GASSENDI

jeudi 4 février 2021

TP monde connecté : cours du 04/02/2021 : exercice 7 en Python sur des LED

Élaboration	
4 février 2021	
Jean D	GASSENDI
Animateur	
Administration informatique	
Nom du fichier	00_TP_monde_connecte_cours_21_01_2021_Exercice_4_5_LED_V0.1.odt

Tutoriels avec exercices LED

Généralités sur l'architecture du TP

La progression dans l'apprentissage de Python s'effectuera par petits modules.
- 7 modules LED (led; led1; led2; led3; led4; led5; led6)
- 1 module spécifique multi_LED (**led7**)

- La fin de chaque module comportera un TP

1er module : **led.py** apprentissage des instructions:
(*import ; print ; led.on() et led.off() ; sleep()*)

2eme module : **led1.py** apprentissage des instructions:
(*led.blink() ; for in range(): variables*)

3eme module : **led2.py** apprentissage des instructions:
(*def() fonctions*)

4eme module : **led3.py** apprentissage des instructions:
(*input() ; if: elif: else:*)

5eme module : **led4.py** apprentissage des instructions:
(*while : break*)

6eme module : **led5.py** apprentissage des instructions:
(*PWMLED ; led.value() ; dictionnaire {}*)

7eme module : **led6.py** apprentissage des instructions:
(*RGBLED ; led.color()*)

Module spécifique : **led7.py** apprentissage des instructions:
(*LEDBoard ; led.on() , led.off() et led.toggle() ; LEDBarGraph() ; graph.value= ; graph.off() ; TrafficLights ; liste [ ]*)

