introcode

June 26, 2023

1 Leçon de code

La programmation est devenue un outil essentiel du datascientist mais pas seulement. Beaucoup d'outils pointus sont open source mais uniquement accessibles à ceux qui savent programmer. Après la mise au point d'un modèle statistique, économique, il se pose souvent la question de la mise à jour fréquente des résultats, c'est à dire leur automatisation via la programmation.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 Après la prépas

Programmation, ENSAE, automatisation, emplois

1.1.1 ENSAE

- statistique, finance, économie, actuariat, data science
- utilisation massive des données
- plus de logiciel passe partout

1.1.2 Excel, Matlab, SAS, Python, Notebook

- tableur : difficile de passer à l'échelle
- SAS, Matlab : onéreux
- Python : open source, langage accessible sans être un expert, effort de design comme avec scikit-learn (INRIA)
- notebook : outils permettant de mélanger texte, formules, code et de le partager

1.1.3 Python, R : Python

- tous deux open sources
- python plus complet
- attire les développeurs, pas seulement les chercheurs
- R plus proche de matlab

1.1.4 Installation

- Anaconda

1.1.5 A distance

- colab

1.1.6 Après la prépa

- code efficace
- test unitaires, exceptions ⟶ à connaître pour un entretien d'embauche
- packaging : exemple avec 2048
- manipulation de données, dataframes, graphes.

1.1.7 Un exemple : coût de l'algorithme ?

[2]:
```
def position_max(tableau):
    for i in range(0, len(tableau)):
        if tableau[i] == max(tableau):
            return i

mx = position_max([6, 7, 4, 11, -5, 4])
mx
```

[2]: 3

1.1.8 Ressource

- contenus en ligne xavierdupre.fr
- Google, Bing, Duck Duck Go, Qwant, Baidu, Yandex
- Stackoverflow
- openclassroom
- mail au professeur

1.1.9 COVID

- apprentissage pour tout le monde y compris pour les encadrants
- vidéo, pédagogie inversée
- projet en groupe : application Flask

1.2 Cours

1.2.1 Objectif du cours

- Test unitaire
- Calcul matriciel avec numpy
- Culture algorithmique
- Etre capable de réaliser une application Flask (web) qui récupère des données pour faire des statistiques.
- Etre capable de reproduire ou réutiliser l'algorithme décrit dans un article ou implémenté sur github

1.2.2 Organisation

- Répartition en TDs
- Le chemin vers les objectifs du cours sont différents selon les TDs

1.3 Quizz 1

1.3.1 affectation

[3]:
```
a = 3
```

1.3.2 types

[4]:
```
a = 3
p = 4.56
b = 'r'
c = (4, 6)
g = [5, 4]
d = {'a': 0, 'b': 1}
```

1.3.3 test

[5]:
```
h = 7
if h % 2 == 0:
    msg = 'pair'
else:
    msg = 'impair'
print(msg)
```

```
impair
```

1.3.4 boucle

[6]:
```
for element in [4, 5, 8]:
    print(element)
```

```
4
5
8
```

[7]:
```
it = 0
while it < 24:
    print(it)
    it += 5
```

```
0
5
10
15
20
```

1.3.5 fonction

[8]:
```
def area(l, w):
    return l * w

print(area(4, 5))
```

```
20
```

1.3.6 print / return ?

[9]:
```
def area(l, w):
    print(l * w)

print(area(4, 5))
```

```
20
None
```

1.3.7 import

[10]:
```
import math
from math import cos
cos(5) + math.sin(5)
```

[10]:
```
-0.6752620891999122
```

1.3.8 classes ?

[11]:
```
class Vase:
    def __init__(self, hauteur, diametre):
        self.hauteur = hauteur
        self.diametre = diametre
    def area(self):
        return self.hauteur * self.diametre * math.pi

v = Vase(5, 3)
print(v.area())
```

```
47.12388980384689
```

1.4 Quizz 2 : array, dataframe, graphe

On ne code plus le produit matriciel. Il est très difficile d'être plus rapide que numpy qui utilise des libraires telles que BLAS qui savent tirer parti des optimisations processeurs AVX, voire de processeurs différents GPU.

