GASSENDI

vendredi 22 janvier 2021

TP monde connecté : cours du 21/01/2021 : exercices 4, 5 en Python sur des LED

Élaboration	
22 janvier 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)

- 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:*)

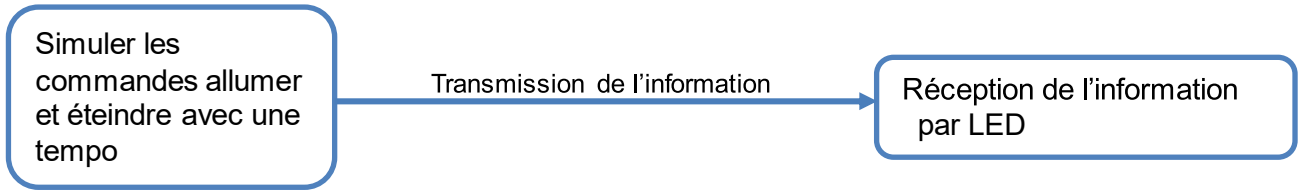
5eme module : **led4.py** apprentissage des instructions:
(*while : break*)

Tutoriels avec exercices LED

. Description textuelle du système

La Raspberry va commander l'allumage et l'extinction successive des 3 LED de couleur pendant n secondes suivant différentes façons et en utilisant de manière progressive des instructions Python.

