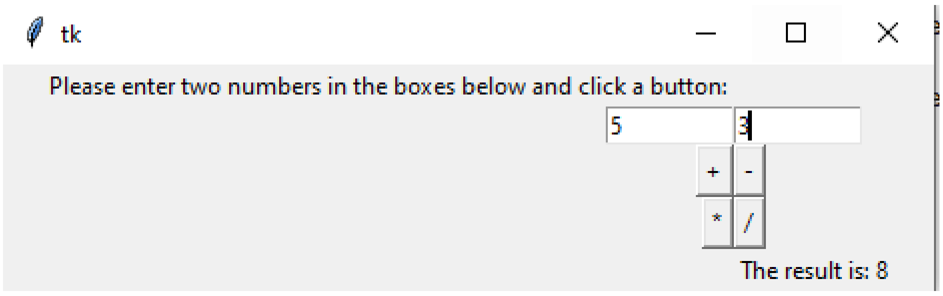
Create a mini calculator

```
import tkinter

def calculateSum():
    number1 = int(value1.get())
    number2 = int(value2.get())
    sum = number1 + number2
    label_value["text"] = "The result is: " + str(sum)

def calculateSubtraction():
    number1 = int(value1.get())
    number2 = int(value2.get())
    subt = number1 - number2
    label_value["text"] = "The result is: " + str(subt)
```

```
def calculateMultiplication():
    number1 = int(value1.get())
    number2 = int(value2.get())
    mult = number1 * number2
    label_value["text"] = "The result is: " + str(mult)

def calculateDivision():
    number1 = int(value1.get())
    number2 = int(value2.get())
    if number2 != 0:
        div = number1 / number2
        label_value["text"] = "The result is: " + str(div)
    else:
        label_value["text"] = "ERROR"


window = tkinter.Tk()

#creating the rows and columns where the calculator will be placed,
in "window":
window.rowconfigure(5, minsize = 10, weight = 1)
window.columnconfigure([0,1,2,3], minsize = 20, weight = 1)

#creating the entry boxes where user enters values, and variables
that store the info user enters:
value1 = tkinter.Entry(master = window, fg = "black", bg = "white",
width = 10) #black font on white background
value1.grid(row = 2, column = 1, sticky = "e") #where entry box will
be placed in the grid, and in the "east" of the area
```

```
value2 = tkinter.Entry(master = window, fg = "black", bg = "white",
width = 10)

value2.grid(row = 2, column = 2, sticky = "w")


#creating buttons:

button_sum = tkinter.Button(master = window, text = "+", command =
calculateSum) #command is the function called when this is clicked

button_sum.grid(row = 3, column = 1, sticky = "e")


button_sum = tkinter.Button(master = window, text = "-", command =
calculateSubtraction)

button_sum.grid(row = 3, column = 2, sticky = "w")


button_sum = tkinter.Button(master = window, text = "*", command =
calculateMultiplication)

button_sum.grid(row = 4, column = 1, sticky = "e")


button_sum = tkinter.Button(master = window, text = "/", command =
calculateDivision)

button_sum.grid(row = 4, column = 2, sticky = "w")


#creating a label, showing the result, displayed after clicking a
button:

label_value = tkinter.Label(master = window, text = "")

label_value.grid(row = 5, column = 2)


#creating a label displaying directions at the top of the page:

label1_value = tkinter.Label(master = window, text = "Please enter
two numbers in the boxes below and click a button:")

label1_value.grid(row = 1, column = 1)


window.mainloop()
```

Test run



Please enter two numbers in the boxes below and click a button:

5 3

The result is: 8

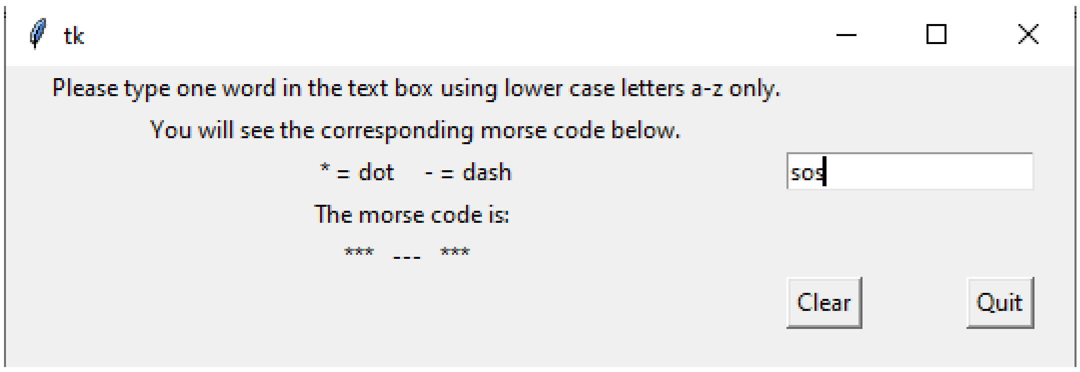
Tasks 54

Create a morse coder

```
import tkinter

window1 = tkinter.Tk()

def key(event):
    char1 = repr(event.char) #the pressed key is read into char1
    if char1 == "'a'":
        label_3["text"] += "*-   " #the morse code of 'a' and three
spaces after it
    elif char1 == "'b'":
        label_3["text"] += "-***   "
    elif char1 == "'c'":
        label_3["text"] += "-*-*   "
    elif char1 == "'d'":
        label_3["text"] += "-**   "
    elif char1 == "'e'":
        label_3["text"] += "*   "
    elif char1 == "'f'":
        label_3["text"] += "**-*   "
    elif char1 == "'g'":
        label_3["text"] += "--*   "
    elif char1 == "'h'":
        label_3["text"] += "****   "
    elif char1 == "'i'":
        label_3["text"] += "**   "
    elif char1 == "'j'":
        label_3["text"] += "*---   "
    elif char1 == "'k'":
        label_3["text"] += "-*-   "
    elif char1 == "'l'":
        label_3["text"] += "*-**   "
    elif char1 == "'m'":
        label_3["text"] += "--   "
    elif char1 == "'n'":
        label_3["text"] += "-*   "
    elif char1 == "'o'":
        label_3["text"] += "---   "
    elif char1 == "'p'":
        label_3["text"] += "*--*   "
    elif char1 == "'q'":
        label_3["text"] += "--*-   "
    elif char1 == "'r'":
        label_3["text"] += "*-*   "
    elif char1 == "'s'":
        label_3["text"] += "***   "
    elif char1 == "'t'":
        label_3["text"] += "-   "
    elif char1 == "'u'":
        label_3["text"] += "**-   "
    elif char1 == "'v'":
        label_3["text"] += "***-   "
    elif char1 == "'w'":
        label_3["text"] += "*--   "
    elif char1 == "'x'":
        label_3["text"] += "-**-   "
    elif char1 == "'y'":
        label_3["text"] += "-*--   "
    elif char1 == "'z'":
        label_3["text"] += "--**   "
    else:
        label_3["text"] += "ERROR   "

def clearText():
    textbox_value.delete(0, 100)
    label_3["text"] = ""

window1.rowconfigure(8, minsize = 20, weight = 1)
window1.columnconfigure([0,1,2,3], minsize = 20, weight = 1)

label_1 = tkinter.Label(master = window1, text = "Please type one word in the text box using lower case letters a-z only.")
label_1.grid(row = 2, column = 1)

label_2 = tkinter.Label(master = window1, text = "You will see the corresponding morse code below.")
label_2.grid(row = 3, column = 1)

label_4 = tkinter.Label(master = window1, text = "* = dot     - = dash")
label_4.grid(row = 4, column = 1)
```

```
textbox_value = tkinter.Entry(master = window1, fg = "black", bg =
"white")

textbox_value.grid(row = 4, column = 2, sticky = "w")


label_4 = tkinter.Label(master = window1, text = "The morse code is:
")

label_4.grid(row = 5, column = 1)


label_3 = tkinter.Label(master = window1, text = "")

label_3.grid(row = 6, column = 1)


button_clear = tkinter.Button(master = window1, text = "Clear",
command = clearText)

button_clear.grid(row = 7, column = 2, sticky = "w")


button_quit = tkinter.Button(master = window1, text = "Quit",
command = quit)

button_quit.grid(row = 7, column = 2, sticky = "e")


window1.bind("<Key>", key) #binds pressing the key by the user to
the event


window1.mainloop()
```

Test run



Please type one word in the text box using lower case letters a-z only.

You will see the corresponding morse code below.

* = dot   - = dash

sos

The morse code is:

```
*** --- ***
```

Tasks 55

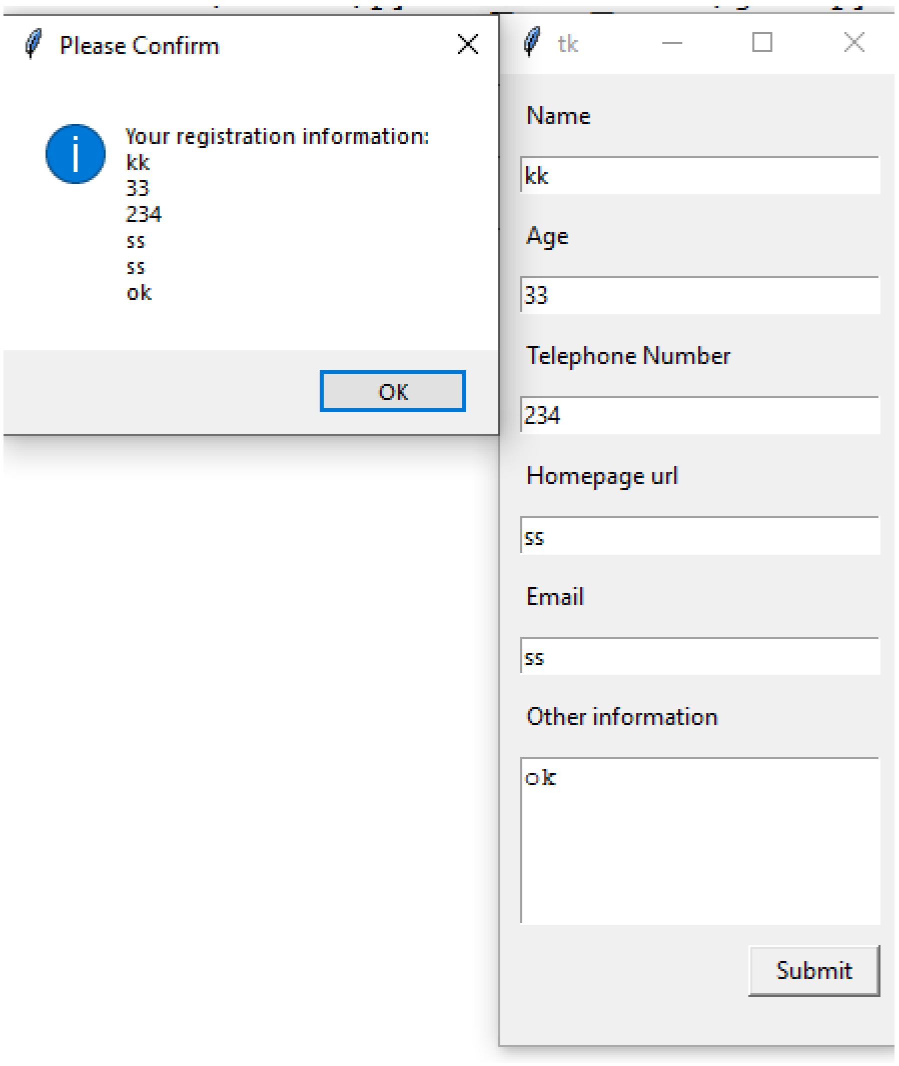
Check 4 different fields of a feedback form before it can be submitted.