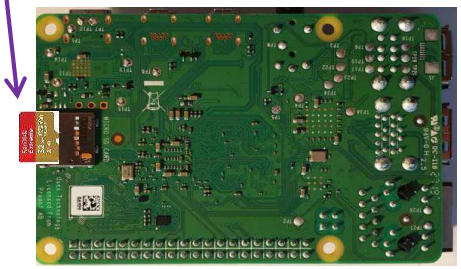
Tutoriels avec exercices LED

A. Matériels

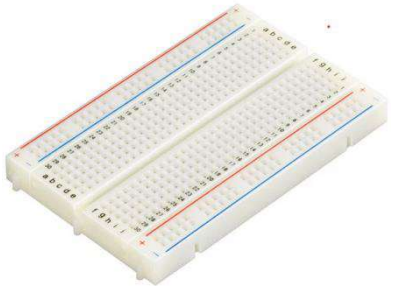
Raspberry Pi

GPIO Pi

Carte µSD avec l'OS Raspbian



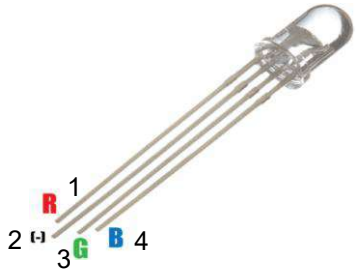
Breadboard



Résistance

LED Rouge, Verte, Jaune

LED RGB



Tutoriels avec exercices LED

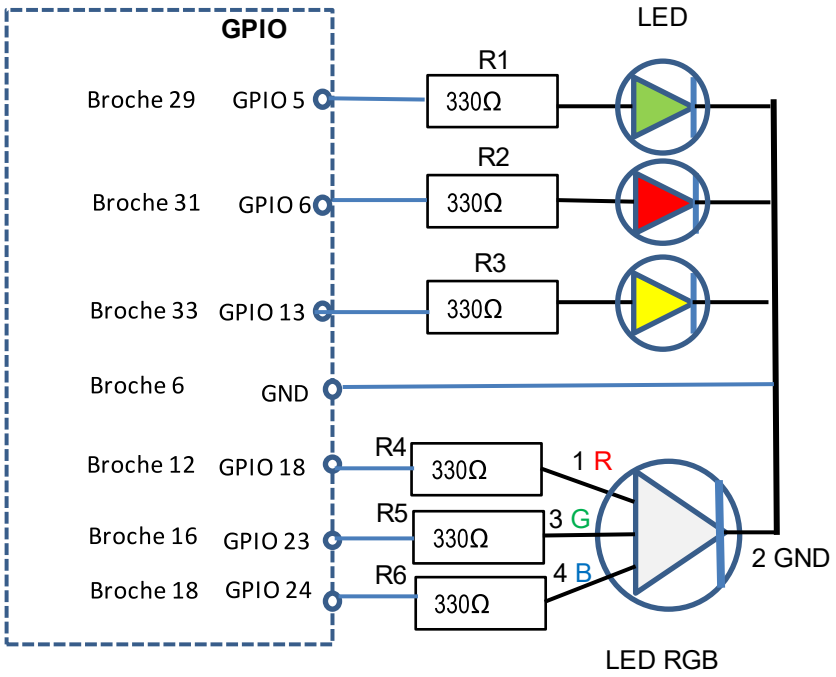
T Cobbler 40 pts

Câble en nappe

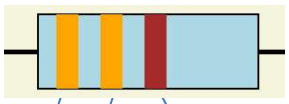
Fils de liaison

Tutoriels avec exercices LED

Représentation schématique



330Ω 33*10



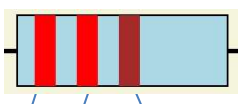
Orange **3**

Orange **3**

Marron 10^{1} = 10

Ou

220Ω 22*10



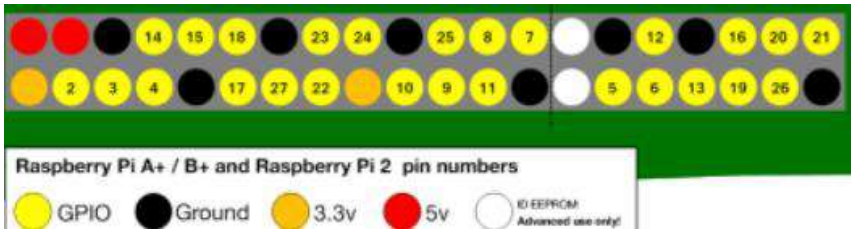
Roue **2**

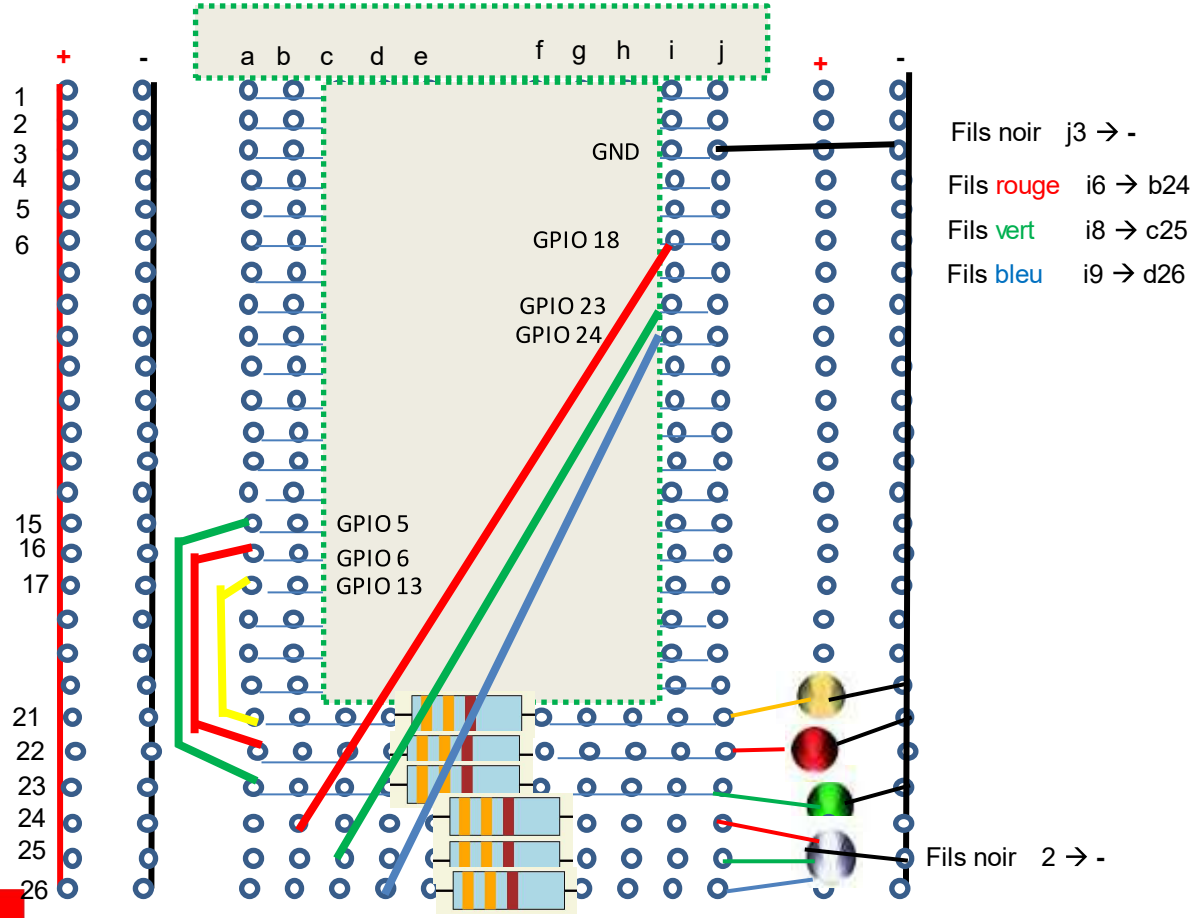
Rouge **2**

Marron 10^{1} = 10

Tutoriels avec exercices LED

Câblage





Fils noir j3 → -

Fils rouge i6 → b24

Fils vert i8 → c25

Fils bleu i9 → d26

Fils vert a15 → a23

Fils rouge a16 → a22

Fils jaune a17 → a21

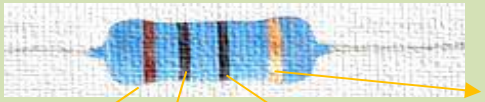
Fils noir 2 → -

Avant tout câblage, éteindre la Raspberry Pi

Tutoriels avec exercices LED

Code couleur résistance

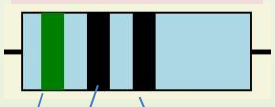