. Description par un schéma



Tutoriels avec exercices LED

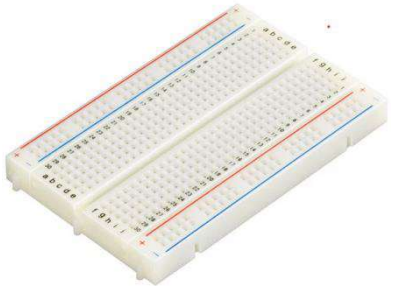
A. Matériels

Raspberry Pi

GPIO Pi

Carte µSD avec l'OS Raspbian

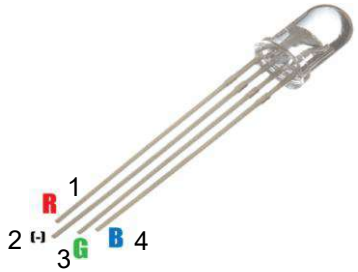
Breadboard



Résistance

LED Rouge, Verte, Jaune

LED RGB



Tutoriels avec exercices LED

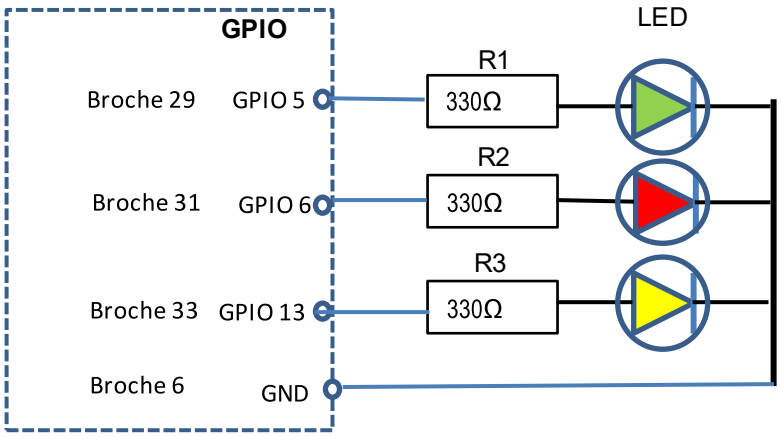
T Cobbler 40 pts

Câble en nappe

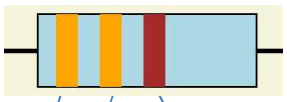
Fils de liaison

Tutoriels avec exercices LED

Représentation schématique



330Ω 33*10



Orange **3**

Orange **3**

Marron 10^{1} = 10

Ou

220Ω 22*10

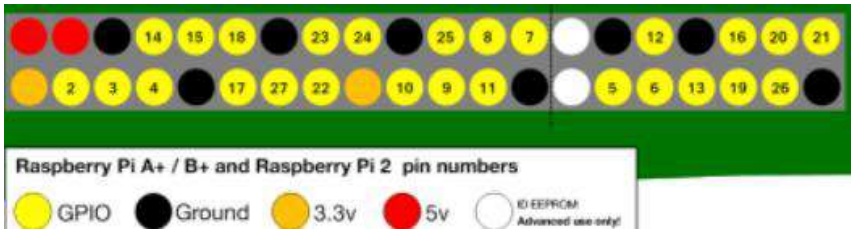
Roue **2**

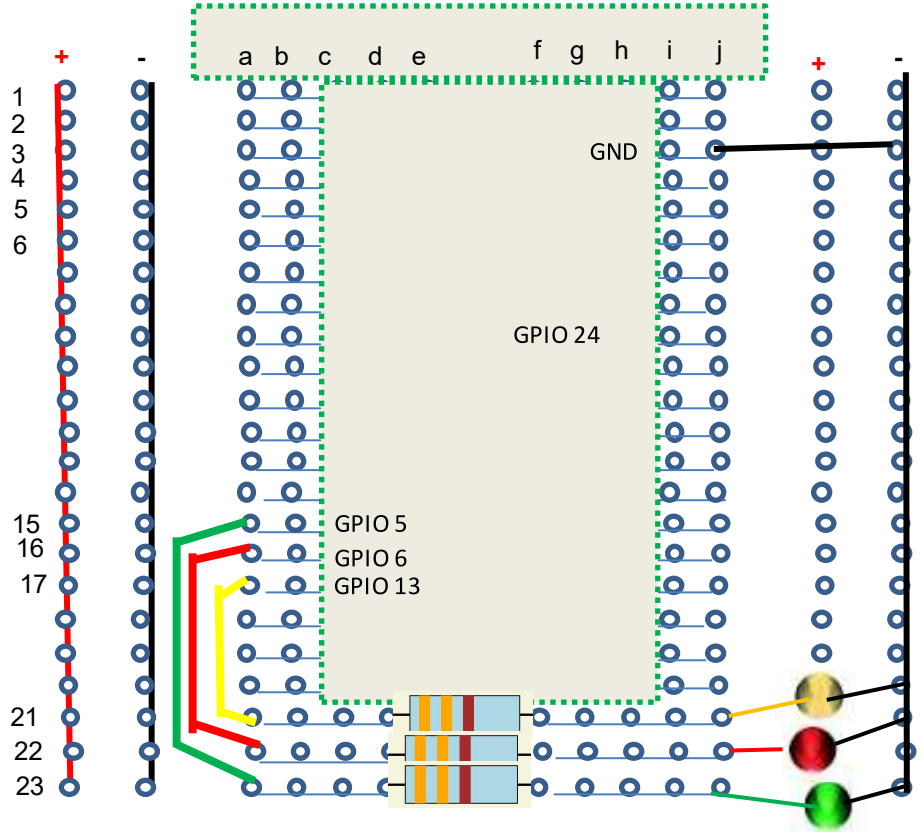
Rouge **2**

Marron 10^{1} = 10

Tutoriels avec exercices LED

Câblage





Fils noir j3 → -

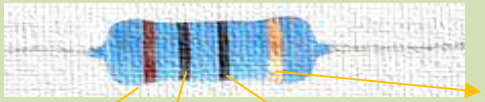
Fils vert a15 → a23
Fils rouge a16 → a22
Fils jaune a17 → a21

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	$10^0 = 1$	Ne
Marron	1	1	$10^1 = 10$	Manger
Rouge	2	2	$10^2 = 100$	Rien
Orange	3	3	$10^3 = 1\,000$	Ou
Jaune	4	4	$10^4 = 10\,000$	Jeûner
Vert	5	5	$10^5 = 100\,000$	Voilà
Bleu	6	6	$10^6 = 1\,000\,000$	Bien
Violet	7	7		Votre
Gris	8	8		Grande
Blanc	9	9		Bêtise



