Python : recap

Objects → Mutable , $a = 5$
Objects → Immutable , $a = (1,3,5)$, dictionaries

Char. of objects : type : number, string, list, ..
value : data value contained
identity : related with CS

Attributes : data, fuctions associated with them . Name of attrs follows the name of obj, follo by a dot

→ Data attributes : Data attached to an objects
→ Methods : Function attached to an object . Typically it performs a fuction on that obj

Example : import numpy as np
x = np.array([1,3,5])

Data attribute : x.shape # elmts in x
no operation

→ Method : x.mean( ) : mean of values
there is operation

Modules : Libraries of code, eg math, numpy, ...

dir(str) gives a list of all available methods for a string

help(name.upper) gives info about the method upper applied to the string name.

Numbers : Integer, floating point, complex.

• 2 + 5 , 2 * 5 , 2 ** 5 = 2^5 , 6/7 , 6//7 = parte entera de 6/7

• In the interactive mode , _ is the value of the last value returned.

, math.factorial (6) = 6!

• input random , random.choice ([———]) gives a random entry of [—]

• Booleans. True, False, and, or, not, < , <=, ==, >= , != (≠)

In lists, == compares elent by elent, whereas is compares whether the objects are the same.

Sequences. lists, tuples, range objects, ~~strings~~

S[0] , = location 1st , S[0:2] = list w/ loc. of 1st & 2nd

• Lists : Mutable sequences , strings are not.

S = [2, 4, 6, 8]

~~T~~ S.append (10) = [2, 4, 6, 8, 10] ← now S

T = [10, 14, 16] , S + T = [2, 4, ..., 16]

S.reverse ( ) , S = [10, 8, ... 2]

S.sort ( ) , S = [8, 2, 10, 6, 4]

(There is a function also for this: sorted ( ))

len (S) = 5 (length)

- Tuples : immutable sequences.

$T = (1, 3, 5, 7)$, can + and access $\rightsquigarrow T[0] = 1$

$x = 12.15$
$y = 75.2$, coordinate $= (x, y)$ is a tuple.

I can do $(c1, c2, c3, c4) = T$, and then $c1 = 1$, $c2 = 3$

Say now that $P = [(0,0), (5,9), \ldots, (4,21)]$ list of tuples :

Can do
```
for (a,b) in P :
    print (a,b)
```
(for a,b in P) also works

$c = (2)$ is an integer, $c = (2,)$ is a tuple w/ 1 object

T.count (7) counts the number of 7 in T.

sum (T) = 1 + 3 + 5 + 7

- Range : Immutable , it is a type!!

range (5) gives range (0, 5). To see the content do

list (range (5)) = [0, ..., 4]

list (range (1,6)) = [1, ... 5]

list (range (1,13,2) = [1, 3, 5, ... 11]

Strings : Immutable sequences of characters

S = "Python" , len(S) = 6 , S[0] = 'P' , S[-1] = 'n'

S[0:3] = 'Pyt' S[-3:0] = 'hon' (last 3 char)

"y" in S : True

"hello" + " word" = "hello word"

name = "Tina Fey" , name.replace("T", "t") = "tina Fey"

names = name.split(" ") (splits when there is space)

names = ["Tina", "Fey"]

"tone".upper() = 'TONE'

lower

" ".join(list of words) = word1 word2 word3 . . .

Sets : Unordered collections of "hashable" objects
means that you can use this for immutable obj.

↳ Frozen set : immutable

Set : mutable

ids = set([1, ..., 7]) turns the list [1,-7] into a set w/ 7 elmts

ids : {1, 7, ..}

ids.add(8) = {1, ..7, 8}

ids.pop( ) removes a random elmt of ids.

Say now ids = set(range(10)) , ids = {0, ..., 9}

males = set([1,3,6,7])

females = ids - males , female = {0,2,4,...}

everyone = male | female (| := ∪ , - := \ , & := ∩)

everyone & set([1,2,3])

word = "antidisestablishmentarianism".

~~set & word~~ letters = set(word), len(letters) gives the # of unique letters.

x.issubset(y) determines whether $x \subseteq y$.

Dictionaries: Maps from key objects to value objects.

Consists of a pairs Key : Value

↑ immutable

{ } curly braces

age = {"Tim": 29, "Jim": 31, "Pam": 27, "Sam": 35}

age["Pam"] = 27

age["Tim"] = age["Tim"] + 1

+= 1

names = age.keys( ) , it is a "dict_key" Type object.

age["Tom"] = 50 adds a new entry in the dict

!! In names "Tom" will be included! "View-object"

ages = age.values( )

Tom in age : True.

age.items( ) # return a dic_key objet, essencially a list of tuples [(name, age)]

Caution !! (Dynamic typing):

```
x = 3
y = x
x = 2
```

Result:
x = 2
y = 3

Because integers are immutable objects

Strings also imtble
Tuples also imtble

```
x = [1, 2, 3]
y = x
x[0] = 7
```

Result
x = [7, 2, 3]
y = [7, 2, 3]

!!