E.g user has to give age, telephone number, homepage url, email and/or other information and those values are checked (contents cannot be empty, either).

```
from tkinter import *

import tkinter.messagebox


root = Tk()

root.geometry('200x500')


label1=Label(root,text="Name")

label1.pack(anchor=W,padx=10,pady=10)

entry1 = Entry(root, width=180)

entry1.pack(anchor=W, padx=10,)


label2=Label(root,text="Age")

label2.pack(anchor=W,padx=10,pady=10)

entry2 = Entry(root, width=180)

entry2.pack(anchor=W, padx=10,)


label3 = Label(root, text="Telephone Number")

label3.pack(anchor=W, padx=10, pady=10)

entry3 = Entry(root, width=180)

entry3.pack(anchor=W, padx=10,)


label4 = Label(root, text="Homepage url")

label4.pack(anchor=W, padx=10, pady=10)

entry4 = Entry(root, width=180)

entry4.pack(anchor=W, padx=10,)
```

```
label5 = Label(root, text="Email")

label5.pack(anchor=W, padx=10, pady=10)

entry5 = Entry(root, width=180)

entry5.pack(anchor=W, padx=10,)

label6 = Label(root, text="Other information")

label6.pack(anchor=W, padx=10, pady=10)


text = Text(root, width=180, height=5)

text.pack(anchor=W, padx=10,)


def send_info():

    entry_text1=entry1.get()

    entry_text2=entry2.get()

    entry_text3=entry3.get()

    entry_text4=entry4.get()

    entry_text5=entry5.get()

    text_text = text.get('1.0',END)


    if entry1.get()=='' or entry2.get()=='' or entry3.get()=='' or
entry4.get()=='' or entry5.get()=='':

        tkinter.messagebox.showinfo('Huom!','NOT allow empty
input!')

    else:

        choice=tkinter.messagebox.askokcancel('','Send your
information?')

        if choice:

            tkinter.messagebox.showinfo('Please Confirm', 'Your
registration information: '+ '\n'+entry_text1 + '\n' +entry_text2+
'\n' +entry_text3+ '\n' +entry_text4+ '\n' +entry_text5 + '\n'
+text_text)

        else:
            tkinter.messagebox.showinfo('','You have cancelled your
registration.')



button = Button(root, text="Submit", width=8, height=1,
command=send_info)

button.pack(anchor=E, padx=10,pady=10)


root.mainloop()
```

Test run



```
Please Confirm

Your registration information:
kk
33
234
ss
ss
ok
```

Name: kk
Age: 33
Telephone Number: 234
Homepage url: ss
Email: ss
Other information: ok

Task set 6: dictionary

Task 56

Create a Finnish - English - Finnish dictionary with some 50 word pairs…

The list that we use now has 100 word pairs and the link is https://randomfinnishlesson.blogspot.com/2014/02/100-very-common-finnish-words.html

```
# Raw data
word_list = """
aika - time, quite
aina - always
antaa - to give
asia - thing, matter
ehkä - maybe
ei koskaan - never
ei kukaan - nobody
ei mikään - nothing
eli - so, in other words
ennen - before
ensi - next
ensin - at first
eri - different
että - that
heti - immediately
huono - bad
hyvä - good
itse - self
ja - and
```

jo -	already
joka -	which, every
joku -	someone
jopa -	even
jos -	if
joskus -	sometimes
jossa -	in which
joten -	so, therefore
jotka -	which (plural)
jälkeen -	after
kaikki -	all, everybody
kaupunki -	a town, a city
kanssa -	with
kello -	a clock
kertoa -	to tell
koko -	whole, all
koska -	because, when
koti -	home
kuin -	than
kuinka -	how
kuitenkin -	however
kun -	when
kuva -	picture
kyllä -	yes, indeed
käydä -	to go, to visit
maa -	a country, a land
mennä -	to go
mies -	a man, a husband
mikä -	what
miksi -	why

miten - how	
monta - many	
mukaan - with, according to	
mutta - but	
muu - other, else	
myös - also, too	
nainen - a woman	
niin - so, like that	
noin - like that, approximately	
nyt - now	
nähdä - to see	
näin - like this, I saw	
nämä - these	
oikea - real, right, correct	
olla - to be	
paitsi - except	
paljon - a lot, much	
pitää - to like, to have to, to keep	
pois - away	
puoli - half, side	
päivä - day	
saada - to get, to receive	
sama - same	
sanoa - to say	
se - it	
siellä - over there	
siinä - in there	
silloin - then	
sillä - because	
sitten - then, when, ago, in that case	

```
taas - again
tai - or
takaisin - back
tehdä - to do, to make
tila - space
tuo - that (something you can point at)
tulla - to come
tämä - this
tässä - here
vaan - but
vai - or
vaikka - although, for example
vain - only
vielä - yet, still, furthermore
viime - last
voida - to be able to
vuosi - a year
vähän - a little
väärä - wrong, false
yli - over, past
älä - don't
"""

# Convert raw data into dictionary
finnish_to_english = {}
for line in word_list.strip().split("\n"):
    finnish, english = line.split(" - ", 1)
    finnish_to_english[finnish.strip()] = english.strip()

# Function to search for English word
```

```
def search_english(finnish_word):
    return finnish_to_english.get(finnish_word, "Word not found")



# Print the dictionary
print(finnish_to_english)

# Example search
finnish_word = input("enter finnish word")
```

Tasks 7: Bitwise operators

Task 57
Create a program that uses all bit operators that are shown in the table below.

So, create 2 integer variables. Assign values and test AND, OR and XOR.
Then try shift operators with one variable.
Print also results.

Here are bitwise operators

Operator	Meaning
&	AND
|	OR
<<	Left shift
>>	Right shift
~	One's complement
^	XOR

Solution

```
    a = 199 #1100 0111
    b = 222 #1101 1110
    c = 0
```

```
# AND  &
"""
11000111
11011110
11000110    => 198
"""
c = a & b
print("a & b is " + str(c));


#  OR  |

a = 199 #1100 0111
b = 222 #1101 1110
c = 0

# |  OR
"""
11000111
11011110
1101 1111   => 223
"""
c = a | b
print("a | b is " + str(c));


// XOR ^

/*

11000111

11011110

00011001   => 25

*/

c = a ^ b;

printf("a ^ b is ", str(c));


a = 199 #1100 0111
b = 222 #1101 1110
c = 0

# <<
```

```
    c = a << 2
    print("a « 2 is " + str(c));



   // shiftvalue of variable a  once to the right   a >> 1

a = 199 #1100 0111
b = 222 #1101 1110
c = 0

# <<

c = a >> 1
print("a >> 1 is " + str(c));;
```

Task 58

Check the state of given bit in a bit queue

Tips: Right shift the original bit queue until the bit that has to be inverted is the first bit. Then take bitwise AND between 1 and shifted bit queue. You get the state of the wanted bit.

Solution

```
We have value 155 in a variable. As bits it is 10011011.

We want to know the 3. bit's state.  (LSB s now position 0).

So we shift 155 3 times to the right and get  00010011.
Then we take AND between that new bit queue and value 1 and we get
0000 0001
that tells that state is 1.


a = 155  # 10011011

n = 3;

state = (a >> n) & 1;

print("state is %d \n", str(state));
```

Task 59

Invert the given bit in a bit queue.

Tips: Create a bit mask that has bits 0 and where value 1 has the same position than the bit that is to b inverted. Then take Xor between the mask and the original bit queue. The result is a new bit queue where wanted bit is inverted....

Solution

```
a = 155

n = 4;

mask = 1 << (n - 1);

a = a ^ mask;

print("a is now ", str(a));
```

Tasks 8: OOP

Task 60

Create class Clock and it's subclass AlarmClock. Test clocks in main. There has to be ticking and alarming methods...

```
import datetime
import time

class Clock:
    def getTime(self):
        current_time = datetime.datetime.now().strftime("%H:%M:
%S")
        time_now = current_time.split(":")

        return time_now

    def keepThiking(self, number):
        result = ""
        if number % 2:
            result = "Tic!"
        else:
            result = "Tac!"
            return result


class Alarm(Clock):
    def __init__(self,alarm_h, alarm_m, alarm_s):
            self.alarm_h = alarm_h
            self.alarm_m = alarm_m
            self.alarm_s = alarm_s

    def getAlarm(self):
            return print(self.alarm_h, self.alarm_m, self.alarm_s)

    def triggerAlarm(self):
            result = "Wakie Wakies!!!"
            return result

def main():
    myTime = Clock()
    myAlarm = Alarm(23, 26, 10)
    myAlarm.getTime()

    while((myAlarm.alarm_h != int(myTime.getTime()[0])) or 
(myAlarm.alarm_m != int(myTime.getTime()[1])) or (myAlarm.alarm_s >= 
int(myTime.getTime()[2]))):
            hours = int(myTime.getTime()[0])
            minutes = int(myTime.getTime()[1])
            seconds = int(myTime.getTime()[2])
            tiking = myTime.keepThiking(seconds)
            print(tiking)
            print(f"Time is: {hours}:{minutes}:{seconds}")
            time.sleep(1)

    print(myAlarm.triggerAlarm())
    print("Thanks for playing!!")

if __name__ == "__main__":
    main()
```

Task 61

Bird has features name and amount of eggs.
Amount of eggs has to be between 1 and 10.