4éme Bande dorée
Précision

1ére Bande

2éme Bande

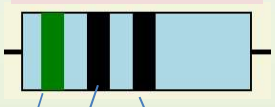
3éme Bande
Coeff de multiplication

Cf. exemples

Tutoriels avec exercices LED

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

50Ω 50*1



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

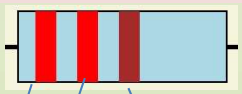
1 000Ω ou **1k Ω** 10*100

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 $10^4 = 10\,000$

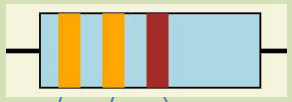
470Ω 47*10

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



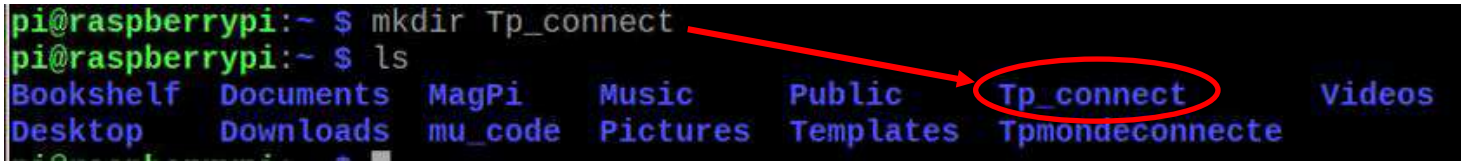
Orange **3**
Orange **3**
Marron $10^1 = 10$

Tutoriels avec exercices LED

Avant de démarrer les travaux pratiques, il est nécessaire de créer un répertoire de travail.

Mode Console

La commande Linux pour créer un répertoire est : **mkdir** (make directory)



Tuto exercice: led4.py

Python

Les nouvelles instructions utilisées.

La boucle *Tant que*

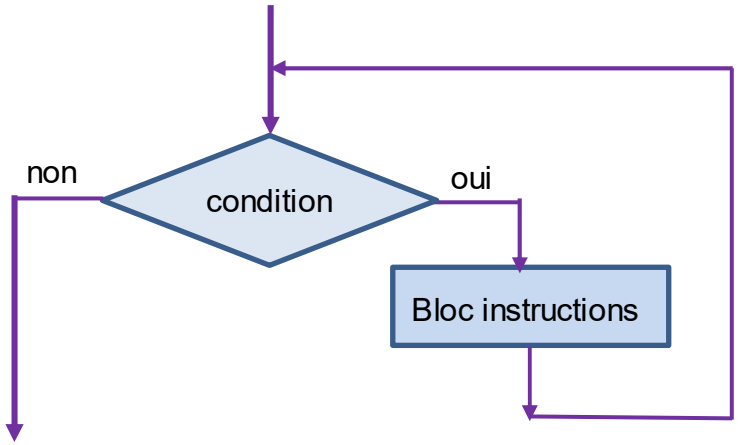
Tant que la condition est vraie **alors** le bloc des instructions sous le **while** est **exécuté**.

