Python tutorial - revisit

Python tutorial videos

Click on the below links to refresh your Python skills.

1. Array Math
2. NumPy Array
3. Functions
4. Data processing using NumPy and Pandas
5. Introduction to object oriented programming

Basic functions

Basic functions to check conditions (if and else)

Run the python program by activating the terminal and type and enter: python function.py

timepass exercise:

update function, speed_function_super() so that considers different fine for speed of 100 - 119, 120 - 139 and 140 - 159, based on speed_function_advanced( ). You can create fine based on your own rules, similar to 80 -100 limit based on speed_function_advanced( ).

Functions with Jupyter Notebook

Simple function example for speed calculator

Numpy - Introduction

We assume that you already know Python, either from doing the course ZZEN9021 Principles of Programming or something equivalent, so we won't be teaching it to you. This lesson is a refresher on the aspects of Python that are particularly important for this course.

In slide focuses on Numpy. Click on the below links to practise Numpy:

Practise link 1

Practise link 2

Now the run the below code.

```
import numpy as np

#create and print arrays

a = np.arange(6)                         # 1d array
print(a, ' a ')

b = np.arange(12).reshape(4,3)           # 2d array
print(b, ' b ')

c = np.arange(24).reshape(2,3,4)         # 3d array
print(c, ' c ')
```

```
import numpy as np

a = np.zeros((2,2))   # Create an array of all zeros
print(a)              # Prints "[[ 0.  0.]
                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones
print(b)              # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7)  # Create a constant array
print(c)               # Printse = np.random.random((2,2))  # Create an array filled with random va


d = np.eye(2)         # Create a 2x2 identity matrix
print(d)              # Prints "[[ 1.  0.]
                      #          [ 0.  1.]]"
```

```
e = np.random.random((2,2))  # Create an array filled with random values
print(e)                     # Might print "[[ 0.91940167  0.08143941]


f = np.random.random((10))  # Create an array filled with random values
print(f)                     # Might print "[[ 0.91940167  0.08143941]
```

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
print(a[0, 1])   # Prints "2"
b[0, 0] = 77     # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1])   # Prints "77"
```

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Two ways of accessing the data in the middle row of the array.
# Mixing integer indexing with slices yields an array of lower rank,
# while using only slices yields an array of the same rank as the
# original array:
row_r1 = a[1, :]    # Rank 1 view of the second row of a
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
print(row_r1, row_r1.shape)  # Prints "[5 6 7 8] (4,)"
print(row_r2, row_r2.shape)  # Prints "[[5 6 7 8]] (1, 4)"

# We can make the same distinction when accessing columns of an array:
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print(col_r1, col_r1.shape)  # Prints "[ 2  6 10] (3,)"
print(col_r2, col_r2.shape)  # Prints "[[ 2]
```

```
#           [ 6]
#           [10]] (3, 1)"
```

Numpy Array Math

Run the below Python code on an array.