Couleur	Valeur 1ére Bande	Valeur 2éme Bande	Valeur 3éme Bande	Mnémotechnique
Noir	0	0		Ne
Marron	1	1		Manger
Rouge	2	2		Rien
Orange	3	3		Ou
Jaune	4	4		Jeûner
Vert	5	5		Voilà
Bleu	6	6		Bien
Violet	7	7		Votre
Gris	8	8		Grande
Blanc	9	9		Bêtise



Tutoriels avec exercices LED

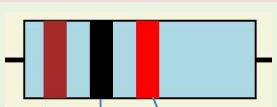
Exemples réalisés à partir de l'utilitaire: **code_couleur.py**

50Ω 50*1



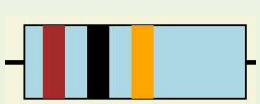
Vert **5**
Noir **0**
Noir $10^0 = 1$

1 000Ω ou **1k Ω** 10*100



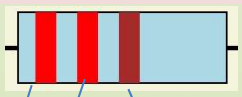
Marron **1**
Noir **0**
Rouge $10^2 = 100$

10000Ω ou **10k Ω** 10*1 000



Marron **1**
Noir **0**
Orange $10^3 = 1\,000$

220Ω 22*10



Rouge **2**
Rouge **2**
Marron $10^1 = 10$

2 200Ω ou **2.2k Ω** 10*100

Rouge **2**
Rouge **2**
Rouge $10^2 = 100$

100 000Ω ou **100k Ω** 10*10 000

Marron **1**
Noir **0**
Jaune $10^4 = 10\,000$

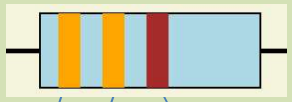
470Ω 47*10

Jaune **4**
Violet **7**
Marron $10^1 = 10$

4 700Ω ou 4.7k Ω 47*100

Jaune **4**
Violet **7**
Rouge $10^2 = 100$

330Ω 33*10



Orange **3**
Orange **3**
Marron $10^1 = 10$

Tuto exercice: led7.py

Python Les nouvelles instructions utilisées.