Syntaxe

while condition**:**

....bloc instruction

....incrémentation de la boucle



Tuto exercice: led4.py

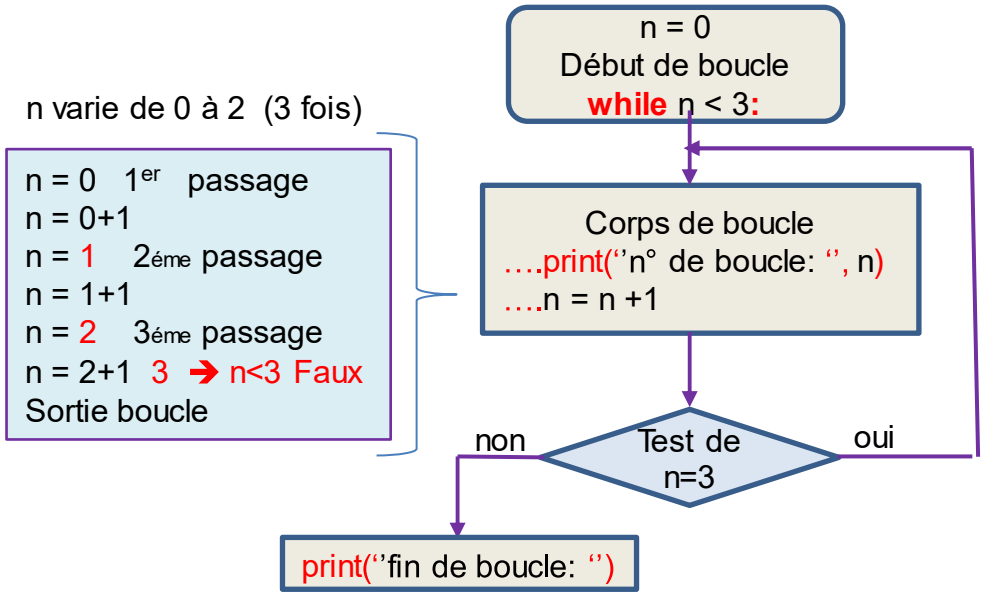
Python

Les nouvelles instructions utilisées.

Exemples:

Paramètre de boucle défini

```
n = 0
while n < 5 :
....bloc instruction
.... n = n + 1
print("fin de boucle")
```



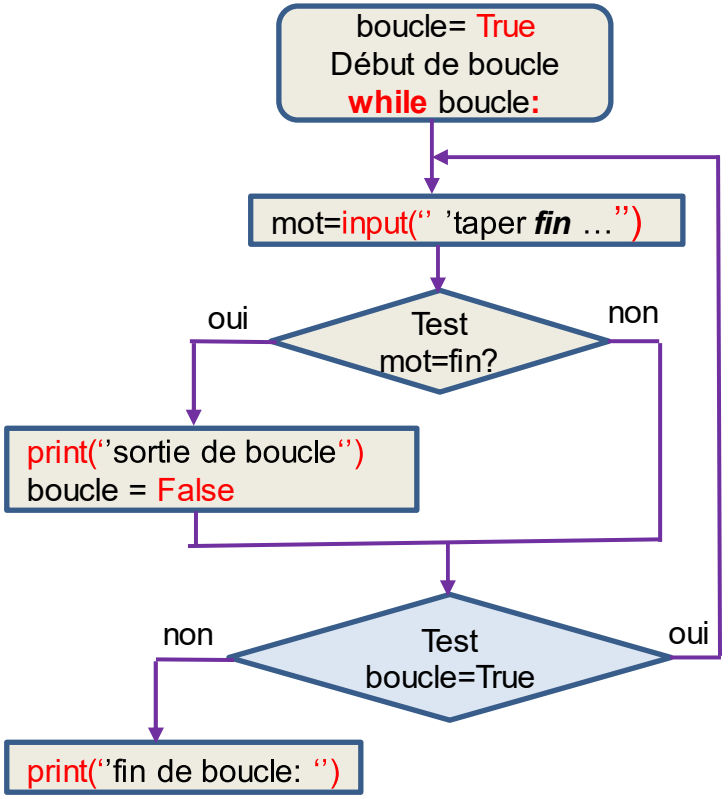
Tuto exercice: led4.py

Python

Les nouvelles instructions utilisées.

boucle en attente d'un mot particulier pour sortir de la boucle

```
boucle = True
while boucle:
....mot=input(''taper fin pour quitter la boucle'')
....if mot == 'fin":
........print(''sortie de boucle'')
........boucle=False
print(''fin de boucle")
```



Tuto exercice: led4.py

Python Les nouvelles instructions utilisées.

Syntaxe

break Permet de sortir d'un boucle **for** ou **while**.

Exemple:

```
while True :
....m=int(input("entrer un entier >0 :"))
....print("vous avez entrez :", m)
.........if m > 0:
.............break
print("fin de boucle")
```

Tuto exercice: led4.py

Python

Les nouvelles instructions utilisées.

Comment choisir entre boucle **for** et boucle **while** ?