Migratory has special features: there is attribute named country that is the destination country and month when the migration mainly occurs.

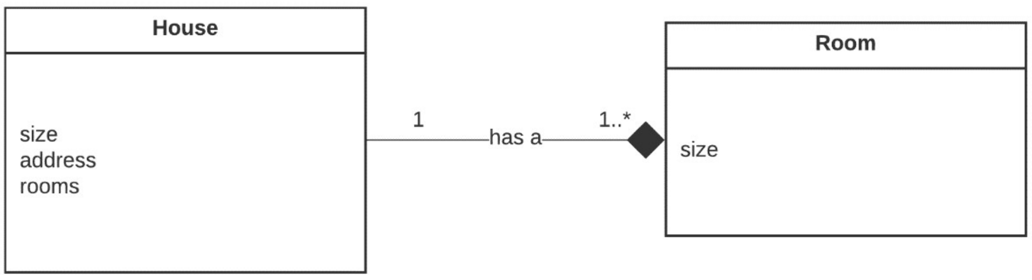
Country name has to begin with a cap and its length has to be between 5 to 20. Month has to be between 1 and 12.

```
class Bird:

  valid_range = range(1, 11)

  def __init__(self, name, eggs):
      self.name = name
      self.eggs = eggs

  @property
  def eggs(self):
      return self._eggs

  @eggs.setter
  def eggs(self, eggs):
      if eggs in Bird.valid_range:
          self._eggs = eggs
      else:
          raise ValueError(f"Birds eggs has to be between 1 and 10
(both inclusive). You entered {eggs}")


class Migratory(Bird):

  country_range = range(5, 21)
  month_range = range(1, 13)

  def __init__(self, name, eggs, country, month):
    self.country = country
    self.month = month
    super().__init__(name, eggs)

  @property
  def month(self):
    return self._month

  @month.setter
  def month(self, month):
    if month in Migratory.month_range:
      self._month = month
    else:
      raise ValueError(f"Enter a month between 1 and 12 (both
inclusive). You entered {month}")
```

```
  @property
  def country(self):
    return self._country

  @country.setter
  def country(self, country):
    if country == country.capitalize():
      if len(country) in Migratory.country_range:
        self._country = country
      else:
        raise ValueError(f"Enter a country between 5 and 20 (both
inclusive). You entered {country}")
    else:
      raise ValueError(f"Make sure that the first letter of your
country is capitalized and the rest lower case. You entered
{country}")


def main():
  # falcon = Bird("falcon", 10)
  # print(falcon._eggs)

  falcon = Migratory("falcon", 5, "Spain", 12)


if __name__ == "__main__":
    main()
```

Task 62

Create a OOP app about this topic. House with 3 Rooms (note: composition)



```
class House:
    def __init__(self, size, address, rooms):
        self.size = size
        self.address = address
        self._room = Room(rooms)

    def setSize(self, size):
        self.size = size

    def setAddress(self, address):
        self.address = address

    def setRoomSize(self, room):
        self._room = room

    def getSize(self):
        return self.size

    def getAddress(self):
        return self.address

    def getRoomSize(self):
        return self._room


class Room:
    def __init__(self, size):
        self.size = size
    def setRoomSize(self, rooms):

        self.size = rooms


    def getRoomSize(self):

        return self.size


# main

# printInfo() function

def printInfo(house):

    print("House size:", house.getSize())

    print("House address:", house.getAddress())

    print("House rooms:", house.getRoomSize().getRoomSize())



house = House("108sq", "Randomstreet 101", 4)

printInfo(house)

room = Room(3)

print("-"*20)

printInfo(house)
```

Task 63

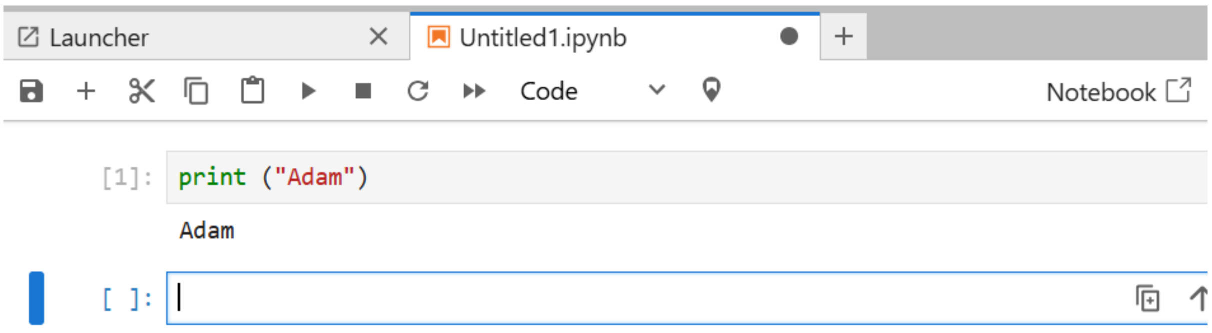
Band and members

```
#Define Member class
class Member:

    #Init member class with name and role
    def __init__(self, name, role):
        self.Name = name
        self.Role = role

    def PrintInfo(self):
        print("Member: {} Role: {}".format(self.Name,self.Role))

#define Band class
class Band:

    #Init band with name and genre
    def __init__(self,name, genre):
        self.Name = name
        self.Genre = genre
        self.Members = []

    #Adds a member to this band
    def AddMember(self,member):
        self.Members.append(member)

    def PrintInfo(self):
        print("Band: {} Genre: {}".format(self.Name, self.Genre))
        for x in self.Members:
            x.PrintInfo()




#Create band
band = Band("Best Band", "Rock")

#Create 4 members
member1 = Member("Jack", "Guitar")
member2 = Member("John", "Singer")
member3 = Member("Will", "Drums")
member4 = Member("Peter", "Bass")


#Add the members to the band
band.AddMember(member1)
band.AddMember(member2)
band.AddMember(member3)
band.AddMember(member4)

#Print band info
band.PrintInfo()
```

Task 9: Jupyter

Task 64

Create a small app that prints your name using Jupyter.



```
[1]: print ("Adam")
Adam
[ ]:
```

Task 65

Try with Jupyter: program checks if given email address contains @. Use own code and then library function.

```
email = "don@donn.com"
print("The string '@' is present in the string: ",email.__contains__
('@'))
```

Tasks 10: Exceptions

Get to know exceptions! Try to open a file for reading that does not exist.

```
try:
    with open("countries.odt", "r") as reader:
        for line in reader:
            print

except Exception as exc:
    print("File cannot be opened!", exc)
```

Task 68
Use the file "countries.txt".
Catch min 2 different exceptions when reading or writing to the file.

```
population = 23000000
country = "France"
countries_list = []

try:
    with open("countries.txt", "r") as reader:
        for line in reader:
            formatted_line = line.strip()
            countries_list.append(formatted_line)

        print(countries_list)

    with open("countries.txt", "a") as writer:
        for element in countries_list:
            if country in element:
                raise Exception
        writer.write("\n" + country + " " + str(population))

except Exception as exc:
    print("{} already exists!".format(country), exc)
```

Task 69

**Catch min 2 different exception when user gives a year number.
Give proper messages to the user.**

```
# exceptions

try:

    x = input("Please enter a number: ")

    x = int(x)

    x = x + 1

    y = x/0

except ValueError:

    print("Oops!  That was no valid number.  Try again...")

except ZeroDivisionError:

    print("Division by 0!!")

except BaseException:

    print("Something weird happened...")
```

Tasks 11: charts

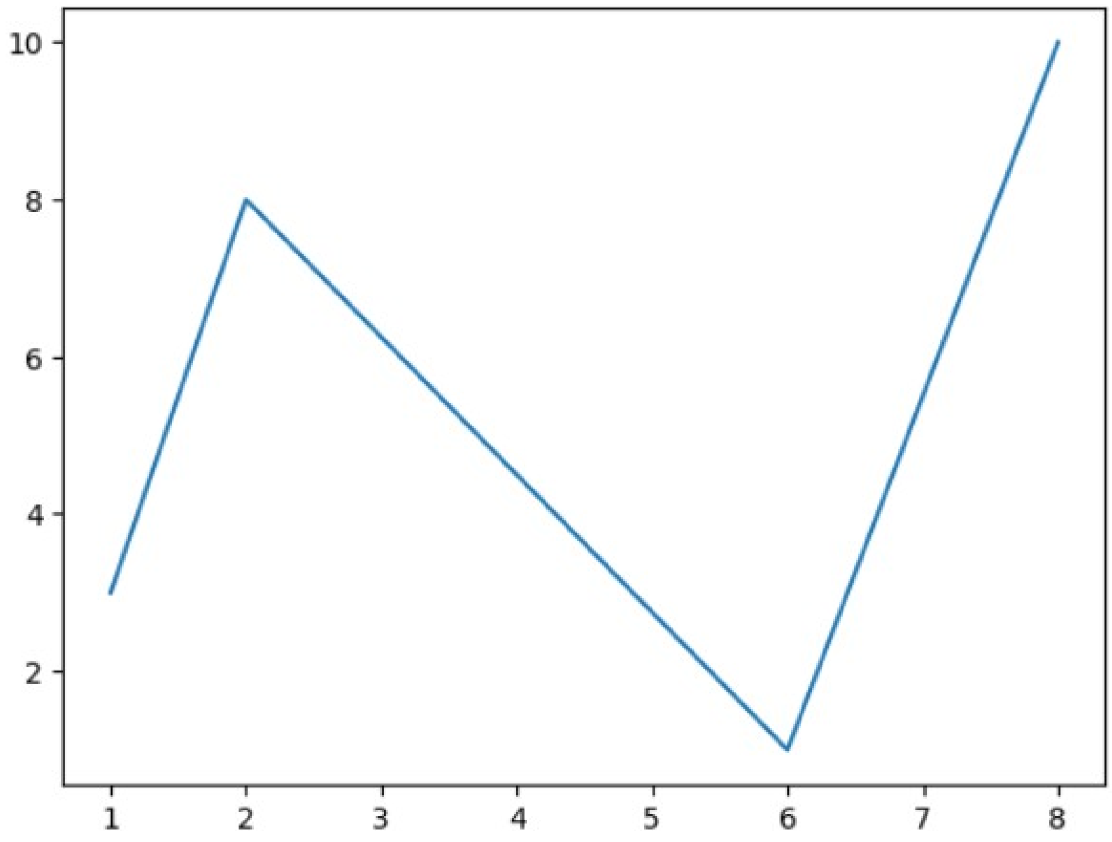
Task 70

Create a basic line plot using some x- and y-points:

```
import matplotlib.pyplot as plt
import numpy as np
x_value = np.array([1, 2, 6, 8])

## The coordinates of the points on the y-axis:

y_value = np.array([3, 8, 1, 10])
```