1.4.1 Produit matriciel

[12]:
```
import numpy
mat1 = numpy.array([[1, 2, 3], [4, 5, 6]])
mat2 = numpy.array([[1, -1], [-1, 1], [0, 0]])
mat1 @ mat2
```

[12]:
```
array([[-1,  1],
       [-1,  1]])
```

[13]:
```
mat2 @ mat1
```

[13]:
```
array([[-3, -3, -3],
       [ 3,  3,  3],
```

```
        [ 0,  0,  0]])
```

1.4.2 Dataframe

[14]:
```
import pandas
df = pandas.DataFrame([{'col1': 4.5, 'col2': "legend"},
                       {'col1': 4.5, 'col3': -8},
                       {'col1': 14.5, 'col3': -80, 'col2': 'note'}])
df
```

[14]:
```
   col1    col2  col3
0   4.5  legend   NaN
1   4.5     NaN  -8.0
2  14.5    note -80.0
```

[15]:
```
df.isna()
```

[15]:
```
    col1   col2   col3
0  False  False   True
1  False   True  False
2  False  False  False
```

[16]:
```
df.isna().astype(numpy.int64)
```

[16]:
```
   col1  col2  col3
0     0     0     1
1     0     1     0
2     0     0     0
```

[17]:
```
df[df['col1'] >= 10]
```

[17]:
```
   col1  col2  col3
2  14.5  note -80.0
```

1.4.3 Graphes

[18]:
```
rnd = numpy.random.randn(50, 3) @ numpy.array([[1, 0, 1], [0, 1, 0], [0, -3, 1]])
rnd[:5]
```

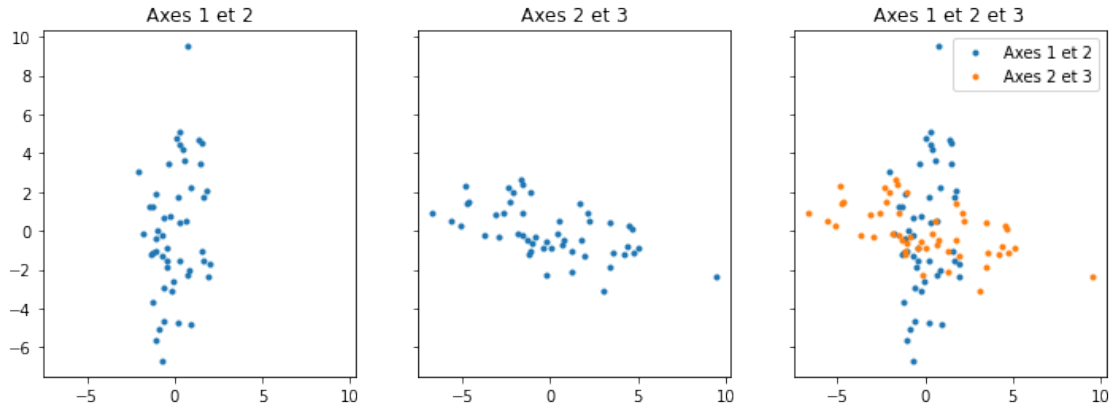
[18]:
```
array([[ 0.06607333,  4.74896639, -1.11535865],
       [-1.28671779,  1.2653011 , -2.10633038],
       [-0.40579191, -0.85452334, -0.34367823],
       [-1.10590692,  1.92898689, -1.25570647],
       [ 0.43969349,  4.22563223, -1.25008265]])
```

[19]:
```
%matplotlib inline
```

[20]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 3, figsize=(12, 4), sharex=True, sharey=True)
ax[0].plot(rnd[:, 0], rnd[:, 1], '.')
ax[0].set_title("Axes 1 et 2")
```