from gpiozero import LEDBoard

De la bibliothèque **gpiozero** j'importe toutes les fonctions appartenant à l'Objet **LEDBoard**
syntaxe:
LEDBoard(les n°GPIO) ; led.on() ; led.off() ; led.toggle()

leds = **LEDBoard(**led1, led2, led3, …, ledn**)** Permet de commander plusieurs leds.

N° de led	0	1	2	n

led.**on(**n°led**)** Allume la led indiquée

led.**off(**n°led**)** Eteint la led indiquée

led.**toggle(**n°led**)** Inverse l'état de la led indiquée

Exemple:

leds = **LEDBoard(**5, 6, 13**)** Permet commander les leds GPIO5, GPIO6, GPIO13.

Tuto exercice: led7.py

Python

Les nouvelles instructions utilisées.

Détail du paramétrage de l'objet LEDBoard

class gpiozero.LEDBoard (*pins, pwm=False, active_high=True, initial_value=False, pin_factory=None, **named_pins)

***pins**	Les broches GPIO auxquelles sont connectées le leds.
pwm(*bool*)	Si True chaque led peut être contrôlée par PWMLED. Si False (par défaut) fonctionnement nominal
active_high (*bool*)	Si True (par défaut). Le **.on()** met toutes les leds à **HIGH** Si False le **.on()** met toutes les leds à **LOW** le **.off()** met toutes les leds à leur l'opposé
initial_value (*bool* or *None*)	Si False (par défaut), toutes les leds sont éteintes au démarrage. Si None chaque led garde l'état d'avant Si True toutes les leds seront mises à 1 à l'initialisation.
****named_pins**	Permet de donner un nom à chaque led

Méthodes disponibles avec l'Objet LEDBoard

led.**on(**n°led**)**

led.**off(**n°led**)**

led.**toggle(**n°led**)**

Tuto exercice: led7.py

Python

Les nouvelles instructions utilisées.

Si pas de n° alors l'action est faite sur toutes les leds

Exemples:

Allume toutes les leds et les éteint une à une

```
from gpiozero import LEDBoard
trois_leds = LEDBoard(5, 6, 13)

trois_leds.on()        résultat:  allume les 3 leds
trois_leds.off(0)      résultat:  éteint 1ére led  n°5
trois_leds.off(-1)     résultat:  éteint dernière led  n°13
trois_leds.off(1)      résultat:  éteint 2éme led  n°6
```

Allume toutes les leds une à une

```
from gpiozero import LEDBoard
trois_leds = LEDBoard(5, 6, 13)

trois_leds.off()       résultat:  éteint les 3 leds
trois_leds.on(0)       résultat:  allume 1ére led  n°5
trois_leds.off(-1)     résultat:  allume dernière led  n°13
trois_leds.off(1)      résultat:  allume 2éme led  n°6
trois_leds.toggle()    résultat:  éteint les 3 leds
```

Tuto exercice: led7.py

Python

Les nouvelles instructions utilisées.

from gpiozero import **LEDBarGraph**

De la bibliothèque **gpiozero** j'importe toutes les fonctions appartenant à l'Objet **LEDBarGraph**
syntaxe:
LEDBarGraph(les n°GPIO) ; graph.value ; graph.off()

graph = **LEDBarGraph(**led1, led2, led3, …, ledn**)**

Permet de commander un groupe de leds.

graph.**value =** nombre de leds à gérer / la totalité des leds

Exemples:

graph = **LEDBarGraph(**5, 6, 13**)**

graph.**value =** 1 / 3 — Allume seulement la première led

graph.**value =** 2 / 3 — Allume les 2 premières leds

graph.**value =** -2 / 3 — Allume les 2 dernières leds

Tuto exercice: led7.py

Python

Les nouvelles instructions utilisées.