## Draw the points and connect them with straight lines:

plt.plot(x_value, y_value)



Task 70

Open this place

https://python-graph-gallery.com/

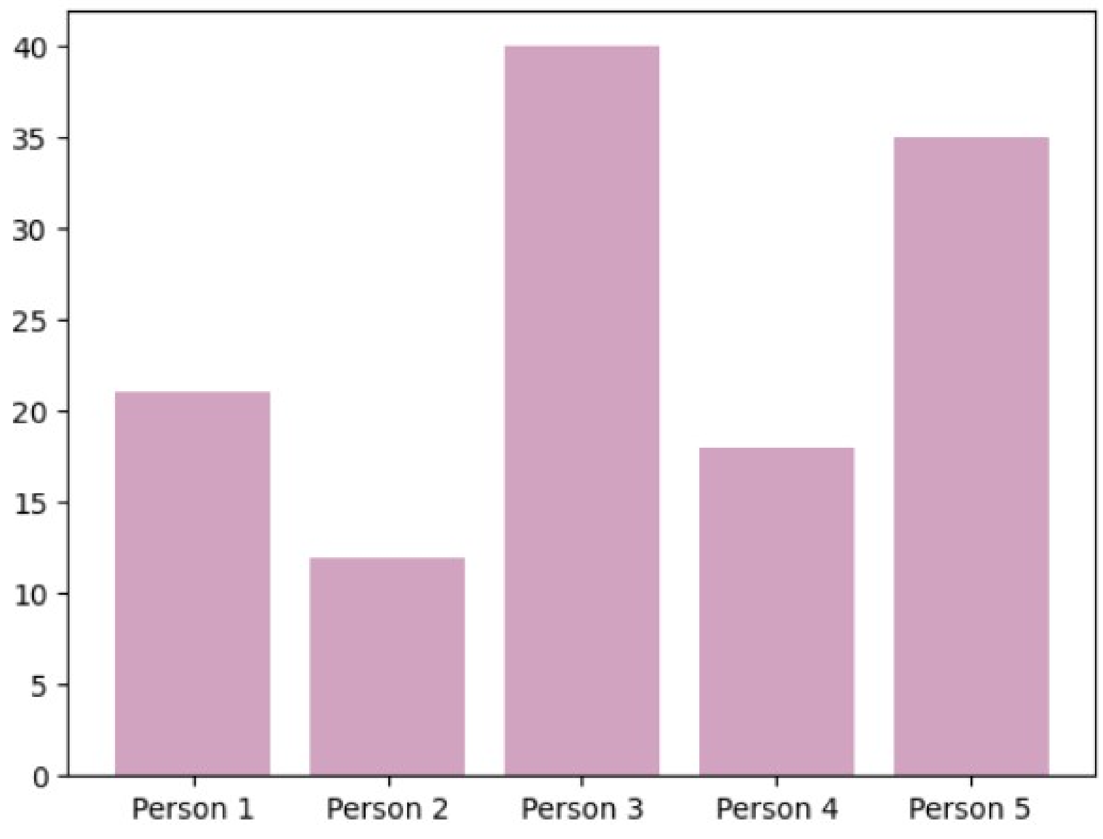
Present 4 different chart types using your own data.
Add there chart figure and code.

a)

#barplot chart

```
import numpy as np

import matplotlib.pyplot as plt


# making a dataset

age = [21, 12, 40, 18, 35]

bars = ('Person 1', 'Person 2', 'Person 3', 'Person 4', 'Person 5')

y_pos = np.arange(len(bars))


# plotting the chart

plt.bar(y_pos, age , color=(0.7, 0.4, 0.6, 0.6)) #Uniform (all bars
the same) color using RGB: An amount of red, green and blue + the
transparency and it returns a color.

plt.xticks(y_pos, bars)

plt.show()
```

Result

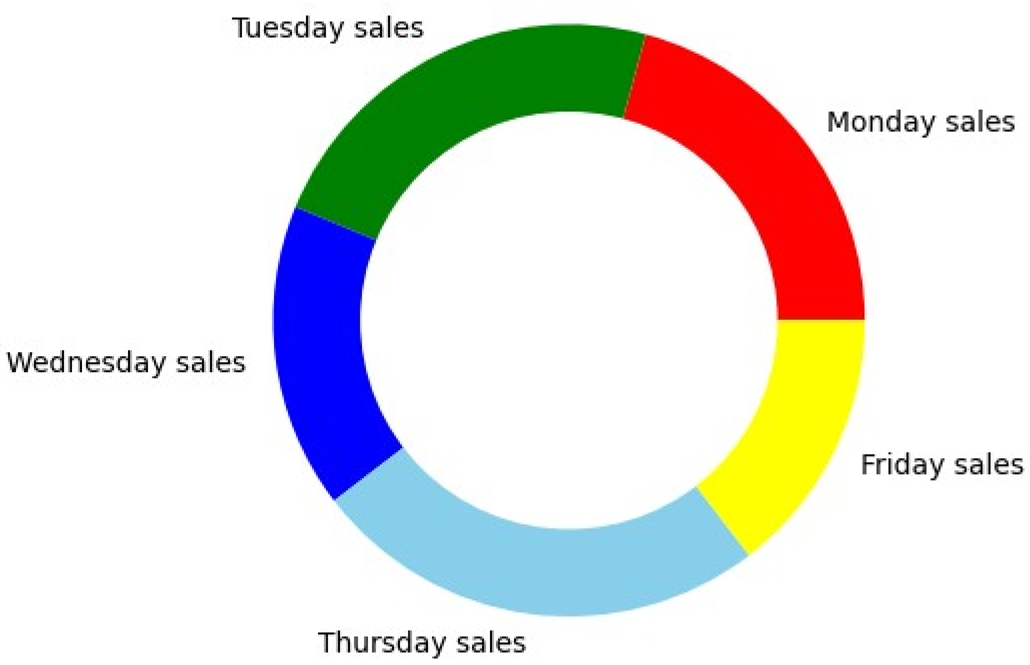


b)

#donut plot

```
import matplotlib.pyplot as plt


# create data
names='Monday sales', 'Tuesday sales', 'Wednesday sales', 'Thursday
sales', 'Friday sales'
size=[10,11,8,12,7]


# Create a circle for the center of the plot
my_circle=plt.Circle( (0,0), 0.7, color='white')
```

```
# Give color names

plt.pie(size, labels=names,
colors=['red','green','blue','skyblue','yellow'])

p=plt.gcf()

p.gca().add_artist(my_circle)

plt.show()


# Custom colors --> colors will cycle

plt.pie(size, labels=names, colors=['red','green'])

p=plt.gcf()

p.gca().add_artist(my_circle)

plt.show()


from palettable.colorbrewer.qualitative import Pastel1_7

plt.pie(size, labels=names, colors=Pastel1_7.hex_colors)

p=plt.gcf()

p.gca().add_artist(my_circle)

plt.show()
```



c)

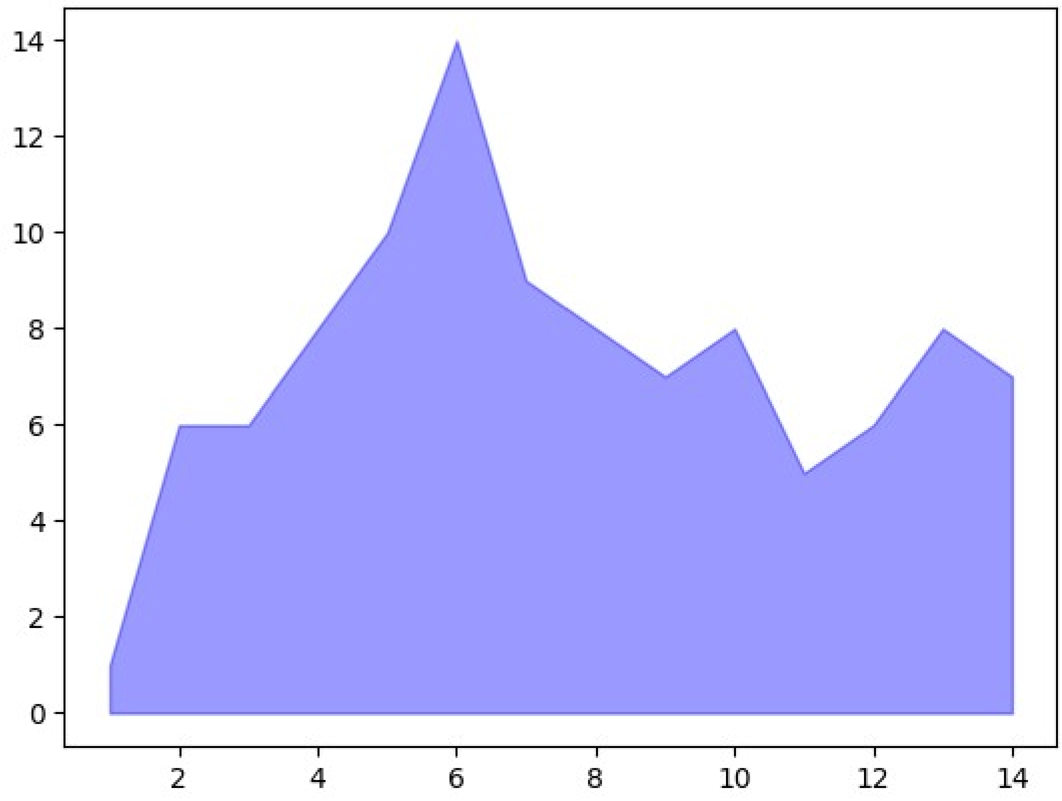
#wordcloud chart

```
# Libraries
from wordcloud import WordCloud
import matplotlib.pyplot as plt
# Create a list of word
text=("Moon, Phobos, Deimos, Io, Europa, Ganymede, Callisto, Titan,
Enceladus, Iapetus, Ariel, Miranda, Umbriel, Triton, Nereid")


wordcloud = WordCloud(width=480, height=480, max_font_size=28,
min_font_size=16).generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```

d)

#area chart