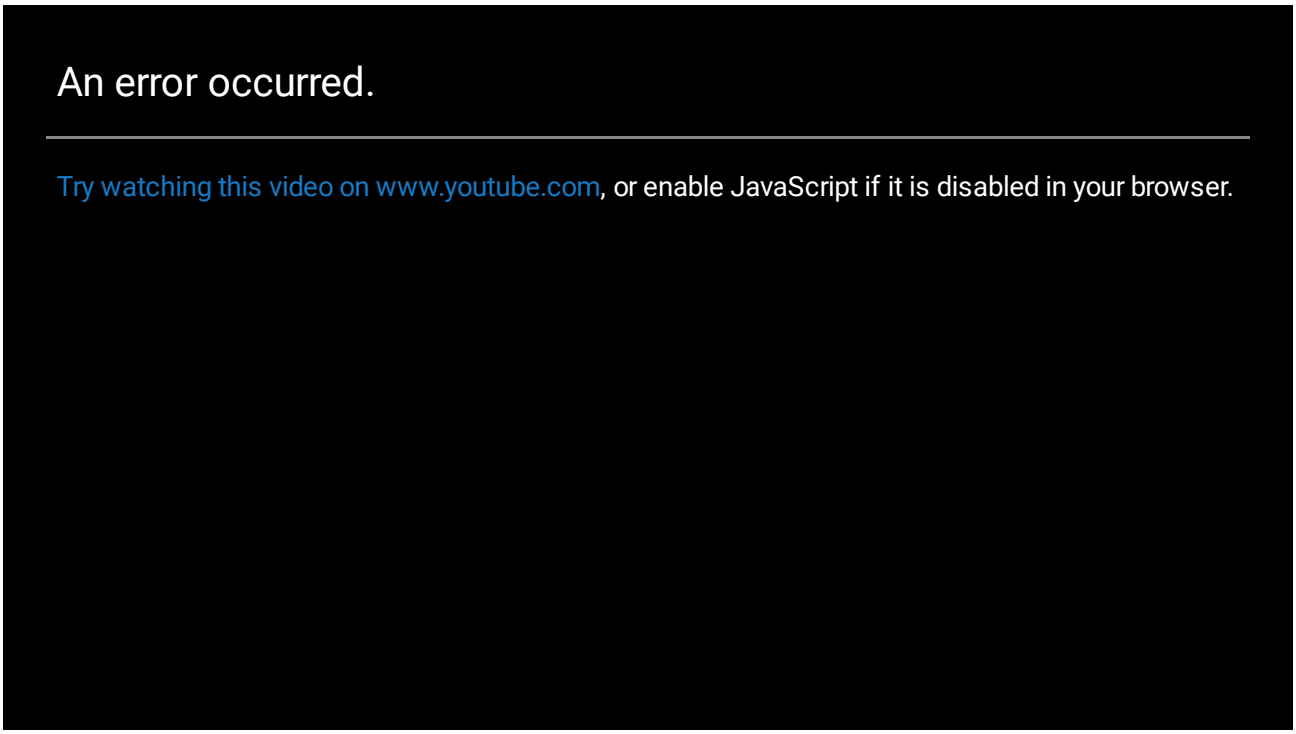
```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))

# Elementwise difference; both produce the array
# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print(x - y)
print(np.subtract(x, y))

# Elementwise product; both produce the array
# [[ 5.0 12.0]
#  [21.0 32.0]]
print(x * y)
print(np.multiply(x, y))

# Elementwise division; both produce the array
# [[ 0.2         0.33333333]
#  [ 0.42857143  0.5       ]]
print(x / y)
print(np.divide(x, y))

# Elementwise square root; produces the array
# [[ 1.          1.41421356]
#  [ 1.73205081  2.        ]]
print(np.sqrt(x))

#https://cs231n.github.io/python-numpy-tutorial/
```

Inner products

Note that unlike MATLAB, `*` is elementwise multiplication, not matrix multiplication. We instead use the `dot` function to compute inner products of vectors, to multiply a vector by a matrix, and to multiply matrices. `dot` is available both as a function in the Numpy module and as an instance method of array objects.

Run the code.

```
import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print(v.dot(w))
print(np.dot(v, w))

# Matrix / vector product; both produce the rank 1 array [29 67]
print(x.dot(v))
print(np.dot(x, v))

# Matrix / matrix product; both produce the rank 2 array
# [[19 22]
#  [43 50]]
print(x.dot(y))
print(np.dot(x, y))
```



https://cs231n.github.io/python-numpy-tutorial/

```
#source: https://cs231n.github.io/python-numpy-tutorial/

import numpy as np

x = np.array([[1,2],[3,4]])

print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"

print(' ----------')

x = np.array([[1,2], [3,4]])
print(x)    # Prints "[[1 2]
            #          [3 4]]"
print(x.T)  # Prints "[[1 3]
            #          [2 4]]"

# Note that taking the transpose of a rank 1 array does nothing:
v = np.array([1,2,3])
print(v)    # Prints "[1 2 3]"
print(v.T)  # Prints "[1 2 3]"



print(' ----------')

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = np.empty_like(x)   # Create an empty matrix with the same shape as x

# Add the vector v to each row of the matrix x with an explicit loop
for i in range(4):
    y[i, :] = x[i, :] + v
```

```
# Now y is the following
# [[ 2  2  4]
#  [ 5  5  7]
#  [ 8  8 10]
#  [11 11 13]]
print(y)
```