Détail du paramétrage de l'objet LEDBarGraph

class gpiozero.LEDBarGraph (*pins, pwm=False, active_high=True, initial_value=0, pin_factory=None)

***pins**	Les broches GPIO auxquelles sont connectées le leds.
pwm(*bool*)	Si True chaque led peut être contrôlée par PWMLED. Si False (par défaut) fonctionnement nominal
active_high (*bool*)	Si True (par défaut). Le **.on()** met toutes les leds à **HIGH** Si False le **.on()** met toutes les leds à **LOW** le **.off()** met toutes les leds à leur l'opposé
initial_value (*bool* or *None*)	

Méthodes disponibles avec l'Objet LEDBarGraph

graph.**value =** n / p

n = nombre de leds

p = nombre total de leds

Tuto exercice: led7.py

Python

Les nouvelles instructions utilisées.

```
from gpiozero import TrafficLights
```

De la bibliothèque **gpiozero** j'importe toutes les fonctions appartenant à l'Objet **TrafficLights**
syntaxe:
TrafficLights(les n°GPIO) ; traffic.red.on() ; traffic.yellow.on() ; traffic.green.on()

```
traffic = TrafficLights(led1, led2, led3)
```

Permet de gérer les feux tricolore.

```
traffic.red.on()
traffic.yellow.on()
traffic.green.on()
```

Allume led rouge
Allume led jaune
Allume led verte

Exemples:

```
traffic = TrafficLights(5, 6, 13)
traffic.red.on()
traffic.green.on()
```

Tuto exercice: led7.py

Python

Les nouvelles instructions utilisées.

Détail du paramétrage de l'objet TrafficLights

class gpiozero.TrafficLights red, amber, green, *, yellow=None, pwm=False, initial_value=False, pin_factory=None)

red (*int ou str*)	La broche GPIO connectée à la led rouge.
amber (*int ou str*)	La broche GPIO connectée à la led jaune.
green (*int ou str*)	La broche GPIO connectée à la led verte.
yellow (*int ou str*)	pas utilisé.
pwm(*bool*)	Si True chaque led peut être contrôlée par PWMLED. Si False (par défaut) fonctionnement nominal
initial_value (*bool* or *None*)	Si False (par défaut), toutes les leds sont éteintes au démarrage. Si None chaque led garde l'état d'avant Si True toutes les leds seront mises à 1 à l'initialisation.