Lists are mutable objects
Dicts also mutable

To keep y as before, two possibilities
- y = list(x)
- y = x[:]

In both cass it creats a new object.

Remark:
- `==` equality elem-wise
- `is` equality as objects.

```
x = [1,2,3]
y = [1,2,3]
x == y : True
x is y : False
```

```
x = [1,2,3]
y = x
x is y : True
```

Statements: Return, import, pass (do nothing, just in some cases for syntactical reasons)

```
if ...
elif ...
else ...
```

For:

```
for n in range(5):
    print(n)
```
0 1 2 3 4

```
names = ["Pepe", "Juan", ...]
for name in names:
    print(name)
```
'Pepe' 'Juan' ...

```
for i in range(len(names)):
    print(names[i])
```
"Pepe" "Juan"

In other languages one does (because iterate over a list is not possible)

Can iterate over a dictionary as follows:

age = { "Ivan": 55, "Pepe" = 20, ... }

```
for name in age:
    print (name, age[name])
```

Ivan 55
Pepe 20

* Can also do ... in age.keys()

Or if I want the names to be showed alphabetically, ... in sorted(age.keys()).

sorted(___ , reverse = True) will order the list in reverse alphabetically and

List comprehension: q = [a**2 for a in range(10) if a%2 == 0].

Reading and writing files

```
for line in open("oso.txt"):
    print (line)
```

Soy
un
oso

← w/ extra spaces. This is because in the txt file it actually says

Soy\n
un\n
oso.

This can be avoided using rstrip() method:

```
for line in open("oso.txt"):
    line = line.rstrip()
    print (line)
```

Soy
un
oso

split("a") method takes a string and divides it when there's a "a".

Create a txt file:

```
F = open("output.txt", "w")        # "w" ← write
F.write("Python\n")
F.close()                          # ← to close
```

w+ : write & read

a+ : append

```
for i in range(1,11):
    F.write("The square of " + str(i) + " is " + str(i**2) + "\n")
```

F.read() returns a single string with the whole text (also \n).

F.readlines() returns a list with entries the lines of the file (also with \n, since they are strings)

lol.txt

```
This is line 1
This is line 2
_____ 3
```

```
Q = open("lol.txt", "r")
linea = Q.readlines(16)
print(linea)
linea = Q.readline(5)
print(linea)
linea = Q.readline(9)
print(linea)
```

```
This is line 1

This
is line 2
```

Eg: If a file has

```
Pedro, 22, 1.78
Jorge, 25, 1.67
   :     :    :
```

how to extract this data? Use split(",")!

~~[illegible]~~

~~[illegible]~~

```
D = open("datos.txt", "r")
Z = []
for line in D:
    currentline = line.split(",")
    Z.append(currentline)
```

(even better, `line.rstrip().split(",")`)

Functions

inputs

```
def add(a,b):
    mysum = a+b
    return mysum
```

```
def add_and_sub(a,b):
    mysum = a+b
    mydiff = a-b
    return (mysum, mydiff)
```

⚠ Warning: For mutable objects, a function might change its value!

```
def modify(mylist)
    mylist[0] += 10
    return (mylist)
```

```
L = [1,3,5,7,9]
M = modify(L)
M = [10,3,5,7,9]
L = [10,3,5,7,9)
```

!!!

Example : Function that creates a password:

import random (random.choice([....]) selects randomly an elmt of the list)

```
def password(length):
    pw = str()
    chars = "abcd... yz01...9"
    for i in range(length):
        pw = pw + random.choice(chars)
```

Scope rules : It might be that at different points of a program there are several functions or variables with the same name (eg inside a function and outside). Python follows

Local	- Current function
Enclosing function	- Function that called the current function, if any
Global	- Module that the function was defined
Built-in	- Python's built-in name space

If Python cannot resolve a name, error!

Remark: Argument : Object that is passed to a function as its input when the function is called

Parameter : Variable that is used in the function definition to refer to that argument

(n = 300)

↑ variable (name/label of obj) ↖ object

Example: Look how it works depends on mutable / immutable objects:

```
def update(n, x):
    n = 2
    x.append(4)
    print('update:', n, x)

def main():
    n = 1
    x = [0, 1, 2, 3]
    print('main:', n, x)
    update(n, x)
    print('main', n, x)

main
```

Returns

```
main: 1 [0, 1, 2, 3]
update: 2 [0, 1, 2, 3, 4]
main: 1 [0, 1, 2, 3, 4]
```

n does not change its original value,

but x does

Object-oriented programming: Creating new object type ~~extending~~ extending another one.