Numpy array examples

```
import numpy as np

import random

def numpy_lists():

 magic = np.random.rand(3,4)
 magic_three = np.random.rand(3,4,2)
 # homework, write a function for summing 3D magic
 magic_one = np.random.rand(10)

 print(magic, ' is magic ')

 print(np.sum(magic), 'is np sum')

 print(magic_one.shape, ' is magic one shape')
 print(magic_three.shape, ' magic three shape')
 print(magic.shape, ' is the shape')
 print(np.sum(magic_three), 'is np sum for magic three')



 magic_sum = sum_numpy(magic)
 print( magic_sum, ' is magic sum')
 return magic_sum

def sum_numpy(a):
  sum = 0
  for x in range(a.shape[0]):
    for y in range(a.shape[1]):
        print(x, y, ' *' , a[x][y] )
        sum = sum + a[x][y]
  return sum

def sum_3D(a):
  print('homework - sum elements of 3d numpy array')

  # use nested for loops
  #https://www.ict.social/python/basics/multidimensional-lists-in-python


def main():

    print('running: numpy_lists()')
    sum = numpy_lists()
    print(sum, ' is sum')
```

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

```
import numpy as np
import random
import sys  # for end or endline in nested food loops, python 3


def nested_loops():

  print(' nested loops function')
  #https://www.ict.social/python/basics/multidimensional-lists-in-python

  length = 3
  width = 4
  height = 3

  two_dimen = np.random.rand(length, width)
  print(two_dimen)

  first_row = two_dimen[0,:]
  second_col =   two_dimen[:,1]

  print(first_row, ' first_row')
  print(second_col, ' second_col')

  three_dimen = np.zeros((length, width, height))

  print(three_dimen)

  for i in range(length):
    for j in range(width):
      two_dimen[i,j] = i*j

  print (two_dimen)

      #print (k, end =' ')  # end refers new line
  print (' 3 nested ')
  for i in range(1,3):
    for j in range(1,3):
      for k in range(1,3):
        z = i*j * k
        print (z)  # end refers new line

  print (' 3 nested save to np array ')
```

```
  for i in range(length):
    for j in range(width):
      for k in range(height):
        three_dimen[i,j,k] = i*j * k
  print(three_dimen)

  #operations to select parts of 2d array


def main():

    print('running: nested_loops()')
    nested_loops()

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

Sum Matrices

Sum elements in a 2D Matrix example given using nested loops (to demonstrate nested loops)

Challenge: implement a function for 3D case using nested loops

Sum Matrix based on a condition

This code slide does not have a description.

Nested loops: 2D and 3D

Example of operations with nested loops

Advanced Numpy methods

This code slide does not have a description.

Load and save files

This code slide does not have a description.

Give change example

An example of how you would give change with different denominations.

cashier simulation

Lets simulate a give change cashier situation.

Sum list revisited

```
# Python 3 code to demonstrate - Sum elements matching condition using loop
#source: https://www.geeksforgeeks.org/python-sum-elements-matching-condition/
# initializing list
test_list = [3, 5, 1, 6, 7, 9]
print ("The original list is : " + str(test_list))
# using loop - Sum elements matching condition
# checks for odd
res = 0
for ele in test_list:
    if ele % 2 != 0:
        res = res + ele


print ("The sum of odd elements: " + str(res))

res = 0

for i in range(len(test_list)):
    ele = test_list[i]
    print(i, ele, ' * ')
    if ele % 2 != 0:
        res = res + ele
        print(res, ' ** ')


# printing result
print ("The sum of odd elements: " + str(res))
```

```
# Python 3 code to demonstrate - Sum elements matching condition
# using sum() + generator expression
#source: https://www.geeksforgeeks.org/python-sum-elements-matching-condition/
test_list = [3, 5, 1, 6, 7, 9]
# printing original list
print ("The original list is : " + str(test_list))
# using sum() + generator expression - Sum elements matching condition
# checks for odd
res = sum(i for i in test_list if i % 2 != 0)
# printing result
print ("The sum of odd elements: " + str(res))
```

Pandas data processing