Si le nombre d'itérations à exécuter est connu avant de démarrer alors choisir la boucle **for**

Si la décision d'arrêter la boucle ne peut se faire que par un test alors choisir la boucle **while**

Transformation d'une boucle **for** en une boucle **while**

```
for n in (4):
....print("n a pour valeur", n)
```

```
n = 0
while n < 4 :
.... print("n a pour valeur", n)
....n = n + 1
```

Tuto exercice: led4.py

Travaux Pratique

Modifier le programme pour :

1. dans la fonction« **nbre_car** » remplacer la boucle for par une boucle while
2. Que faudrait-il faire dans la boucle du programme principal pour quitter la boucle avec l'instruction break ?

Glossaire

Sigle	
LED	**L**ight **E**mitting **D**iode (Diode Electroluminescente)

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)

- 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*)

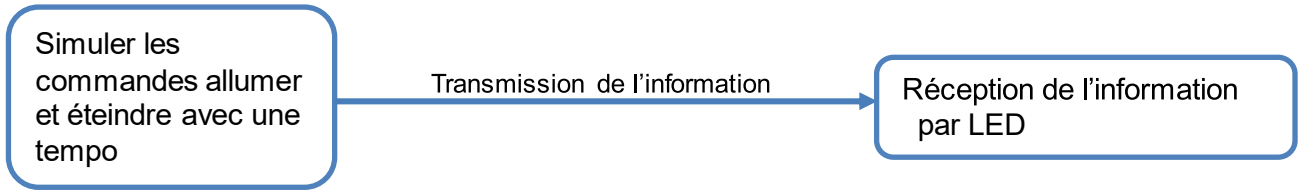
6eme module : **led5.py** apprentissage des instructions:
(*PWMLED; led.value() ; dictionnaire { }*)

Tutoriels avec exercices LED

. Description textuelle du système

La Raspberry va commander l'allumage et l'extinction successive des 3 LED de couleur pendant n secondes suivant différentes façons et en utilisant de manière progressive des instructions Python.