```
ax[1].plot(rnd[:, 1], rnd[:, 2], '.')
ax[1].set_title("Axes 2 et 3")

ax[2].plot(rnd[:, 0], rnd[:, 1], '.', label="Axes 1 et 2")
ax[2].plot(rnd[:, 1], rnd[:, 2], '.', label="Axes 2 et 3")
ax[2].set_title("Axes 1 et 2 et 3")
ax[2].legend();
```



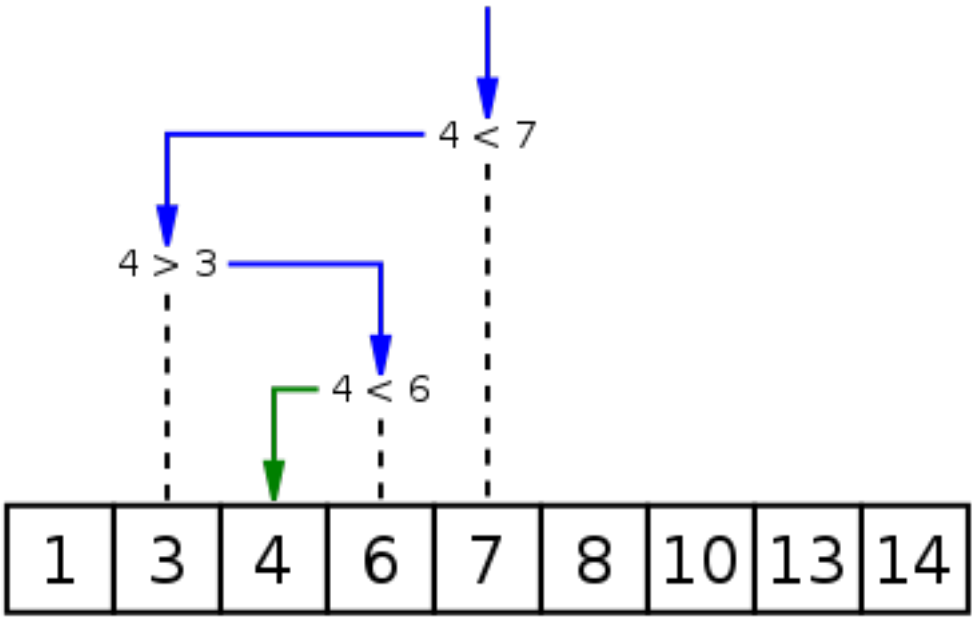
1.5 Quizz 3 : algorithme

[21]:
```
from IPython.display import SVG, Image
```

1.5.1 Recherche dichotomique