```
import numpy as np

import matplotlib.pyplot as plt


# create data

x=range(1,15)

y=[1,6,6,8,10,14,9,8,7,8,5,6,8,7]


# Change the color and its transparency

plt.fill_between( x, y, color="blue", alpha=0.4)

plt.show()


# Same, but add a stronger line on top (edge)

plt.fill_between( x, y, color="skyblue", alpha=0.2)

plt.plot(x, y, color="Slateblue", alpha=0.6)
```



e) Create a best fit line to the chart:

sample points are ehere

X = [0, 6, 11, 14, 22]

Y = [1, 7, 12, 15, 21]

```
import matplotlib.pyplot as plt
# best fit example
# sample points
X = [0, 6, 11, 14, 22]
Y = [1, 7, 12, 15, 21]
# solve for a and b
def best_fit(X, Y):
   xbar = sum(X)/len(X)
   ybar = sum(Y)/len(Y)
   n = len(X) # or len(Y)
   numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
   denum = sum([xi**2 for xi in X]) - n * xbar**2

   b = numer / denum
   a = ybar - b * xbar

   print('best fit line:\ny = {:.2f} + {:.2f}x'.format(a, b))

   return a, b

# solution
a, b = best_fit(X, Y)

# plotting in separate process
xbar = sum(X)/len(X)
ybar = sum(Y)/len(Y)
n = len(X) # or len(Y)

numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
denum = sum([xi**2 for xi in X]) - n * xbar**2

b = numer / denum
a = ybar - b * xbar

fitArray1 = [];
fitArray2 = [];
for s in range (5):
    fitArray1.append(X[s])
    fitArray2.append(a + b*X[s])
```

```
plt.plot(fitArray1, fitArray2)
plt.plot(X,Y,linestyle='none', marker='o')
plt.show()
```

****

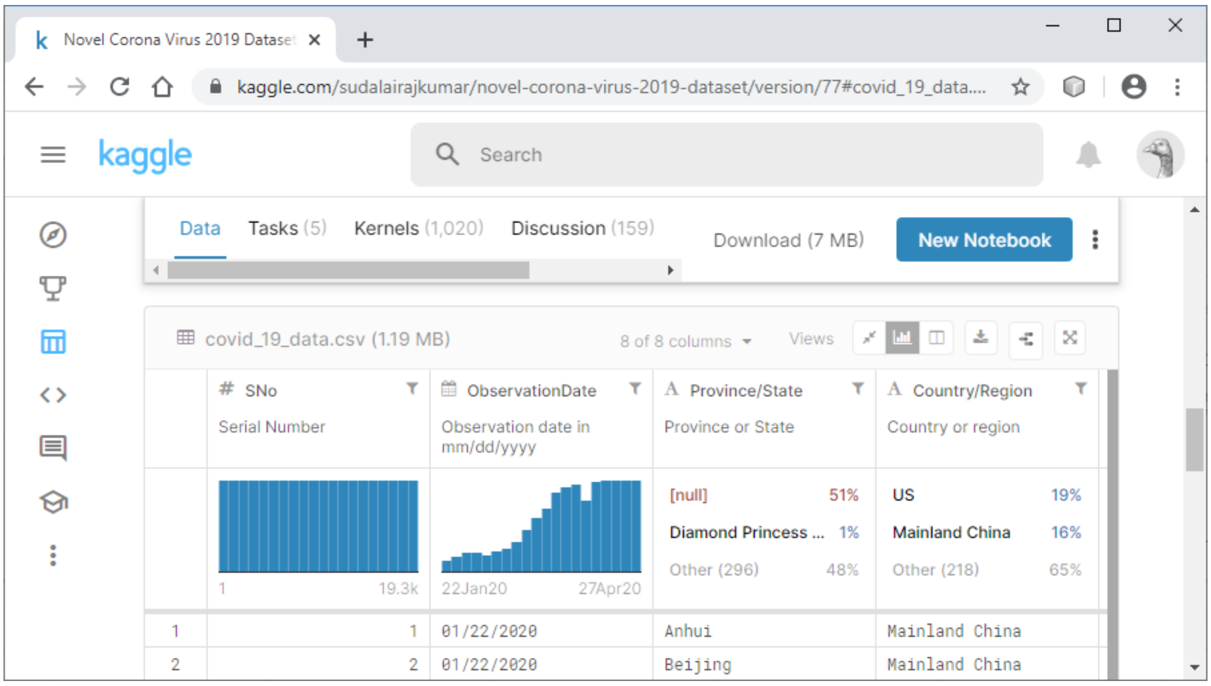
Tasks 12: Python data modules

Task 80
Create you own data analysis example using pandas and numpy ans seaborn.

The whole task is here below: try to complete it step by step and learn about those important modules and big data!

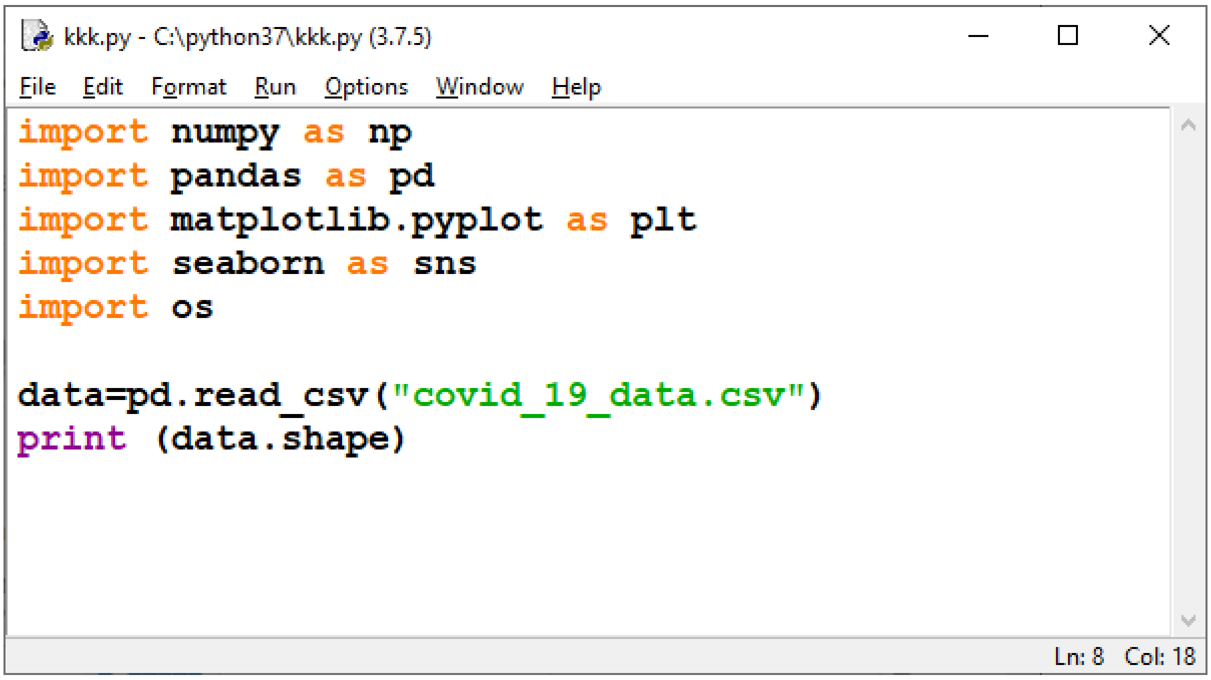
Python & Data Analysis example

Data is taken from Internet now


Novel Corona Virus 2019 Dataset
kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset/version/77#covid_19_data....
kaggle
Search
Data
Tasks (5)
Kernels (1,020)
Discussion (159)
Download (7 MB)
New Notebook
covid_19_data.csv (1.19 MB)
8 of 8 columns
Views

	# SNo	ObservationDate	Province/State	Country/Region
	Serial Number	Observation date in mm/dd/yyyy	Province or State	Country or region
	1 – 19.3k	22Jan20 – 27Apr20	[null] 51% Diamond Princess ... 1% Other (296) 48%	US 19% Mainland China 16% Other (218) 65%
1	1	01/22/2020	Anhui	Mainland China
2	2	01/22/2020	Beijing	Mainland China

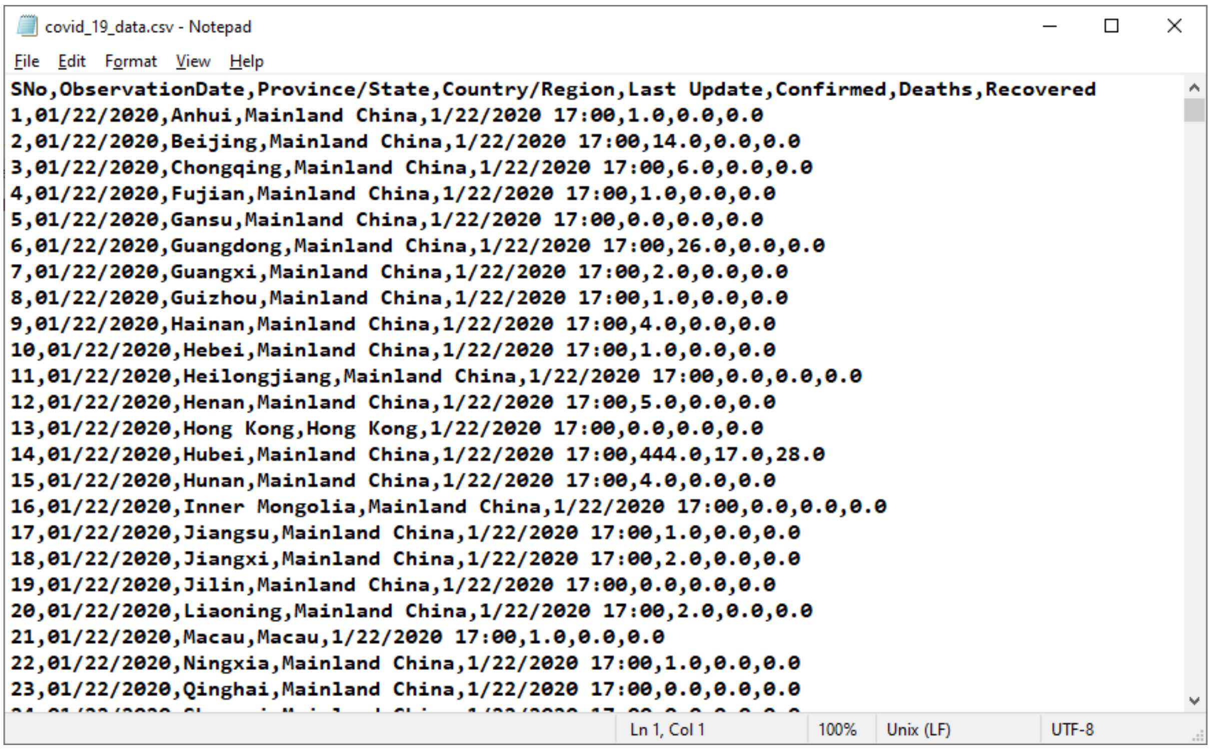
Main python code to read data:


kkk.py - C:\python37\kkk.py (3.7.5)
File Edit Format Run Options Window Help
Ln: 8 Col: 18

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

data=pd.read_csv("covid_19_data.csv")
print (data.shape)
```