~~Class~~ (classes). This is analogous to define a function, but now this allows us to define new methods, to new .something().

Eg: Want to create a method ~~that~~ remove_min which removes the minimum elmt of a list:

```
class MyList(list):
    def remove_min(self):
        self.remove(min(self))
```

```
x = [2, 5, 10, 1]
y = MyList(x)
dir(x) = (all methods for lists)
dir(y) = (                    ) + remove_min
y.remove_min()
# 1 disappeared from y
```

Numpy

Provides new data type.

```
import numpy as np

zv = np.zeros(5)        # zero vector of dim 5
zm = np.zeros((5,3))    # zero matrix of dim 5x3
     np.empty

x = np.array([7,-1])
A = np.array([[1,2],[3,4]])

A.transpose()
```

x[1] = -1

A[0,0] = 1

A[:,1] = array([2,4])
column

A[0,:] = array([1,2])

⚠ : [1,2] + [3,4] = [1,2,3,4]

np.array([1,2]) + np.array([3,4]) = array([4,6])
||
z1

z1 + 1 = array([2,3]).

~~z[[0,1]] = 1~~

v = np.array([3,5,8,16,24])

v[[1,2]] = array([5,8])

list of indices

I can use booleans as array in the following sense:

z_1 = np.array([1,3,5,7,9])

$z_1 > 6$ ⇝ gives me an array([False, False, False, True, True])

Then I can do

$z_1[z_1 > 6]$ ⇝ gives an array([7,9]), ie, returns an array only with the True entries

↓

This is a "logical array"

Warning! ⚠ : Slicing an array using colon $v[3:5]$ gives a view of the object, but does not create a new object.

Using indexing does create a copy:

z_1 = np.array([1,3,5,7,9])

$w = z_1[0:3]$

$w[0] = 3$

w = array(3,3,5)

z_1 = (3,3,5,7,9) !!

z_1 as before

ind = np.array([0,1,2]) # indexes

$w = z_1$[ind]

$w[0] = 3$

w = (3,3,5)

z_1 = (1,3,5,7,9)

Better: z_1 as before

w = np.copy(z_1[0:3])

w[0] = 3

`np.linspace(0, 100, 11)` gives an array of 11 points spaced equally between 0 & 100, ie, 0, 10, 20 ...

`np.logspace(1, 4, 4)` gives an array of 4 points spaced "log. equally" bt. 10^1 & 10^4, ie, $10^1, 10^2, 10^3, 10^4$.

`A = np.array([[1,2,3],[4,5,6]])`, A.shape : (2, 3)

A.size : 6

`x = np.random.random(10)` # randomly chooses 10 num in [0, 1)

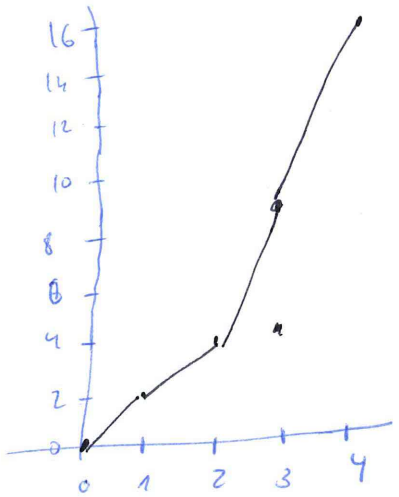
`np.any(x > 0.9)` ⇝ True # is any of its elts >0.9?

`np.all(x >= 0.1)` ⇝ Are all of its elts ≥0.1?

Matplotlib & Pyplot

```
import matplotlib.pyplot as plt

plt.plot([0, 1, 4, 9, 16])
```

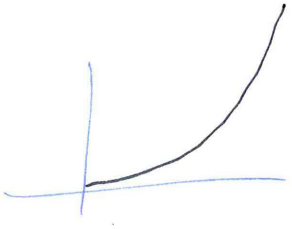


(use ; at the end to not show the object ie the name of the object)

ie it plots the piecewise linear function passing through (0,0), (1,1), (2,4), (3,9), (4,16)

```
x = np.linspace(0, 10, 20)
y = x**2
plt.plot(x, y)
```

⇝ gives



smoother

x = (as before)

y1 = x ** 2.0

y2 = x ** 1.5

b = blue

g = green

r = red, etc

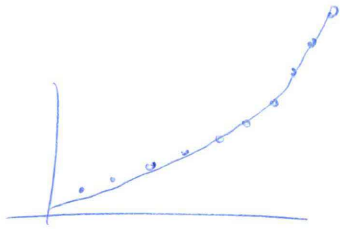
o = circles

s = squares

blue nodes as circles

solid line

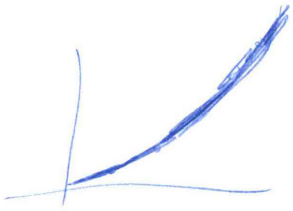
```
plt.plot(x, y1, "bo-")
```



```
plt.plot(x, y1, "bo-", linewidth=2, markersize=4)
```