[22]:
```
Image("https://upload.wikimedia.org/wikipedia/commons/f/f7/Binary_search_into_array.
↪png")
```

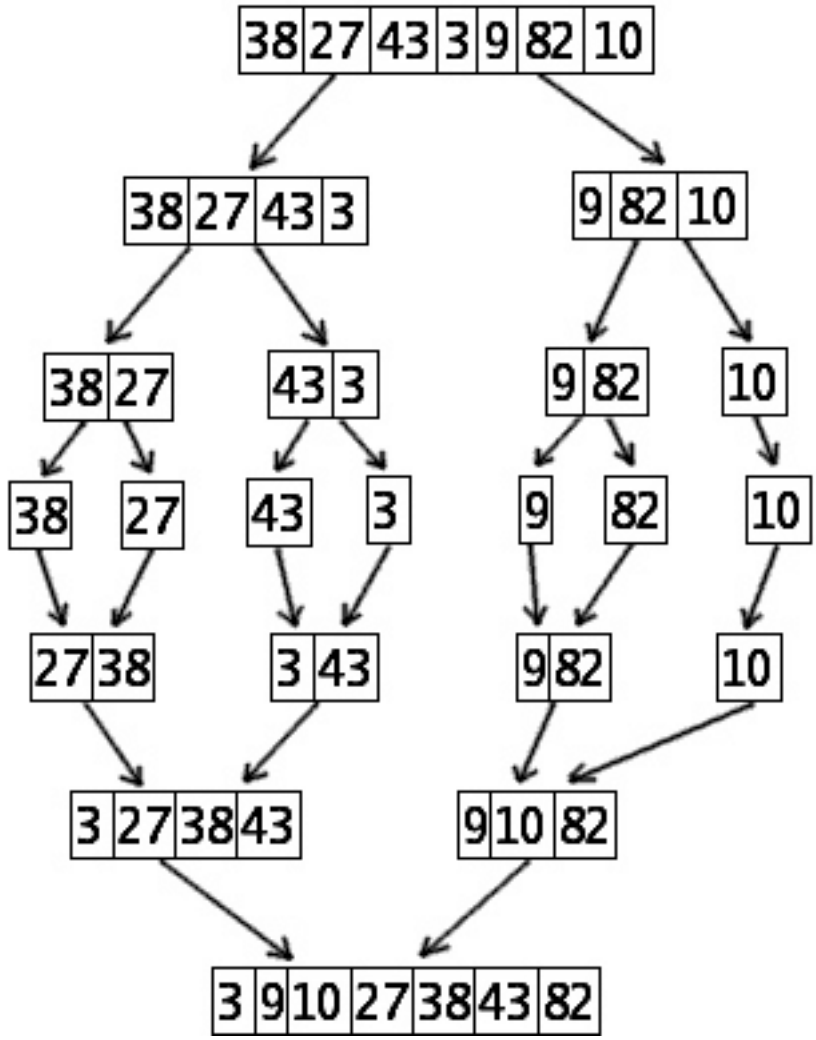
[22]:



1.5.2 Tri fusion

[23]:
```
Image("https://upload.wikimedia.org/wikipedia/commons/6/60/Mergesort_algorithm_diagram.
↪png")
```

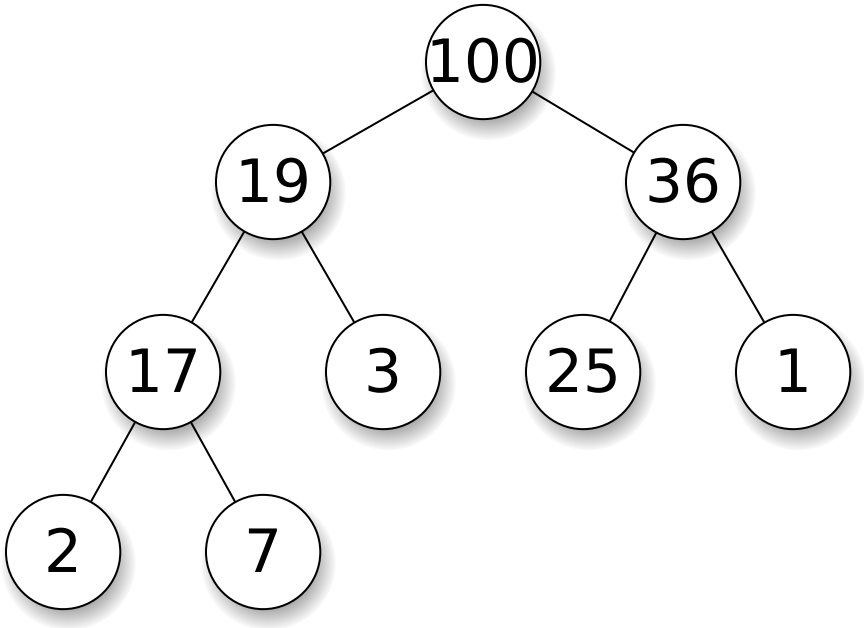
[23]:



1.5.3 Tas

[24]:
```
SVG("https://upload.wikimedia.org/wikipedia/commons/3/38/Max-Heap.svg")