Méthodes disponibles avec l'Objet TrafficLights

traffic.**red.on()**

traffic.**yellow.on()**

traffic.**green.on()**

Tuto exercice: led7.py

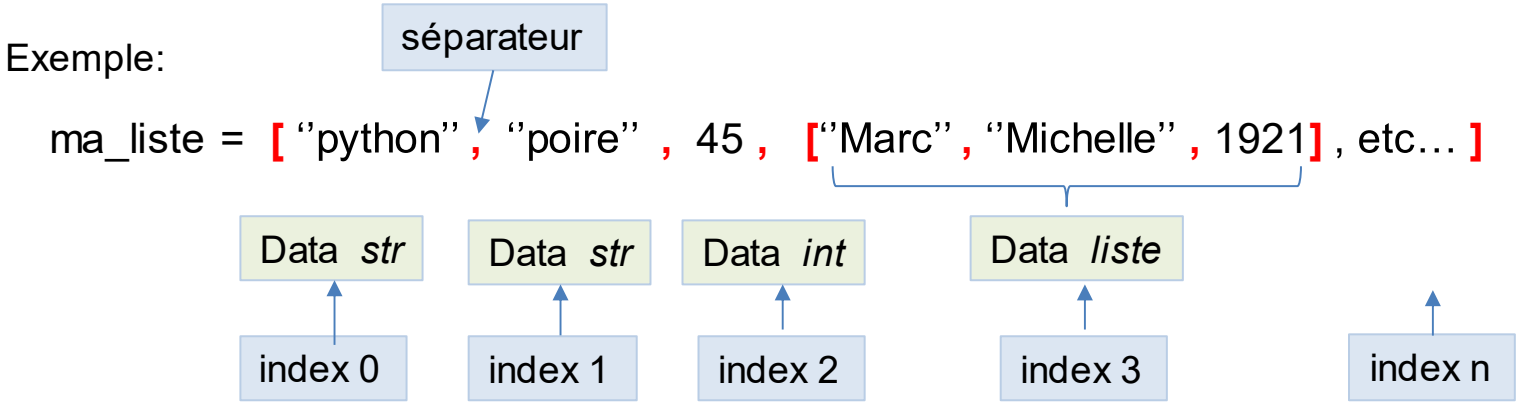
Python

Variable de type ‘’Liste”

Une liste est composée d’un ensemble de données hétérogènes ordonnées. (Liste à la Prévert)
L’accès aux éléments se fait à travers un **index** numérique.

Syntaxe d’une liste: une suite de **valeurs** séparées par **,** et encadrées par des crochets **[ ]**

Exemple:



Tuto exercice: led7.py

Python

Variable de type "Liste"

Initialisation d'une liste

liste_fruits = [ ]

Construction d'une liste

liste_fruits = ['pomme' , 'poire' , 'prune' , 'tomate' , 'cerise', 55]

Affichage du contenu d'une liste

print(liste_fruits)

résultat
['pomme', 'poire', 'prune', 'tomate', 'cerise', 55]

print(liste_fruits**[0:]**)

résultat
['pomme', 'poire', 'prune', 'tomate', 'cerise', 55]

L'affichage est la copie de la liste.

Tuto exercice: led7.py

Exemples:

liste_fruits = [‘pomme’ , ‘poire’ , ‘prune’ , ‘tomate’ , ‘cerise’, 55]

0 1 2 3 4 5

Affichage des éléments à partir du 4éme

```
print(liste_fruits[3:])
```

résultat
[‘tomate’, ‘cerise’, 55]

Affichage de tous les éléments sauf les 4 derniers

```
print(liste_fruits[:3])
```

résultat
[‘pomme’, ‘poire’, ‘prune’]

index commence à 0

Affichage du dernier élément

```
print(liste_fruits[-1])
```

résultat
[55]

Affichage du 3éme élément de la liste

```
print(liste_fruits[2])
```

résultat
prune

Tuto exercice: led7.py

Synthèse des opérations sur les listes

liste_fruits.**append(**valeur**)** Ajout d'un élément dans la liste. Toujours en fin de liste

Exemple: ajout de **fraise**
liste_fruits.append('fraise')

Résultat
liste_fruits = **[**'pomme'**,** 'poire'**,** 'prune'**,** 'tomate'**,** 'cerise'**,** 55**,** 'fraise'**]**

del liste_fruits**[index]** Suppression d'un élément dans la liste à l'aide de **l'index**

Exemple: suppression de **55**
del liste_fruits[5]

Résultat
liste_fruits = **[**'pomme'**,** 'poire'**,** 'prune'**,** 'tomate'**,** 'cerise'**,** 'fraise'**]**

liste_fruits.**remove(**valeur**)** Suppression d'un élément dans la liste

Exemple: suppression de **poire**
liste_fruits.remove('poire')

Résultat
liste_fruits = **[**'pomme'**,** 'prune'**,** 'tomate'**,** 'cerise'**,** 'fraise'**]**

Tuto exercice: led7.py

Synthèse des opérations sur les listes

liste_fruits.**index(**valeur**)** Afficher l'index d'un élément

Exemple: afficher l'index de **tomate**
print(liste_fruits.index('tomate'))

Résultat
2

len(nom de la liste**)** Affiche le nombre d'élément de la liste

Exemple: nombre d'élément de la liste fruits
print(len(liste_fruits))

Résultat
5

liste_fruits.**sort()** Trier une liste soit par ordre alphabétique ou numérique

Exemple: trier la liste_fruits (ordre alphabétique)
liste_fruits.sort()