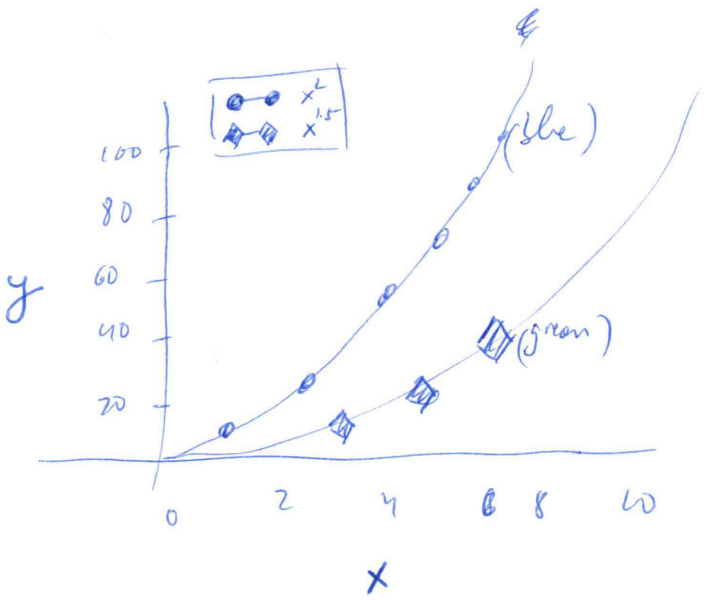
linewidth = 2 → line

markersize = 4 → nodes



Example:

```
x = np.linspace(0, 10, 20)
y1 = x ** 2.0
y2 = x ** 1.5
plt.plot(x, y1, "bo-", linewidth=2, markersize=12, label="$x^2$")
plt.plot(x, y2, "gs-", label="$x^1.5$")
plt.xlabel("$x$")
plt.ylabel("$y$")
plt.axis([-0.5, 10.5, -5, 105])
plt.legend(loc="upper left")
plt.savefig("myplot.pdf")
```



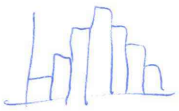
Remark : plt.loglog(x, y, "bo-") returns a plot with both axes in log scale.

plt.semilogx( ) ——— with log scale in x axes

plt.semilogy( ) ——— y

np.logspace(-1, 1, 20) gives log-eq. space pts from 10^{-1} to 10^{1}.

Histograms :

```
x = np.random.normal(size = 1000)        # generate an array of 1000 numbers
                                           get from the standard normal distribution
                                           N(0,1).
plt.hist(x);   # returns 
```

```
plt.hist(x, normed = True);              # instead of plotting the number of observations, it plots
        density                            the proportion of observations
```

```
plt.hist(x, normed = True, bins = np.linspace(-5, 5, 21))   # set differences btw of
                                                              0.5, in 20 bins
```

```
                    ~ Γ(2,3)
x = np.random.gamma(2,3,100000)
```

```
plt.hist(x, bins = 30, cumulative = True)   # gives the cumulative histogram,
                                              eq
        (histtype = "step")
                ↑ # appearance of the histogram
```

Can put many plots into one:

```
plt.figure()
plt.subplot(221)        # 22 indicates 2x2
plt.hist(x, ...)
plt.subplot(222)
  ⋮
```

223

224

Random processes

```
import random
random.choice([0,1])      # returns 0 or 1 randomly
```

} or any list, or a range object, eg range(1,7) for 1,...,6.

```
random.choice(random.choice([range(1,7), range(1,9), range(1,11)]))   # encodes the
```

situation where you have 3 dice of 6, 8 & 10 faces, you choose one at random and you roll the die.

Example: Roll the dice a number of times and plot the results. (say 100 times)

```
rolls = []
for k in range(100):
    rolls.append(random.choice([1, ...6]))
plt.hist(rolls, bins = np.linspace(0.5, 6.5, 7);
```

Example: What probability distribution follows the random variable consisting of the sum of rolling 10 dices?

```
ys = []
for rep in range(500000)
    y = 0
    for k in range(10):
        x = random.choice([1,...6])
        y = y + x
    ys.append(y)
plt.hist(ys);  # normal distribution, (Central Limit Theorem)
```

—o—

np.random.random(5) creates a list w/ 5 numbers chosen from the uniform dist in [0,1)

_____ ((5,3)) _____ matrix w/ 5×3 _____

np.random.normal(0,1) # chooses a number from N(0,1)

_____ (0,1,(6,8)) # ~~choose~~ creates a 6×8 matrix of random numbers.

np.random.randint(1,7) # chooses a random integer 1,...6.