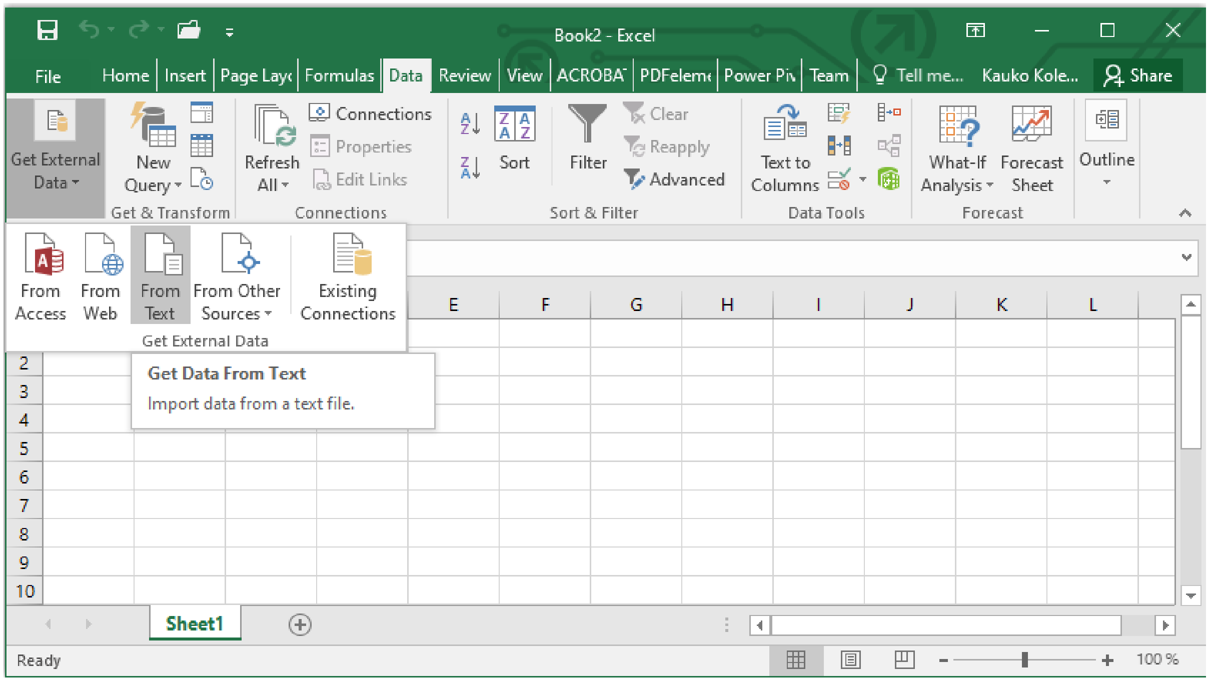
Take a look at the data in Excel

Data looks like this



```
SNo,ObservationDate,Province/State,Country/Region,Last Update,Confirmed,Deaths,Recovered
1,01/22/2020,Anhui,Mainland China,1/22/2020 17:00,1.0,0.0,0.0
2,01/22/2020,Beijing,Mainland China,1/22/2020 17:00,14.0,0.0,0.0
3,01/22/2020,Chongqing,Mainland China,1/22/2020 17:00,6.0,0.0,0.0
4,01/22/2020,Fujian,Mainland China,1/22/2020 17:00,1.0,0.0,0.0
5,01/22/2020,Gansu,Mainland China,1/22/2020 17:00,0.0,0.0,0.0
6,01/22/2020,Guangdong,Mainland China,1/22/2020 17:00,26.0,0.0,0.0
7,01/22/2020,Guangxi,Mainland China,1/22/2020 17:00,2.0,0.0,0.0
8,01/22/2020,Guizhou,Mainland China,1/22/2020 17:00,1.0,0.0,0.0
9,01/22/2020,Hainan,Mainland China,1/22/2020 17:00,4.0,0.0,0.0
10,01/22/2020,Hebei,Mainland China,1/22/2020 17:00,1.0,0.0,0.0
11,01/22/2020,Heilongjiang,Mainland China,1/22/2020 17:00,0.0,0.0,0.0
12,01/22/2020,Henan,Mainland China,1/22/2020 17:00,5.0,0.0,0.0
13,01/22/2020,Hong Kong,Hong Kong,1/22/2020 17:00,0.0,0.0,0.0
14,01/22/2020,Hubei,Mainland China,1/22/2020 17:00,444.0,17.0,28.0
15,01/22/2020,Hunan,Mainland China,1/22/2020 17:00,4.0,0.0,0.0
16,01/22/2020,Inner Mongolia,Mainland China,1/22/2020 17:00,0.0,0.0,0.0
17,01/22/2020,Jiangsu,Mainland China,1/22/2020 17:00,1.0,0.0,0.0
18,01/22/2020,Jiangxi,Mainland China,1/22/2020 17:00,2.0,0.0,0.0
19,01/22/2020,Jilin,Mainland China,1/22/2020 17:00,0.0,0.0,0.0
20,01/22/2020,Liaoning,Mainland China,1/22/2020 17:00,2.0,0.0,0.0
21,01/22/2020,Macau,Macau,1/22/2020 17:00,1.0,0.0,0.0
22,01/22/2020,Ningxia,Mainland China,1/22/2020 17:00,1.0,0.0,0.0
23,01/22/2020,Qinghai,Mainland China,1/22/2020 17:00,0.0,0.0,0.0
```

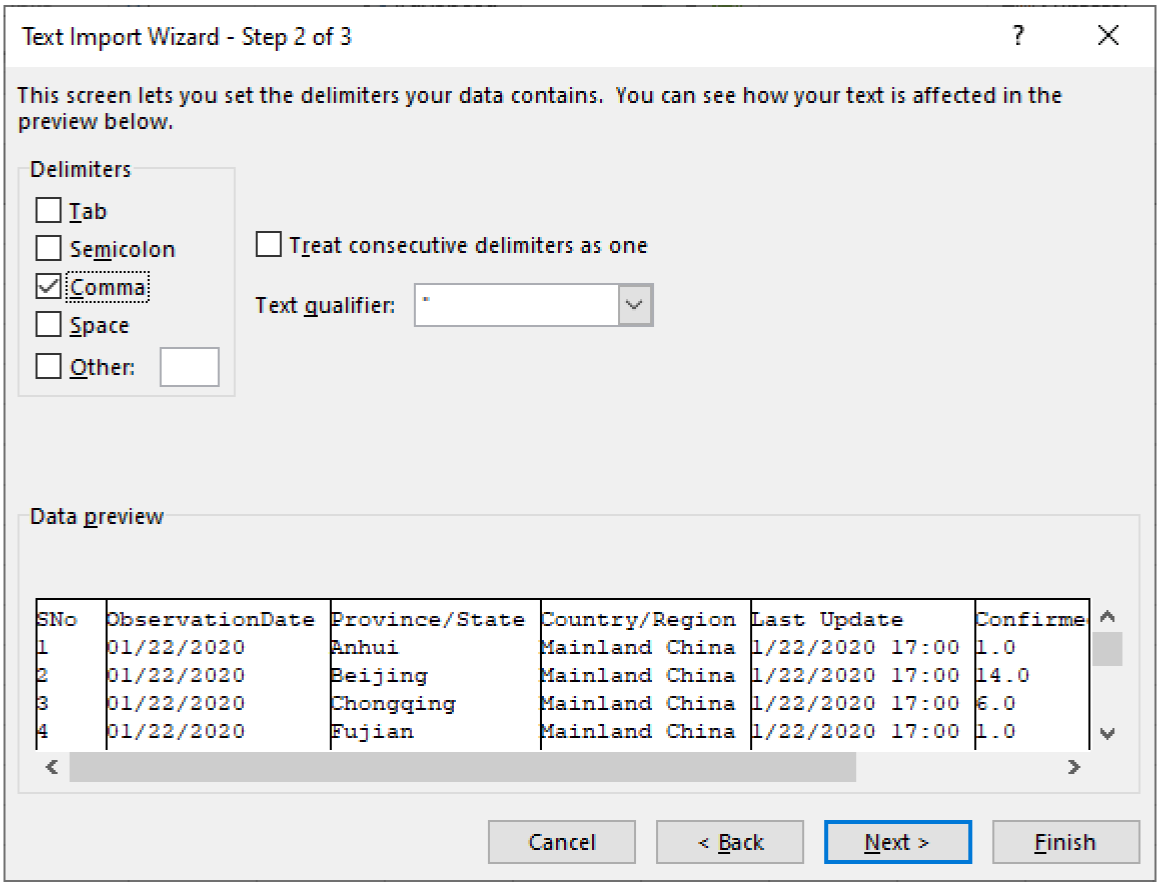
Import data to Excel first (easier to read)



Get Data From Text

Import data from a text file.