```

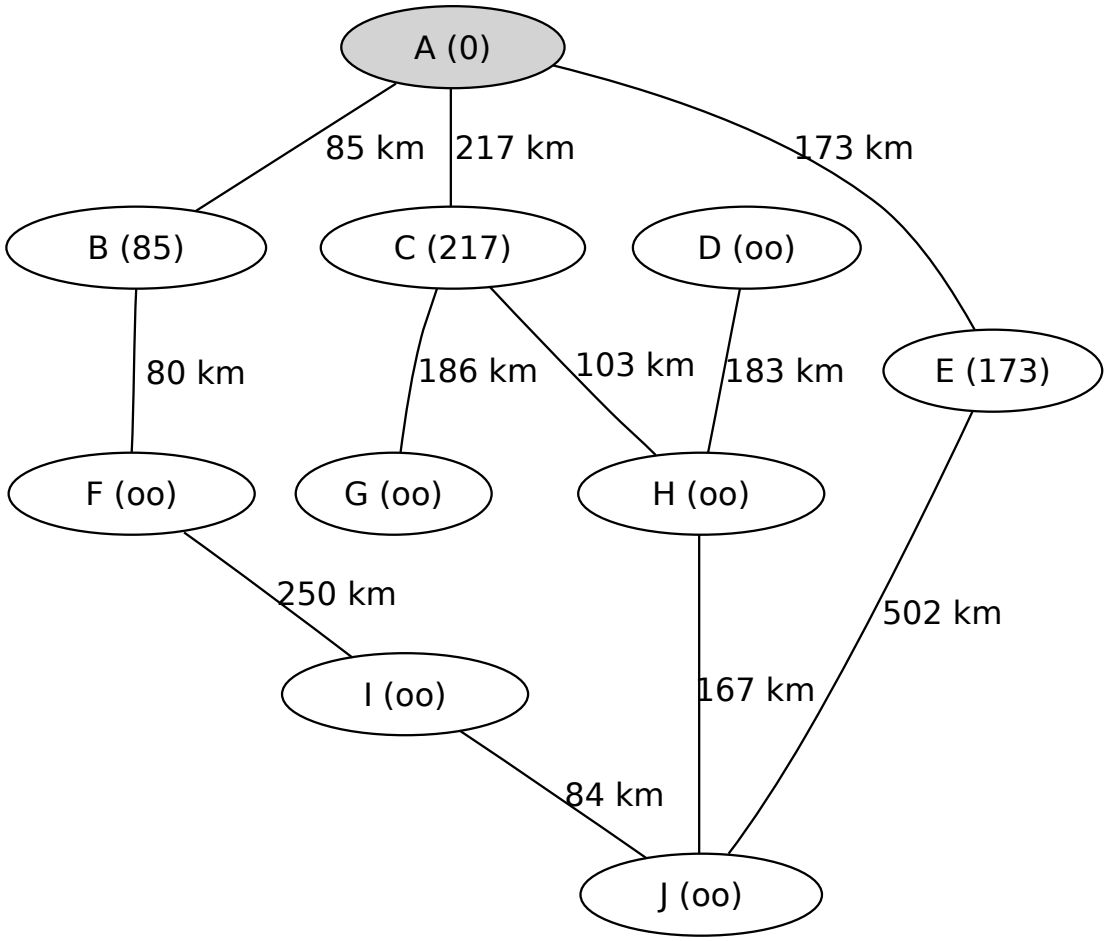
[24]:



1.5.4 Plus court chemin dans un graphe

[25]:
```
SVG("https://upload.wikimedia.org/wikipedia/commons/2/29/DijkstraBis01.svg")
```

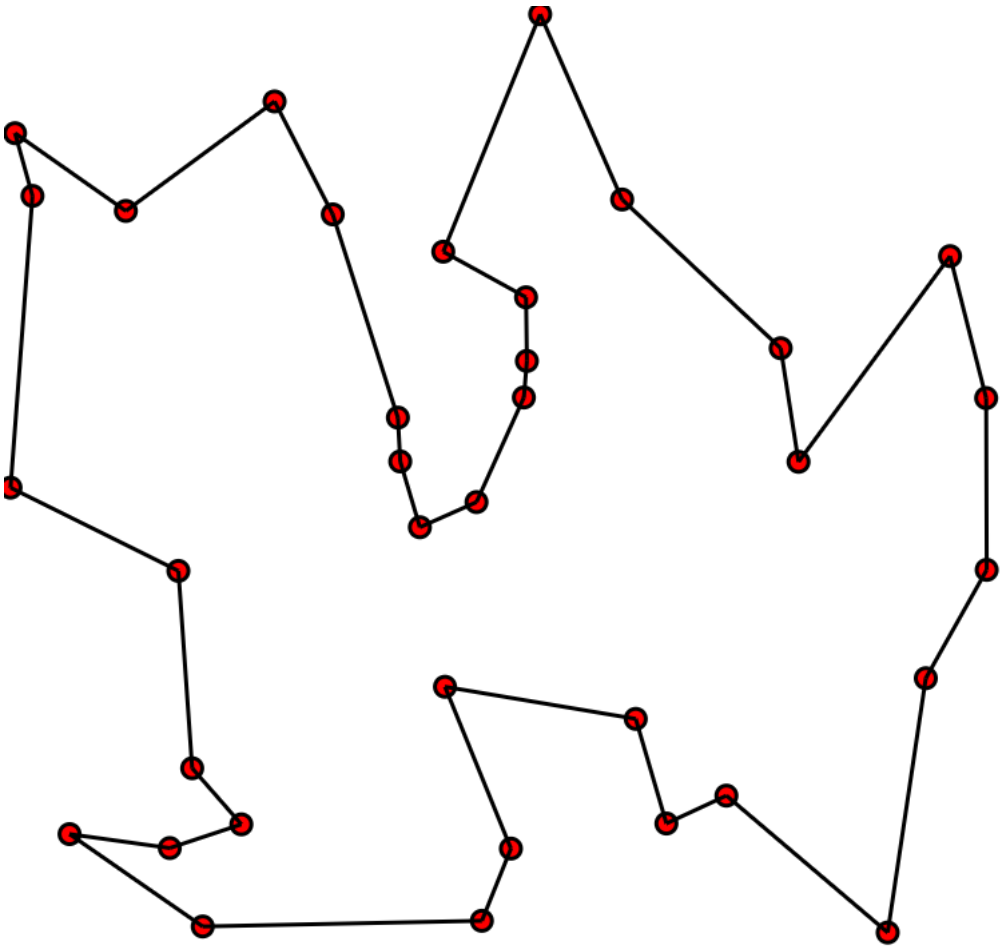
[25]:



1.5.5 Voygeur de commerce

[26]:
```
Image("tsp.png", width=400)
```

[26]:



1.5.6 distance d'édition

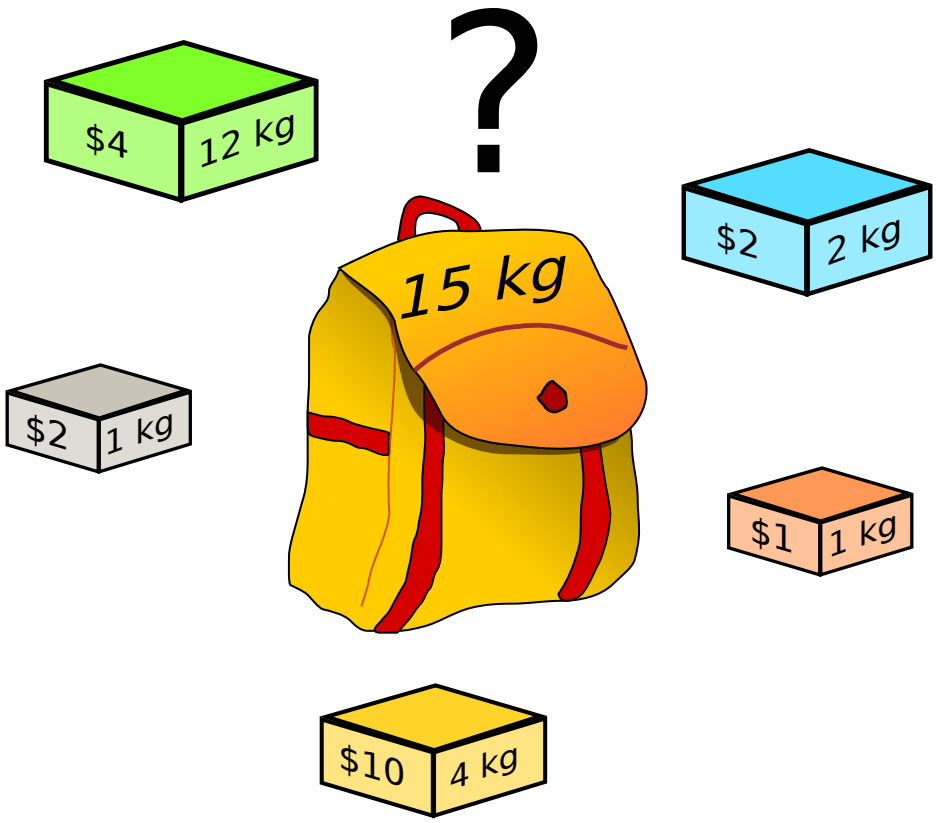