A = np.array(Some matrix)

np.sum(A) # sum all entries

np.sum(A, axis=0) # sum the elents in every column (so over the rows)

np.sum(A, axis=0) # sum the elents in every row (so over the columns)

Example (The previous example revisited):

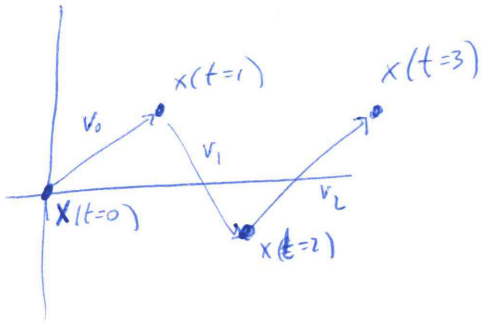
```
X = np.random.randint(1,7,(100000,10))
Y = np.sum(X, axis=1)
plt.hist(Y).
```

• How to measure time?

```
import time
start_time = time.clock()
  ⋮
end_time = time.clock()
total_time = end-time - start_time
```

• Random walk : A walker moves in $\mathbb{R}^2$ randomly at times $t=0, 1, 2, 3, \ldots$.

$$x(t=k) = x(t=0) + \sum v_i .$$




```
np.cumsum(matrix, axis=0 or 1)   # returns a matrix with partial sums in every column/row
np.concatenate(a1, a2)           # concatenate two np arrays
```

Example: Plot a 5-iteration random walker starting from $(0,0) \in \mathbb{R}^2$:

```
X_0 = np.array([[0],[0]])
delta_X = np.random.normal(0, 1, (2,5))
X = np.concatenate((X_0, np.cumsum(delta_X, axis=1)), axis=1)
plt.plot(X[0], X[1], "ro--")
plt.savefig("rw.pdf")
```

More about ~~dos~~ string of characters

- F.read() creates a unique string with all the text, also \n. ~~How~~ How to remove those? A easy form is to use F.replace("\n", "").

⚠ Since strings are immutable, this does not change F, one has to do F = F.replace(...)

⚠ In some computers, there are \r also hidden. Better do F = F.replace("\r", "") as well

- If you create a fuction and want to create a "help" entry, this can be done by creating a "docstring": after def ... : , write """ This is the manual ... """.

Remark. A safer way to read files is

```
with open("dna.txt", "r") as F:
    seq = F.read
```

- Count words from a text

```
from collections import Counter
text = "hola me llamo Pablo Casado y he perdido cinco elecciones en un año".
word_counts = Counter(text.split(" "))   # creates (almost) a dict with keys the different words and values the frecuency of the words.
```

Remark: To turn text into a string with no capital letters or punctuation, do

```
text = text.lower()
skip = ['.', ',', ';', ':', '"', "'", '(', ')']
for ch in skip:
    text = text.replace(ch, "")
```

- Read a book and save result

```
with open("book.txt", "r", encoding="utf8") as current_file:
    text = current_file.read()
    text = text.replace("\n", "").replace("\r", "")
```

- Find fragment in a book:

```
ind = text.find("What's in a name?")
print(text[ind:ind+100])        # prints the 100 char after the start of the sentence.
```

- Navigate file directories

```
import os
os.listdir("./Descargas")       # list files in ./Descargas
                                #   ↓ where you already are
```

what if I have many directories English, German, ... each of them w/ books of diff (as more folders) authors Shakespeare.... and inside those the actual .txt's files with the books? Can do

```
for language in os.listdir("./Books"):
    for author in os.listdir("./Books/" + language):
        for title in os.listdir("./Books/" + langage + "/" + author):
            inputfile = "./Books/" + langage + "/" + author + "/" + title
```

(forgot the identation)

• Create table of data with Pandas

```
import pandas as pd

table = pd.DataFrame (columns = ("name", "age"))

table.loc [1] = "James", 22
table.loc [2] = "Pete", 30
```

Result:

	name	age
1	James	22
2	Pete	30

```
table.head ( )
table.tail ( )
```
} # gives the first / last 5 lines

Result: "hole".capitalize ( ) returns "Hole"

```
table.name  # gives a list with all entries of the column 'name'.

table = pd.read_csv ("data.txt", index_col = 0)  # reads a csv table
```

More numpy

• Distance bet. pts:

```
import numpy as np
p1 = np.array ([1,1])
p2 = np.array ([4,4])
p2 - p1  ⇝ array ([3,3])
np.power (p2 - p1, 2)  # does (3², 3²), to return array ([9,9])
np.sqrt (np.sum (np.power (p2 - p1, 2)))
```

- Find the mode of a list (most repeated value)

```
import scipy.stats as ss

votes = [1, 2, 3, 3, 1, 2, 1, 33, 3, 3]

mode, count = ss.mstats.mode(votes)
```

mode: value; count: number of times it appears

- Submatrices

Given a np.array A (matrix), one can get a submatrix by doing A[1:2, 0:2] eg.

the first column is A[:, 0], second column is A[:, 1], etc.