AND



This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

- [ ] Tab
- [ ] Semicolon
- [✓] Comma
- [ ] Space
- [ ] Other:

[ ] Treat consecutive delimiters as one

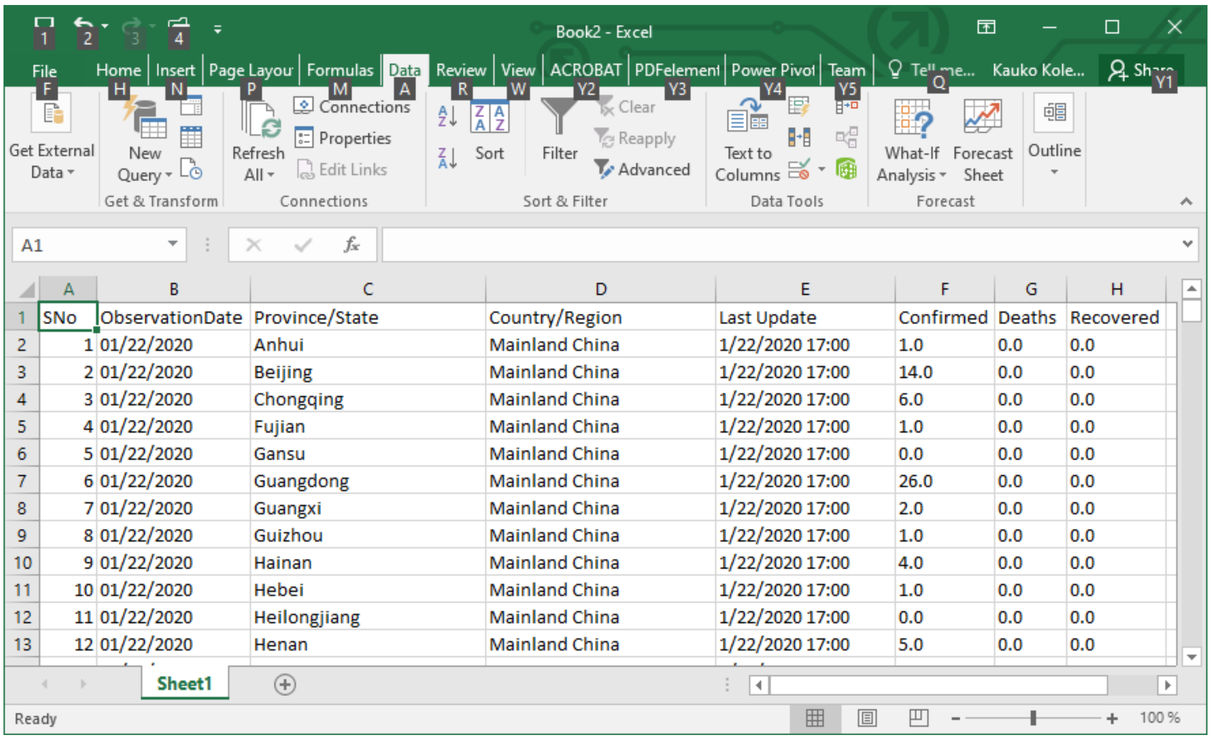
Text qualifier: "

Data preview

```
SNo  ObservationDate  Province/State  Country/Region  Last Update      Confirme
1    01/22/2020       Anhui           Mainland China  1/22/2020 17:00  1.0
2    01/22/2020       Beijing         Mainland China  1/22/2020 17:00  14.0
3    01/22/2020       Chongqing       Mainland China  1/22/2020 17:00  6.0
4    01/22/2020       Fujian          Mainland China  1/22/2020 17:00  1.0
```

89

AND



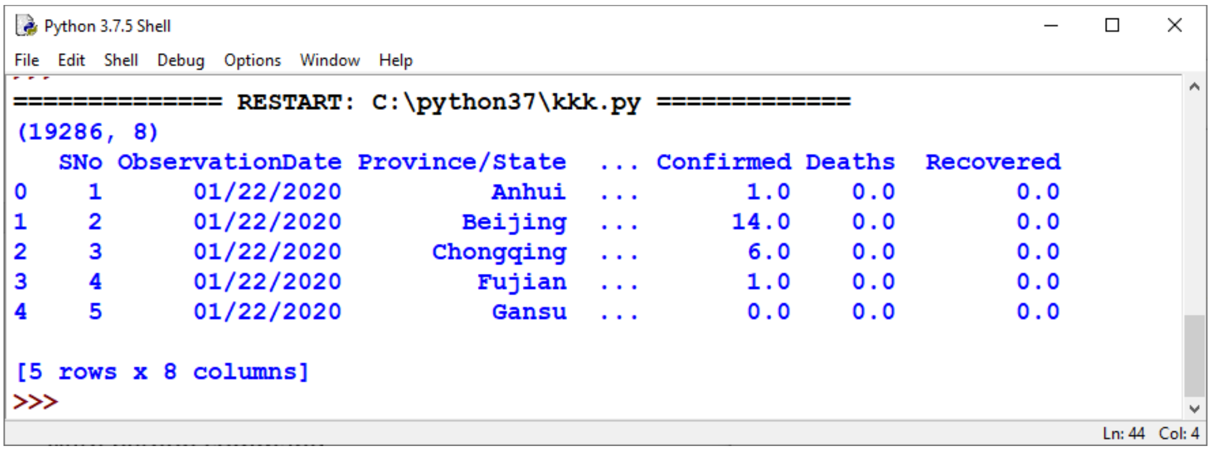
	A	B	C	D	E	F	G	H
1	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
2	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
3	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
4	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
5	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
6	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
7	6	01/22/2020	Guangdong	Mainland China	1/22/2020 17:00	26.0	0.0	0.0
8	7	01/22/2020	Guangxi	Mainland China	1/22/2020 17:00	2.0	0.0	0.0
9	8	01/22/2020	Guizhou	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
10	9	01/22/2020	Hainan	Mainland China	1/22/2020 17:00	4.0	0.0	0.0
11	10	01/22/2020	Hebei	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
12	11	01/22/2020	Heilongjiang	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
13	12	01/22/2020	Henan	Mainland China	1/22/2020 17:00	5.0	0.0	0.0

Data to Python file

With python command we can print data

```
print(data.head())
```

Shows first lines with headers



```
============== RESTART: C:\python37\kkk.py =============
(19286, 8)
   SNo ObservationDate Province/State  ... Confirmed Deaths  Recovered
0    1      01/22/2020          Anhui  ...       1.0    0.0        0.0
1    2      01/22/2020        Beijing  ...      14.0    0.0        0.0
2    3      01/22/2020      Chongqing  ...       6.0    0.0        0.0
3    4      01/22/2020         Fujian  ...       1.0    0.0        0.0
4    5      01/22/2020          Gansu  ...       0.0    0.0        0.0

[5 rows x 8 columns]
>>>
```

Parsing

Parsing or at least checking which columns to take with

We could now parse the data list a bit. E.g. column Sno is not needed, neither column Last Update.

```
## cleaning data
data.drop("SNo", axis=1, inplace=True)
data.drop("Last Update", axis=1, inplace=True)
data.info()
```

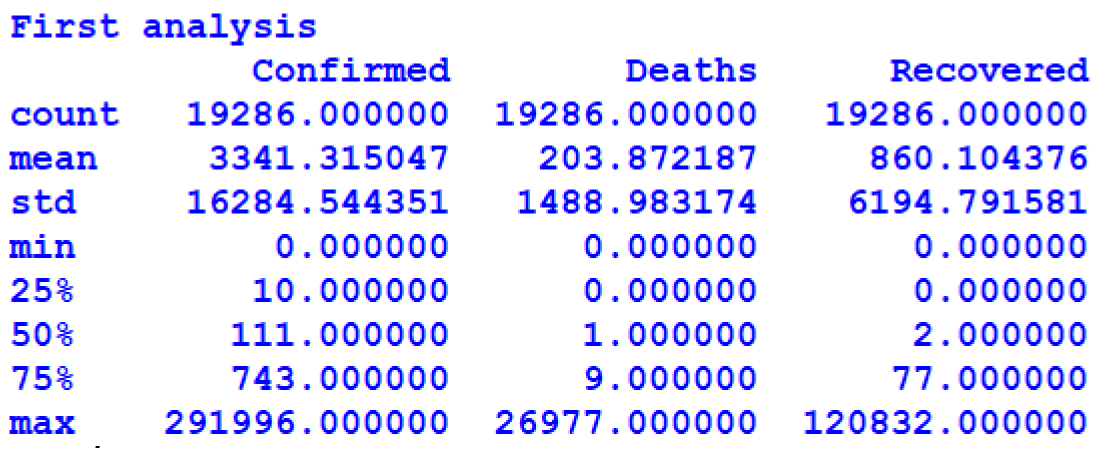
Result is

```
[5 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19286 entries, 0 to 19285
Data columns (total 6 columns):
ObservationDate    19286 non-null object
Province/State     9466 non-null object
Country/Region     19286 non-null object
Confirmed          19286 non-null float64
Deaths             19286 non-null float64
Recovered          19286 non-null float64
```

First analysis of numerical data

```
print("First analysis")
print(data.describe())
```

We get:



```
First analysis
            Confirmed        Deaths      Recovered
count    19286.000000  19286.000000   19286.000000
mean      3341.315047    203.872187     860.104376
std      16284.544351   1488.983174    6194.791581
min          0.000000      0.000000       0.000000
25%         10.000000      0.000000       0.000000
50%        111.000000      1.000000       2.000000
75%        743.000000      9.000000      77.000000
max     291996.000000  26977.000000  120832.000000
```

Duplicate rows?

We can now check if there are duplicate rows:

```
## check duplicates
duplicate_rows=data.duplicated(subset=['Country/Region','Province/State','ObservationDate'])
print(data[duplicate_rows])
```

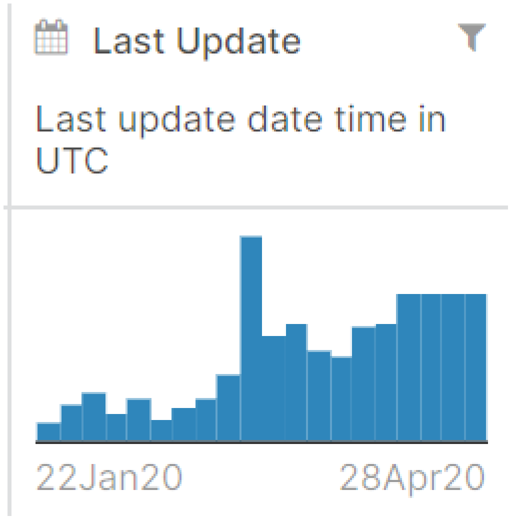
We got

```
     ObservationDate Province/State  ... Deaths  Recovered
4925      03/11/2020          Gansu  ...    0.0        0.0
4926      03/11/2020          Hebei  ...    0.0        0.0
5146      03/12/2020          Gansu  ...    0.0        0.0
5147      03/12/2020          Hebei  ...    0.0        0.0

[4 rows x 6 columns]
```

Still we can see that main data is ok.

Countries

How many countries are there? Countries that have infections when this doc is written.