. Description par un schéma



Tutoriels avec exercices LED

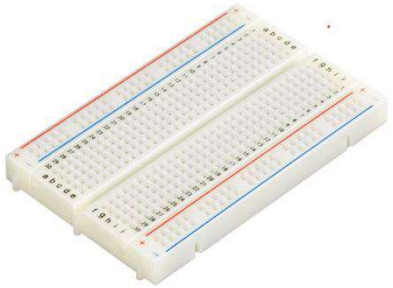
A. Matériels

Raspberry Pi

GPIO Pi

Carte µSD avec l'OS Raspbian

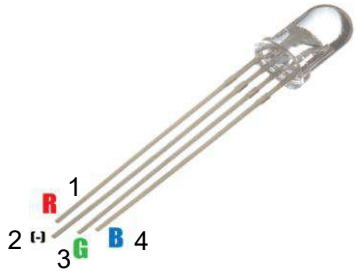
Breadboard



Résistance

LED Rouge, Verte, Jaune

LED RGB



Tutoriels avec exercices LED

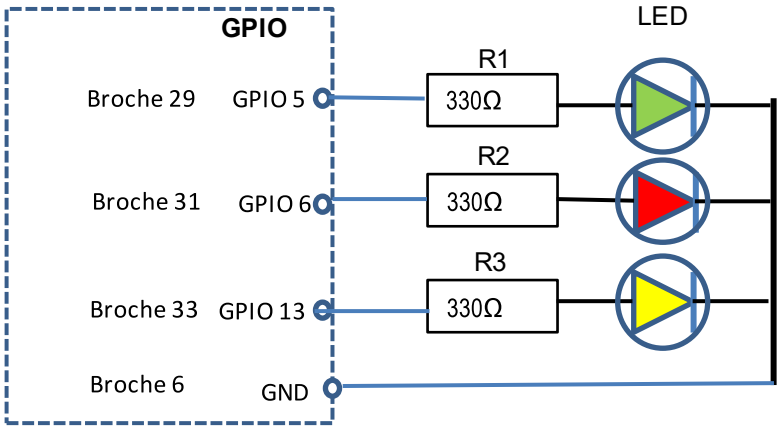
T Cobbler 40 pts

Câble en nappe

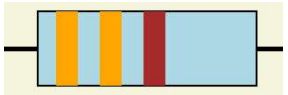
Fils de liaison

Tutoriels avec exercices LED

Représentation schématique



330Ω 33*10



Orange **3**
Orange **3**
Marron 10^{1} = 10

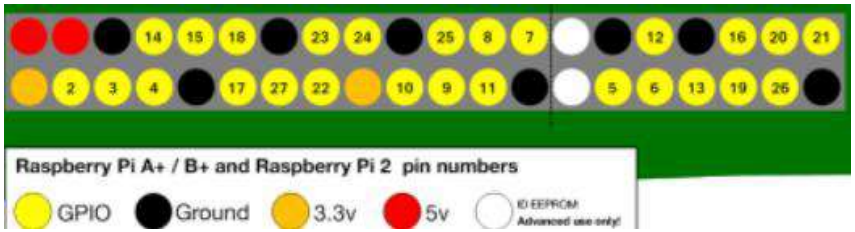
Ou

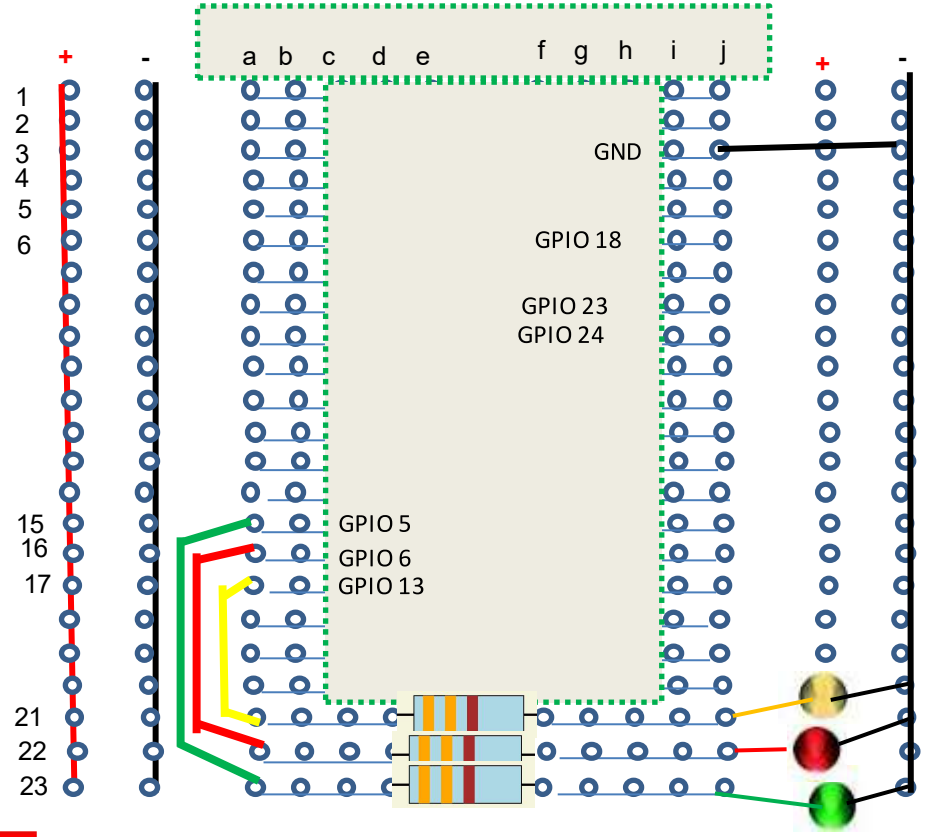
220Ω 22*10

Roue **2**
Rouge **2**
Marron 10^{1} = 10

Tutoriels avec exercices LED

Câblage





Fils noir j3 → -

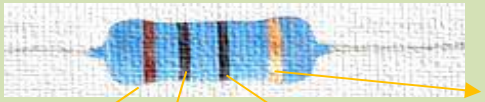
Fils vert a15 → a23
Fils rouge a16 → a22
Fils jaune a17 → a21