```
import numpy as np
import pandas as pd

s = pd.Series([1, 3, 5, np.nan, 6, 8])
print(s, ' s series')

dates = pd.date_range('20130101', periods=6)
print(dates, ' dates')
```

source: https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

```
import numpy as np
import pandas as pd

data = np.array([['','Col1','Col2'], ['Row1',1,2], ['Row2',3,4]])

print(data)

print(pd.DataFrame(data=data[1:,1:], index=data[1:,0], columns=data[0,1:]))
```

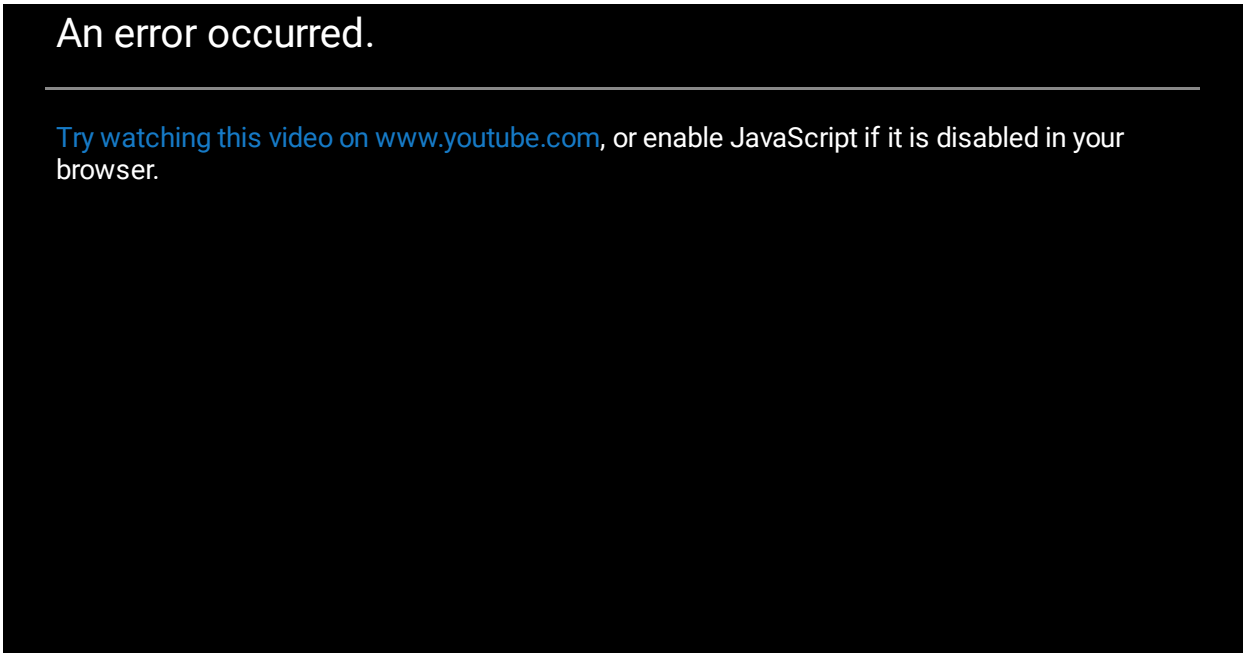
```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6]]))

# Using `iloc[]`
print(df.iloc[0][0])  #https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.
#https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/