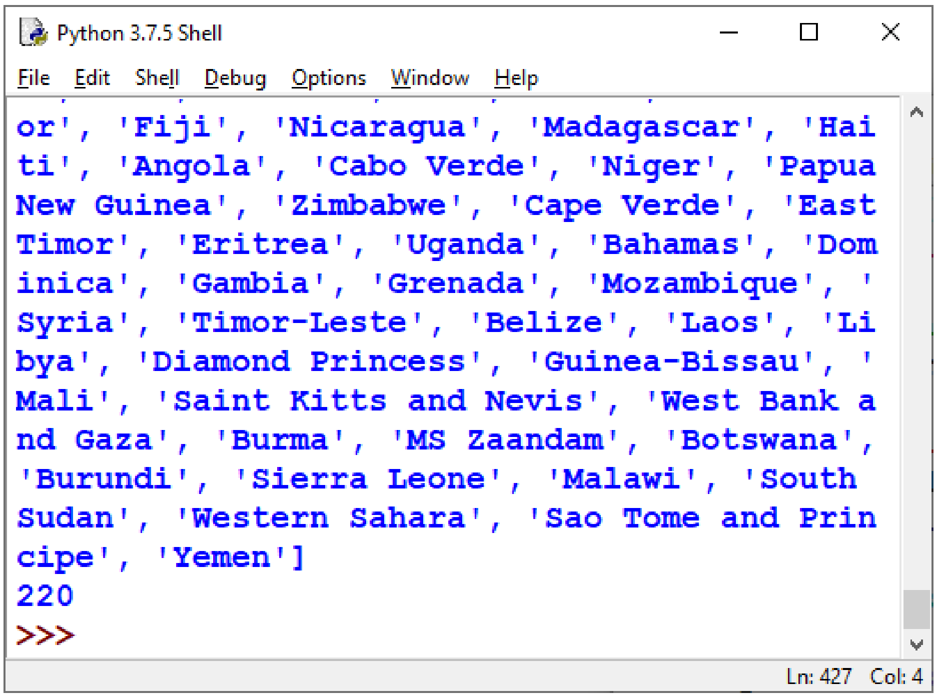


You can see here last update date.

Python code

```
print("Countries")
## countries
country_list=list(data['Country/Region'].unique())
print(country_list)
print (len(country_list))
```

We get



```
or', 'Fiji', 'Nicaragua', 'Madagascar', 'Hai
ti', 'Angola', 'Cabo Verde', 'Niger', 'Papua
New Guinea', 'Zimbabwe', 'Cape Verde', 'East
Timor', 'Eritrea', 'Uganda', 'Bahamas', 'Dom
inica', 'Gambia', 'Grenada', 'Mozambique', '
Syria', 'Timor-Leste', 'Belize', 'Laos', 'Li
bya', 'Diamond Princess', 'Guinea-Bissau', '
Mali', 'Saint Kitts and Nevis', 'West Bank a
nd Gaza', 'Burma', 'MS Zaandam', 'Botswana',
'Burundi', 'Sierra Leone', 'Malawi', 'South
Sudan', 'Western Sahara', 'Sao Tome and Prin
cipe', 'Yemen']
220
>>>
```

220 countries…

Cases per nation

```
## cases pr nation
df_country=data.groupby(['Country/Region']).max().reset_index(drop=None)
print(df_country[['Country/Region','Confirmed','Deaths','Recovered']])
```

We get

```
                     Country/Region  Confirmed   Deaths  Recovered
0                        Azerbaijan        1.0      0.0        0.0
1                   ('St. Martin',)        2.0      0.0        0.0
2                       Afghanistan     1703.0     57.0      220.0
3                           Albania      736.0     28.0      422.0
4                           Algeria     3517.0    432.0     1558.0
5                           Andorra      743.0     40.0      385.0
6                            Angola       27.0      2.0        6.0
7               Antigua and Barbuda       24.0      3.0       11.0
8                         Argentina     4003.0    197.0     1140.0
9                           Armenia     1808.0     29.0      848.0
10                            Aruba        4.0      0.0        0.0
11                        Australia     3004.0     34.0     2227.0
12                          Austria    15274.0    549.0    12362.0
13                       Azerbaijan     1678.0     22.0     1162.0
14                          Bahamas       80.0     11.0       22.0
15                     Bahamas, The        4.0      0.0        0.0
16                          Bahrain     2723.0      8.0     1218.0
17                       Bangladesh     5913.0    152.0      131.0
18                         Barbados       80.0      6.0       39.0
19                          Belarus    11289.0     75.0     1740.0
20                          Belgium    46687.0   7207.0    10878.0
21                           Belize       18.0      2.0        6.0
22                            Benin       64.0      1.0       33.0
23                           Bhutan        7.0      0.0        4.0
24                          Bolivia     1014.0     53.0       98.0
25           Bosnia and Herzegovina     1565.0     60.0      659.0
26                         Botswana       22.0      1.0        0.0
27                           Brazil    67446.0   4603.0    31142.0
28                           Brunei      138.0      1.0      124.0
29                         Bulgaria     1363.0     58.0      206.0
```

Smaller list

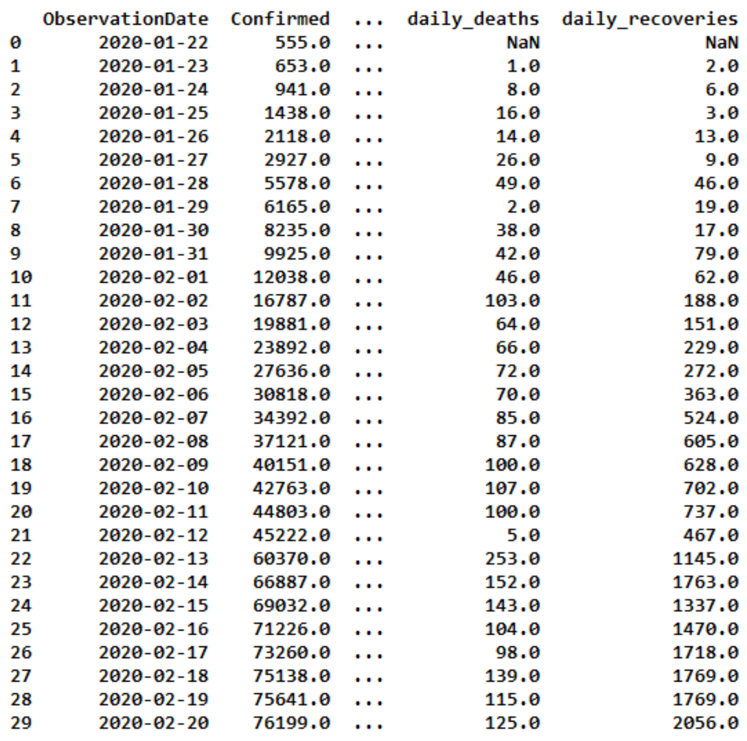
Bigger list can be put to parts

```
## cases pr nation
df_country=data.groupby(['Country/Region']).max().reset_index(drop=None)
df_part = df_country[50:100]
print(df_part[['Country/Region','Confirmed','Deaths','Recovered']])
```

```
        Country/Region  Confirmed   Deaths  Recovered
50              Cyprus      822.0     15.0      148.0
51      Czech Republic     7445.0    223.0     2826.0
52             Denmark     8698.0    427.0     5959.0
53    Diamond Princess      712.0     13.0      645.0
54            Djibouti     1035.0      2.0      477.0
55            Dominica       16.0      0.0       13.0
56  Dominican Republic     6293.0    282.0      993.0
57          East Timor        1.0      0.0        0.0
58             Ecuador    23240.0    663.0     1557.0
59               Egypt     4782.0    337.0     1236.0
60         El Salvador      323.0      8.0       89.0
61   Equatorial Guinea      258.0      1.0        9.0
62             Eritrea       39.0      0.0       13.0
63             Estonia     1647.0     50.0      233.0
64            Eswatini       65.0      1.0       10.0
65            Ethiopia      124.0      3.0       50.0
66       Faroe Islands        2.0      0.0        0.0
67                Fiji       18.0      0.0       12.0
68             Finland     4695.0    193.0     2500.0
69              France   164589.0  23293.0    45513.0
```

Getting timeseries data

```
## cases per day
df_by_date=data.groupby(['ObservationDate']).sum().reset_index(drop=None)
df_by_date['daily_cases']=df_by_date.Confirmed.diff()
df_by_date['daily_deaths']=df_by_date.Deaths.diff()
df_by_date['daily_recoveries']=df_by_date.Recovered.diff()
print(df_by_date)
```



```
   ObservationDate  Confirmed  ...  daily_deaths  daily_recoveries
0       2020-01-22      555.0  ...           NaN               NaN
1       2020-01-23      653.0  ...           1.0               2.0
2       2020-01-24      941.0  ...           8.0               6.0
3       2020-01-25     1438.0  ...          16.0               3.0
4       2020-01-26     2118.0  ...          14.0              13.0
5       2020-01-27     2927.0  ...          26.0               9.0
6       2020-01-28     5578.0  ...          49.0              46.0
7       2020-01-29     6165.0  ...           2.0              19.0
8       2020-01-30     8235.0  ...          38.0              17.0
9       2020-01-31     9925.0  ...          42.0              79.0
10      2020-02-01    12038.0  ...          46.0              62.0
11      2020-02-02    16787.0  ...         103.0             188.0
12      2020-02-03    19881.0  ...          64.0             151.0
13      2020-02-04    23892.0  ...          66.0             229.0
14      2020-02-05    27636.0  ...          72.0             272.0
15      2020-02-06    30818.0  ...          70.0             363.0
16      2020-02-07    34392.0  ...          85.0             524.0
17      2020-02-08    37121.0  ...          87.0             605.0
18      2020-02-09    40151.0  ...         100.0             628.0
19      2020-02-10    42763.0  ...         107.0             702.0
20      2020-02-11    44803.0  ...         100.0             737.0
21      2020-02-12    45222.0  ...           5.0             467.0
22      2020-02-13    60370.0  ...         253.0            1145.0
23      2020-02-14    66887.0  ...         152.0            1763.0
24      2020-02-15    69032.0  ...         143.0            1337.0
25      2020-02-16    71226.0  ...         104.0            1470.0
26      2020-02-17    73260.0  ...          98.0            1718.0
27      2020-02-18    75138.0  ...         139.0            1769.0
28      2020-02-19    75641.0  ...         115.0            1769.0
29      2020-02-20    76199.0  ...         125.0            2056.0
```

Graphical illustrations help to read data.

Final topics are coming here later…

This is the 1. version

Give comments, please!