A.shape[0] returns # rows; A.shape[1] returns # columns.

- Arrange

np.arrange(0, 10, 2, [dtype = int]) #returns an np array [0, 2, 4, 8].

np.arrange(10).shape(5,2) gives the matrix $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \end{pmatrix}$

np.repeat(n, k) gives an array (n, n, n, ... n) k times

- Sort (Ordering) a list

```
a = np.array([2, 5, 8, -4, 3, 6])
```

np.sort(a) returns the ordered list

np.argsort(a) returns a list with the indices corresponding to the sorted elements, ie

if $a = (x_0, \dots, x_n)$, and np.sort(a) = $(x_4, x_3, x_0, x_2, \dots)$ then

np.argsort(a) returns (4, 3, 0, 2, ...)

- A list with the 3 largest numbers of a can be got by doing

```
ind = np.argsort(a)
a[ind[0:3]]
```

Example (KNN algorithm): Given two sets of pts in $\mathbb{R}^2$ $A=\{x_1,\dots x_m\}$, $B=\{y_1,\dots y_m\}$, and a pt $p\in\mathbb{R}^n$, a function which determines to which set the pt p is the closest, based on the nearest $K\in\mathbb{N}$ points in $A\#B$.

```
def distance(p,q):          (easy)

def find_nearest_nbhds(p, points, K)      # Find the K nearest neighbs and return indices in points

    distances = np.zeros(points.shape[0])

    for i in range(len(distances)):

        distances[i] = distance(p, points[i])

    ind = np.argsort(distances)

    return ind[:K]

def knn_predict(p, points, outcomes, K=5)      # points is supposed to be A#B and outcomes [0,0,...0,1,....1)
                                                                                                  n   m
    ind = find_nearest_nbhds(p, points, K)

    mode, count = ss.mstats.mode(outcomes[ind])

    return mode.
```

- `ss.norm(μ, σ).rvs(n, m)` returns a $n \times m$ matrix with entries chosen at random $\sim N(\mu, \sigma)$

- Enumerate

`seasons = ['spring', 'autumn', ...]`

`list(enumerate(seasons))` returns `[(0, "spring"), (1, "autumn"), ...]`

(0 = index, "spring" = element)

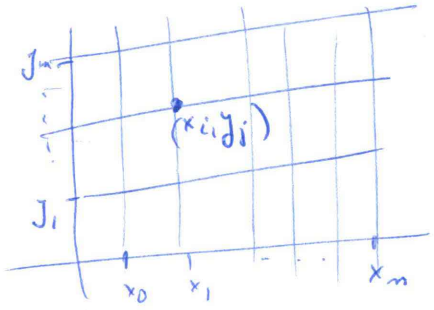
- Meshgrid : Let $x = (x_1 \dots x_n)$, $y = (y_1 \dots y_m)$

`xx, yy = np.meshgrid(x, y)` (np arrays) returns two matrices:

$$xx = \left(x_i \right)_{\substack{i=1\dots n \\ j=1\dots m}}$$

$$yy = \left(y_j \right)_{\substack{i=1\dots n \\ j=1\dots m}}$$

(so xx has repeated rows, yy has repeated columns)



let $A = \left((x_i, y_j) \right)_{\substack{i=1\dots n \\ j=1\dots m}}$

More on Pandas

import pandas as pd

Two main objects: Series & DataFrame

- Series: 1-dim array-like
- DataFrame: 2-dim array-like

• Series

```
x = pd.Series([6,8,3,6])
```

⟶

indices	values
0	6
1	8
2	3
3	6

Can define your own index labels:

```
x = pd.Series([6,8,3,6], index=["q","w","e","r"])
```

⟶

q	6
w	8
e	3
r	6

Can access to the value by doing `x["w"]` or `x[["w","r"]]`.

• If `age = {"Tom":50, ...}` is a dict, get a pd Series by doing

```
x=pd.Series(age)
```

⟶

Tom	50
⋮	⋮

• DataFrame

: Before we saw one way. Can also turn a dict into DataFrame, by using lists in the values:

```
data = {'name':['Tom',...], 'age':[50,...], 'ZIP':['06008',..]}
x=pd.DataFrame(data, columns=["name","age","ZIP"])
```

	name	age	ZIP
0	Tom	50	06008
1			
2			
⋮			

You can access a single column by doing `x["name"]` or `x.name`.