Résultat
['cerise', 'fraise' , 'pomme' , 'prune' , 'tomate']

Tuto exercice: led7.py

Synthèse des opérations sur les listes

liste_msg = **[1,0,2,5,'A','F','F','F',0,0,3,5,6,8,7,9,4,2,2,'B','C','E','F','F',4,5,6,8,2,9,1]**

liste_fruits.**count(**valeur**)** Afficher le nombre de '*valeur*' dans la liste

Exemple: compter le nombre de F dans la liste
print(liste_msg.count('F'))

Résultat
5

for element **in** liste_fruits **:** Afficher les éléments d'une liste à travers une boucle **for in :**

Exemple: afficher tous les éléments de la liste
```
for element in liste_fruits:
    print(element)
```

Résultat
cerise
fraise
pomme
prune
tomate

Tuto exercice: led7.py

Synthèse des opérations sur les listes

```
for element in enumerate(liste_fruits):
```

Afficher les éléments (valeur et index) d'une liste.

Exemple: afficher tous les éléments avec leurs valeurs et index de la liste

```
for element in enumerate(liste_fruits):
    print(element)
```

Résultat

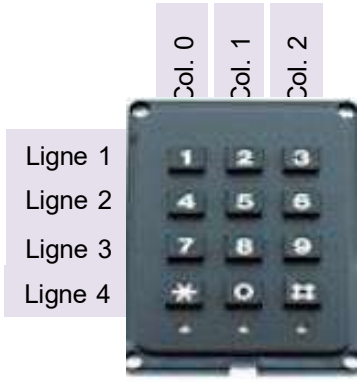
```
(0, 'cerise')
(1, 'fraise')
(2, 'pomme')
(3, 'prune')
(4, 'tomate')
```

Tuto exercice: led7.py

Synthèse des opérations sur les listes

Créer un tableau 2D (ligne, colonne)

Clavier




```
clavier = [ [1, 2, 3], [4, 5, 6], [7, 8, 9], ['*', 0, '#'] ]
```

L: 1 C: 1,2,3 | L: 2 C: 1,2,3 | L: 3 C: 1,2,3 | L: 4 C: 1,2,3

```
clavier = [ [1, 2, 3],
            [4, 5, 6],
            [7, 8, 9],
            ['*', 0, '#'] ]

print( clavier[2] )       résultat: [7, 8, 9]

print( clavier[0][0] )    résultat: 1
print( clavier[1][0] )    résultat: 4
print( clavier[3][1] )    résultat: 0
```

```
clavier = [ [1, 2, 3], [4, 5, 6], [7, 8, 9], ['*', 0, '#'] ]
ligne = [0, 1, 2, 3]
colonne = [0, 1, 2)

for i in range(len(ligne):
    for j in range(len(colonne):
        print("la ligne", i+1, "de la colonne", j+1,"est", clavier[i][j])
```

Résultat

```
La ligne 1 de la colonne 1 est 1
La ligne 1 de la colonne 2 est 2
La ligne 1 de la colonne 3 est 3
La ligne 2 de la colonne 1 est 4
La ligne 2 de la colonne 2 est 5
La ligne 2 de la colonne 3 est 6
La ligne 3 de la colonne 1 est 7
La ligne 3 de la colonne 2 est 8
La ligne 3 de la colonne 3 est 9
La ligne 4 de la colonne 1 est *
La ligne 4 de la colonne 2 est 0
La ligne 4 de la colonne 3 est #
```

TP LED.py

Travaux Pratique

Modifier le programme pour :

1. allumer les leds successivement et les faire clignoter
2. créer des listes de valeurs de température, pression et humidité
3. afficher tous les éléments des listes (index et valeurs)
4. trier les listes par ordre croissant
5. Que faudrait-il faire pour faire un histogramme de chaque liste?

Glossaire

Sigle	
LED	**L**ight **E**mitting **D**iode (Diode Electroluminescente)
PWM	**P**ulse-**W**idth **M**odulation