[27]:
```
Image("https://upload.wikimedia.org/wikipedia/commons/d/d1/
↪Levenshtein_distance_animation.gif")
```

[27]: <IPython.core.display.Image object>

1.5.7 Problème du sac-à-dos

[28]:
```
SVG("https://upload.wikimedia.org/wikipedia/commons/f/fd/Knapsack.svg")
```

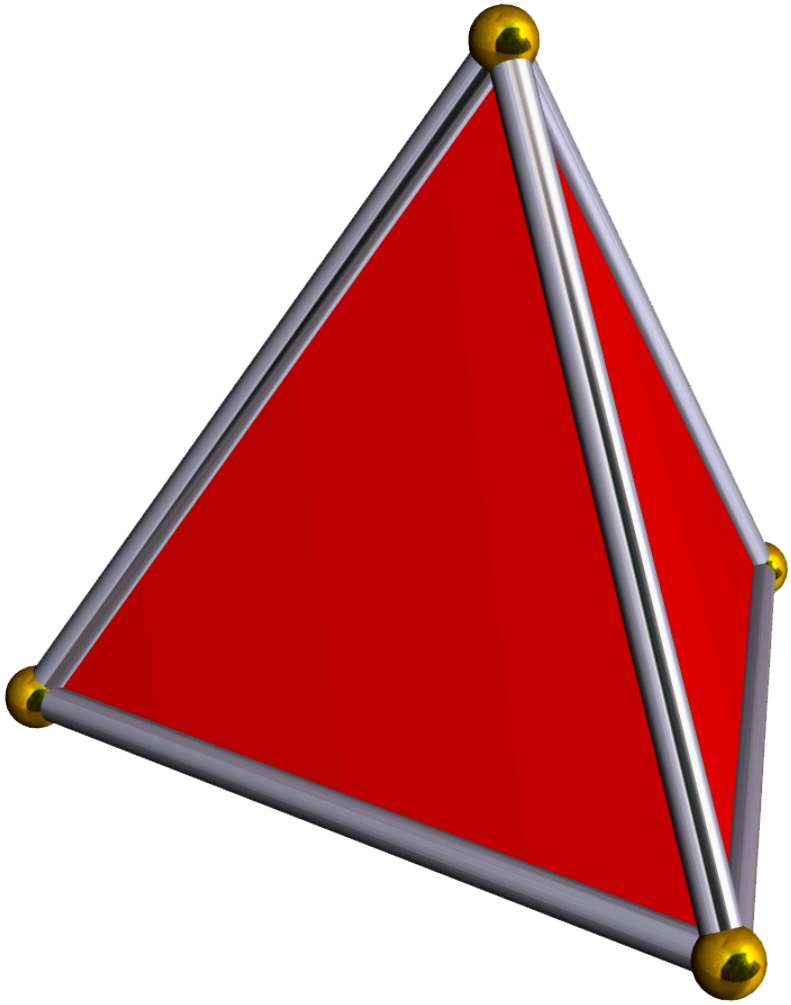
[28]:



1.5.8 Simplexe

[29]:
```
Image('https://upload.wikimedia.org/wikipedia/commons/2/25/Tetrahedron.png', width=400)
```

[29]:



1.5.9 Postier chinois

[30]:
```
Image("postier.png")
```

[30]:

1.5.10 Arbre de décision

[31]:
```
Image("dectree.png")
```

[31]:

1.6 Quizz 4 : génie logiciel

1.6.1 Les exceptions

Les programmes qui plantent mais en fait c'est pas grave.

[32]:
```
try:
    y = 1 / 0
except Exception as e:
    print(type(e), e)
```