- How to change the indices 0, 1 ...? By doing

`x.reindex(["Person1", ...])`

- If x, y are pd Series of the same dim, then `x+y` sums the entries w/ the same labels, and returns NaN if a label appears only in one of x or y.

- Read files: either is it in csv or a txt with data separated by , 's, do

`x = pd.read_csv("datos.txt")`

To add a column from another file,

`x["new_column"] = pd.read_csv("new_datos.txt")`

⚠ `x["new_colum"]` adds a colum, `x.loc[7]` adds a new row

`x.iloc[0:10]` slices the first 10 rows

`x.iloc[0:10, 1:3]` ——— and 2nd, 3rd columns.

`x.columns` #lists the name of the columns.

- Compute correlations

Let x be a pd.DataFrame object w/ colums some attributes of some axis of data.
Would like to compare two such attributes

```
corr_x = pd.DataFrame.corr (x)    # compute correlation matrix!
```

Graphical plotting:

```
import matplotlib.pyplot as plt
plt.figure (figsize = (10,10))
plt.pcolor (x)        # actually plots the content
plt.colorbar ()       # the legend for the colors
plt.savefig ("...pdf").
```

Remark. x.transpose () gives the actual transpose of the table.

- Spectral co-clustering: Attempt to find clusters of related data (eg words from book, and find clusters by language).

```
from sklearn.cluster.bicluster import SpectralCoclustering
model = SpectralCoclustering (n_clusters = 6, random_state = 0)
model.fit (corr_x)
model.rows_    # returns a (n_clusters x (#rows in x)) -matrix with True/False
                                                  ↑ the data
```

```
np.sum(model.rows_, axis=1)   # returns an array with the # of elements in each cluster
```

Examples with Pandas

```
birddata = pd.read_csv("bird_tracking.csv")
ix = birddata.bird_name == "Eric"   # returns the indices corresponding to the value "Eric" in the column "bird_name".
x,y = birddata.longitude[ix], birddata.latitude[ix]   # returns arrays with the columns long, lat corresponding to ix, i.e. Eric.

plt.figure(figsize=(7,7))
plt.plot(x,y,".")

bird_names = pd.unique(birddata.bird_name)   # returns an array w/ the unique entries of the column bird_name (i.e. without repetitions)
```

Example:

```
bird_names (as before)

for bird in bird_names:
    ix = birddata.bird_name == bird
    x,y = bird_data.longitude[ix], bird_data.latitude[ix]
    plt.plot(x,y,".")
plt.xlabel("Longitude"); plt.ylabel("Latitude")
plt.legend(loc="lower right")
```

Useful trick : How to ignore the NaN's of a list? (This eg gives error if you want to plot): let x be an array with numbers and some NaN's. Then

```
ind = np.isnan(x)          # ind returns an array with True if a value is NaN
plt.hist(x[~ind])          # ~ind is the complement of ind, so we only plot the "False" values in ind.
```

- Panda can also plot histograms:

```
bird_data.speed_2d.plot(kind='hist', range=[0,30])
plt.xlabel("2D speed")
```

Good point of this : Do not have to deal with NaN's.

- Handling time with Datetime

```
import datetime
datetime.datetime.today()      # returns a datetime.type obj datetime.datetime(2020,6,18,11,43,50,)
                                                                                  Y M D H M S
time_1 = datetime.datetime.today()
time_2 = ____________
time_2 - time_1   ~> returns the time it has passed (timedelta object)
```

- Say we have a string "2013-08-15 00:18:08+00" (with an x over it). How to deal with the date from this?

```
datetime.datetime.strptime(x[:-3], "%Y-%m-%d %H:%M:%S")
```

- How to add a list to a pandas DataFrame?

```
birddata["newcolumn"] = pd.Series(x, index=birddata.index)
```

↑ the existing DF — ↑ the list

- How many days have passed? I can divide timedelta objects: ~~Cannot~~

~~Traceback~~

```
time_2 - time_1 / ~~datetime~~ datetime.timedelta(days=1)
```

Graphs with NetworkX

```
import networkx as nx
G = nx.Graph()          # creates an empty graph
G.add_node(1)           # creates a node labelled w/ 1
G.add_nodes_from([2,3,"u","v"])
G.nodes()               # show the nodes
G.add_edge(1,2)         # adds a node   1 —— 2
G.add_edges_from([(1,5),("u","w")])   # ⚠ nodes are added automatically!
G.remove_node(2)        # edge
G.remove_nodes_from([4,5])   # edges
G.number_of_nodes()     # edges
```