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	$10^0 = 1$	Ne
Marron	1	1	$10^1 = 10$	Manger
Rouge	2	2	$10^2 = 100$	Rien
Orange	3	3	$10^3 = 1\,000$	Ou
Jaune	4	4	$10^4 = 10\,000$	Jeûner
Vert	5	5	$10^5 = 100\,000$	Voilà
Bleu	6	6	$10^6 = 1\,000\,000$	Bien
Violet	7	7		Votre
Gris	8	8		Grande
Blanc	9	9		Bêtise



4éme Bande dorée
Précision

1ére Bande

2éme Bande

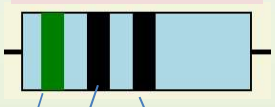
3éme Bande
Coeff de multiplication

Cf. exemples

Tutoriels avec exercices LED

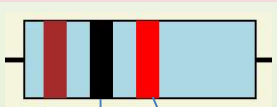
Exemples réalisés à partir de l'utilitaire: **code_couleur.py**

50Ω 50*1



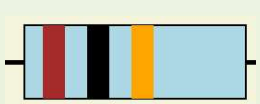
Vert **5**
Noir **0**
Noir $10^0 = 1$

1 000Ω ou **1k Ω** 10*100



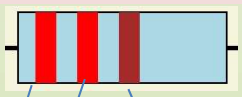
Marron **1**
Noir **0**
Rouge $10^2 = 100$

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 $10^4 = 10\,000$

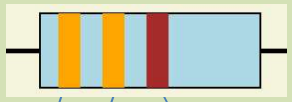
470Ω 47*10

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



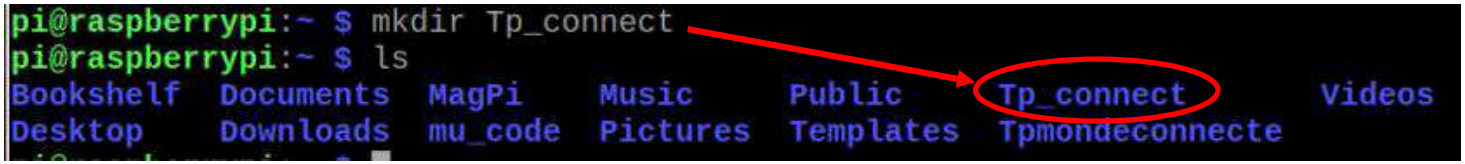
Orange **3**
Orange **3**
Marron $10^1 = 10$

Tutoriels avec exercices LED

Avant de démarrer les travaux pratiques, il est nécessaire de créer un répertoire de travail.

Mode Console

La commande Linux pour créer un répertoire est : **mkdir** (make directory)



Tuto exercice: led5.py

Python

Les nouvelles instructions utilisées.

from gpiozero import **PWMLED**

De la bibliothèque **gpiozero** j'importe toutes les fonctions appartenant à l'Objet **PWMLED**

syntaxe: PWMLED(n°GPIO) ; led.value=

Variable

led = **PWMLED(**n° GPIO**)** Affectation de la led concernée

led**.value** = rapport cyclique Fait varier la luminosité d'une led en paramétrant son rapport cyclique. (durée éclairage / durée d'extinction)

Rapport cyclique varie de 0.0 à 1.0

Exemples:

Luminosité led_jaune 50%	Luminosité led_verte 30%	Luminosité led_rouge 100%
led_jaune = **PWMLED(**5**)**	led_verte = **PWMLED(**6**)**	led_rouge = **PWMLED(**13**)**
led_jaune**.value** = 0.5	led_verte**.value** = 0.3	led_rouge**.value** = 1.0

Tuto exercice: led5.py

Python

Les nouvelles instructions utilisées.