```
<class 'ZeroDivisionError'> division by zero
```

Ou des fois-ci

[33]:
```
try:
    x = -1
    if x < 0:
        raise ValueError("La racine carrée d'un nombre positif est inconnue de ce␣
 ↪programme.")
except Exception as e:
    print(type(e), e)
```

```
<class 'ValueError'> La racine carrée d'un nombre positif est inconnue de ce
programme.
```

1.6.2 Les expressions régulières

[34]:
```
import re
reg = re.compile("[A-Z]{2,}")
texte = "Etrange ces acronymes comme ENSAE ou CPU qui ressortent comme par magie."
reg.findall(texte)
```

[34]: ['ENSAE', 'CPU']

1.6.3 Les tests unitaires

Ou comment protéger son code contre l'intrusion d'un codeur distrait.

[35]:
```
def return_sept(n):
    return int('7' * n)

def test_unitaire():
    # Si ça plante, c'est de votre faute.
    assert return_sept(4) == 7777

test_unitaire()
```

1.6.4 Le packaging...

Il y a plusieurs façons de passer à la postérité. C'est l'une d'elle.

1.7 Problèmes, Exercices

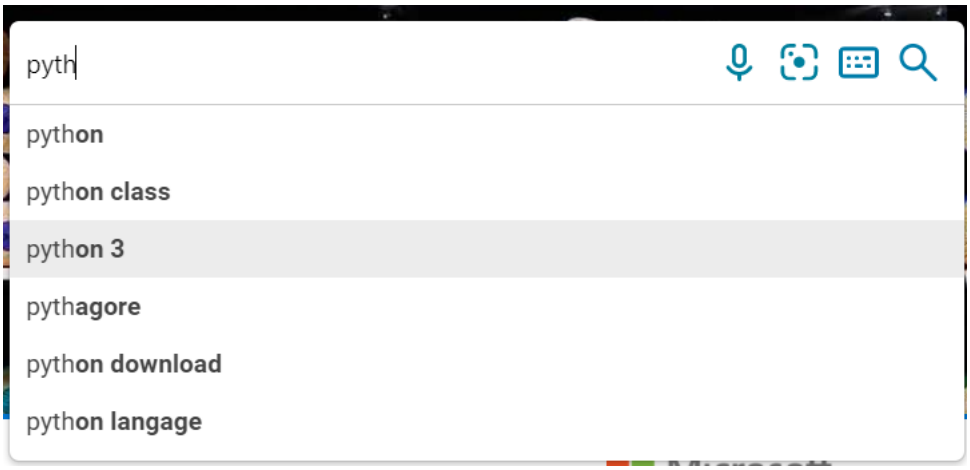
1.7.1 Enumérer les permutations

enumerate_permutations_recursive.

1.7.2 Suggestions

[36]:
```
Image("suggestion.png")
```

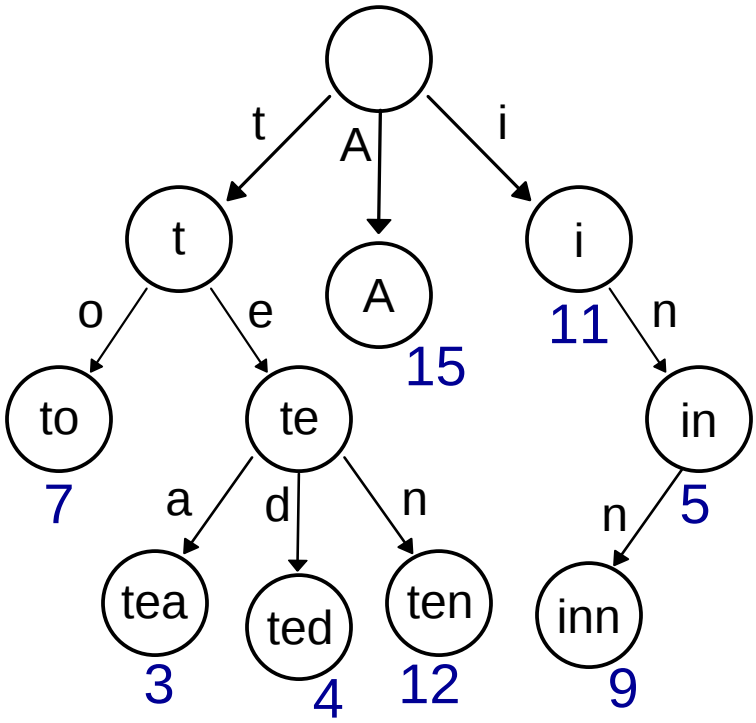
[36]:



Le **trie**

[37]:
```
SVG('https://upload.wikimedia.org/wikipedia/commons/b/be/Trie_example.svg')
```

[37]:



1.7.3 Récupérer des mails automatiquement

…

1.7.4 Calculer des statistiques et les envoyer automatiquement au format PDF par mail

…

1.7.5 Ecrire un système qui note automatiquement les présences

par reconnaissance faciale mais qui échoue pour cause de masques qui passe alors par la voix en demandant de chanter du Johnny.

1.8 A suivre.

[38]:

```
```