# Using `iat[]`
print(df.iat[0,0])
```

source: https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python

tutorial: https://realpython.com/pandas-python-explore-dataset/

https://www.geeksforgeeks.org/python-pandas-dataframe/

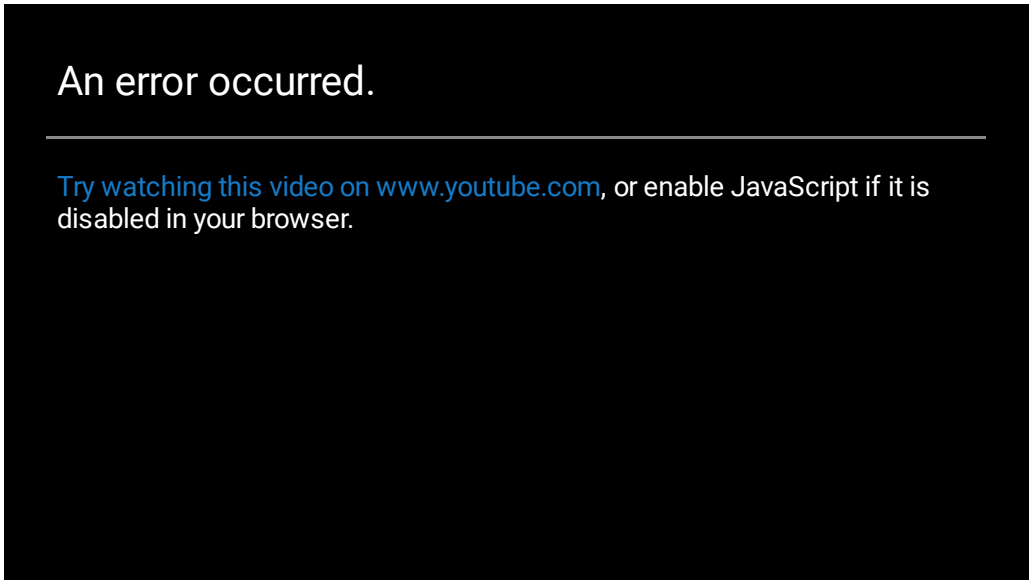
https://www.geeksforgeeks.org/python-pandas-working-with-text-data/


An error occurred.
Try watching this video on www.youtube.com, or enable JavaScript if it is disabled in your browser.