• Plotting a graph

```
import matplotlib.pyplot as plt

nx.draw(G, with_labels=True, node_color="lightblue", edge_color="grey")

G.degree()   # returns a dict-like object w/ the list of nodes & degrees
```

• Erdös-Rényi random graphs

: Parameters (N, p): N = # graphs, p = probability that there is a graph for every pair of nodes

```
from scipy.stats import bernoulli

bernoulli.rvs(p=0.2)        # bernoulli distr, returns 0 or 1 w/ prob p

G = nx.Graph(); N=20; p=0.2
G.add_nodes(range(N))
for nod1 in G.nodes():
    for nod2 in G.nodes:        # can be omitted
        if bernoulli.rvs(p=p) == True:
            G.add_edge(nod1, nod2)
```

(creates a directed graph, if not directed should add (and nod1 < nod2))

• Degree distribution

```
def deg_distr(G):
    deg_sequence = [d for n, d in G.degree]
    plt.hist(deg_sequence, histtype="step")
    plt.xlabel(...); plt.ylabel(...); plt.title(...)
```

Def: Let G be a graph. The *adjacent matrix* of G is $A = (a_{ij})$, $n \times n$ matrix,

n = # nodes, and $a_{ij} = \begin{cases} 0, & \text{if } \nexists \text{ edge bet. } v_i \& v_j \\ 1, & \exists \end{cases}$

- ofc G can be reconstructed from A. To do this w/ Python

```
G = nx.to_network_graph(A)
```

- Connected components of a graph

How to get the largest conn comp? Two ways:

(1) `g = nx.connected_component_subgraphs(G).__next__()`

(2) `g = max(nx.conneted_compont_subgraphs(G), key = len)`

Statistics with Python

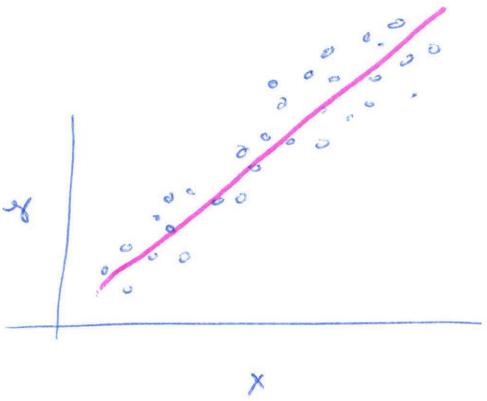
```
import numpy as np
      scipy as ss
      matplotlib.pyplot as plt
```

```
x = 10 * ss.uniform.rvs(size = n)      # creates an array w/ n values chosen from
                                         the uniform distribution 0,10)
y = 5 + 2 * x + ss.norm.rvs(loc = 0, scale = 1, size = 50)
                noise form ~ N(0,1)
```

Can plot this by doing

```
plt.figure()
plt.plot(x, y, "o", ms=5)
xx = np.array([0,10])
plt.plot(xx, 5+2*xx)
plt.xlabel("x"); plt.ylabel("y")
```

ms = marker size



• Least squares problem: Given $Y = \beta_0 + \beta_1 X$ and data $\{(x_i, y_i)\}$, want to build $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \hat{x}$ which minimizes $\sum e_i^2$, where $e_i = \hat{y}_i - y_i$.

```
import statsmodels.api as sm
mod = sm.OLS(y, x)
est = mod.fit()
print(est.summary())
```

for estimation for $Y = \beta \cdot X$

```
X = sm.add_constant(x)
mod = sm.OLS(y, X)
est = mod.fit()
print(est.summary())
```

for $Y = \beta_0 + \beta_1 X$

• scikit-learn for linear regression

```
x1 = 10 * ss.uniform.rvs(size=500)
x2 = ____________________
y = 5 + 2 * x1 - 1 * x2 + ss.norm.rvs(loc=0, scale=1, size=500)

X = np.stack([x1, x2], axis=1)   # takes two arrays and stacks them together, ie, concadenates them, so you get a matrix. In this case as columns (axis=1)
```

Produces a 3d graphic:

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,0], X[:,1], y, c=y)
```

```
from sklearn.linear_model import LinearRegression
lnn = LinerRegresion(fit_intercept=True)
lm.fit(X, y)
lm.intercept_     # returns β0
lm.coef_[0]       # ____ β1
____[1]           # ____ β2
```

• How to "predict" values?

```
X0 = np.array([2,4]).reshape(1,-1)
lm.predict(X0)
```

- (Assessing model accuracy) : To evaluate the performance of the regression model, one has to quantify how much the predictions agree w/ the observed data, eg mean square error. Out of some data, we want to split it randomly and see how they perform differently and compare results.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.5, random_state=1)
lm.LinearRegression(fit_intercept=True)  # for the β0 term
lm.fit(X_train, y_train)  # get the model
lm.score(X_test, y_test)
```