Data cleaning with Pandas

External tutorials here:

1. https://www.kaggle.com/regivm/data-cleaning-and-eda-tutorial
2. https://realpython.com/python-data-cleaning-numpy-pandas/



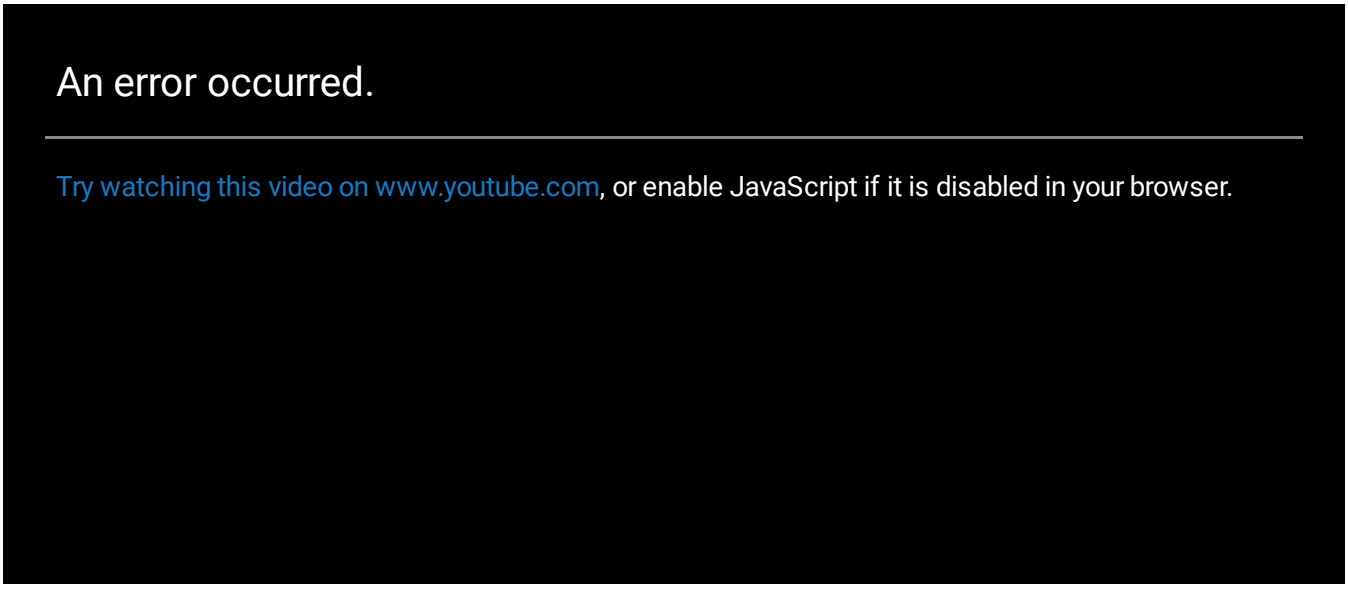
notebook for above video: https://github.com/KeithGalli/pandas

Object oriented programming

Classes

https://www.w3schools.com/python/python_classes.asp

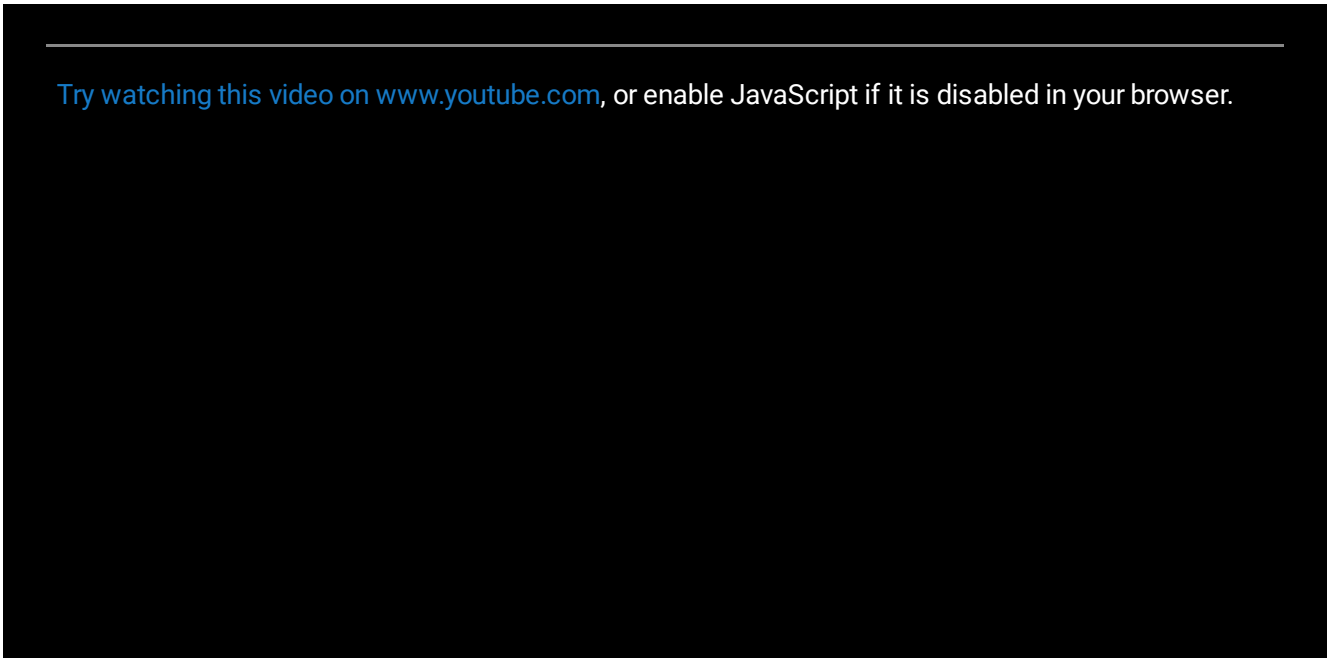
```
# source: https://www.csdojo.io/class

class Robot:
    def __init__(self, name, color, weight):
        self.name = name
        self.color = color
        self.weight = weight

    def introduce_self(self):
        print("My name is " + self.name)

# r1 = Robot()
# r1.name = "Tom"
# r1.color = "red"
# r1.weight = 30
#
# r2 = Robot()
# r2.name = "Jerry"
# r2.color = "blue"
# r2.weight = 40

r1 = Robot("Tom", "red", 30)
r2 = Robot("Jerry", "blue", 40)

r1.introduce_self()
r2.introduce_self()
```



Inheritance

https://www.w3schools.com/python/python_inheritance.asp

```
# source: https://www.csdojo.io/class2

class Robot:
    def __init__(self, n, c, w):
        self.name = n
        self.color = c
        self.weight = w

    def introduce_self(self):
        print("My name is " + self.name)


class Person:
    def __init__(self, n, p, i):
        self.name = n
        self.personality = p
        self.isSitting = i

    def sit_down(self):
        print("My name is " + self.name)


r1 = Robot("Tom", "red", 30)
r2 = Robot("Jerry", "blue", 40)

p1 = Person("Alice", "aggressive", False)
p2 = Person("Becky", "aggressive", True)

p1_robot_owned = r2
p2_robot_owned = r1

p1_robot_owned = r2



p1_robot_owned.introduce_self()
```



some other examples:

https://www.programiz.com/python-programming/object-oriented-programming

https://python.swaroopch.com/oop.html

intro OOP

Robot example revisted

process data

Example on how to process Iris dataset. Wine and Diabetes dataset are included for further analysis.

Exercise 1.1: BMI

Develop a Body mass index program using either Python or R.

Body Mass Index (BMI) is a simple index of weight-for-height that is commonly used to classify underweight, overweight and obesity in adults. It is defined as the weight in kilograms divided by the square of the height in meters (kg/m^2). For example, an adult who weighs 70kg and whose height is 1.75m will have a BMI of 22.9.

BMI = 70 kg / (1.75 m)^2 = 70 / 3.0625 = 22.9

Therefore the formula is:

BMI = mass (kg) / height (m)^2

The following table shows the International Classification by the World Health Organisation of **adult** underweight, overweight and obesity according to BMI (see [1]).

Classifcation	**BMI (kg/m)**
Underweight	<18.50
Severe thinness	<16.00
Moderate thinness	16.00 – 16.99
Mild thinness	17.00 – 18.49
Normal Range	18.50 – 24.99
Overweight	>=25.00
Pre-obese	25.00 – 29.99
Obese	>=30.00
Obese class I	30.00 – 34.99
Obese class II	35.00 – 39.99
Obese class III	>=40.00

BMI is also used as a screening tool to identify possible weight problems for **children**, also known as BMI-for-age. Although BMI-for-age value is calculated with the same formula, the classification is different. The following table shows an example classification.

Classifcation	**BMI (kg/m)**
Underweight	<19.99
Healthy weight	20.00 – 28.00
Overweight	29.00 – 31.00
Obese	>=32.00

Based on these classifications, if the BMI for a user is calculated as 22.9, then if the user is an adult then she would be classified as “Normal range”. However, if the user is a child (younger than 18), then she would be classified as “Healthy weight”.

Your program first needs to collect some information about the user:

- Date of birth (you can decide about the format of the date);
- Your program should allow the user to enter their height either in meters, inches and feet. The

user should be asked to specify whether they are entering their height in meters, or in inches and feet. If the user chooses to enter their height in inches and feet then your program has to make the appropriate conversion to meters. For example, the user can enter their height in meters, for example 1.70 m, or the user can enter the height as inch or feet, for example 5 feet 7 inches, so you need to make the appropriate conversion. Remember 1 inch = 0.0254 meters and 1 foot = 0.3048 meters:

- Weight in kg, for example 53 kg;
- Your program should then display the BMI value and also should show how the user would be classified by WHO (e.g., overweight). If the user is adult or a child (younger than 18) then make sure that you use the correct classification;
- Your program should continue to run and calculate BMI as many times as the user would like, and the program should exit when the user explicitly specifies so.

Exercise 1.1: Solution 1

Exercise 1.2: BMI using OOP

Using object oriented programming, implement BMI program in Exercise 1.1 using Classes. You can use either Python or R for this exercise.

References

1. https://www.datacamp.com/community/tutorials/r-objects-and-classes
2. https://realpython.com/python3-object-oriented-programming/

Additional resources

Video and code available at Python Bootcamp: https://github.com/rohitash-chandra/python-bootcamp