Détail du paramétrage de l'objet PWMLED

class gpiozero.PWMLED (pin, *, active_high=True, initial_value=0, frequency=100, pin_factory=None)

pin (*int or str*) — la broche GPIO à laquelle est connecté la led.
Si non renseignée **alors** une information d'erreur est affichée (GPIODeviceError)

active_high (*bool*) — Si True (par défaut), la led fonctionne nominalement. Le **.on()** met le GPIO à **HIGH**
Si False le **.on()** met le GPIO à **LOW**
le **.off()** fait l'opposé

initial_value (*float*) — Si 0 (par défaut), la led est éteinte.
Les autres valeurs entre 0 et 1 peuvent servir d'initialisation pour le niveau de luminosité

frequency (*int*) — 100Hz (par défaut)

Méthodes disponibles avec l'Objet PWMLED

led.value = Permet de paramétrer le rapport cyclique (0.0 à 1.0).

led.pulse() Allume et éteint la led progressivement

Tuto exercice: led5.py

Python

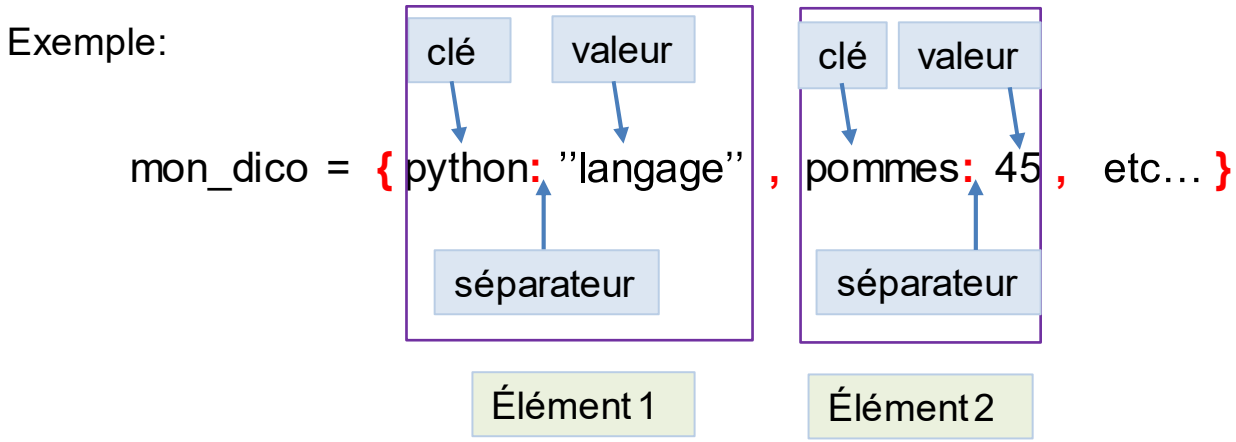
Les nouvelles instructions utilisées.

Variable de type ‘’Dictionnaire”

Un dictionnaire est composé d’un ensemble de données hétérogènes non séquentiel car les éléments ne sont pas disposés dans un ordre immuable. L’accès aux éléments se fait à travers une **clé** qui peut être alphabétique ou numérique.

Syntaxe d’un dictionnaire: une **clé** et une **valeur** séparée par **:** et encadrées par des accolades **{}**

Exemple:



dico_fct={1:led_v, 2:led_r, 3:led_j, 4:feux_tricolore, 5:fin}

Tuto exercice: led5.py

Python

Les nouvelles instructions utilisées.

Variable de type ''Dictionnaire"

Initialisation d'un dictionnaire

```
dico_fruits = { }
```

Construction d'un dictionnaire

```
dico_fruits = {'pomme':23, 'poire':45, 'prune':66, 'tomate':14}
```

Les clés sont alphabétiques

Affichage du contenu d'un dictionnaire

```
print(dico_fruits)
```

résultat

```
{ 'poire':45, 'tomate':14, 'prune':66, 'pomme':23}
```

Affichage dans un ordre différent

valeur

Affichage des **valeurs** d'un dictionnaire

```
print(dico_fruits.values())
```

résultat

```
dict.values( [ 45, 66,14, 23] )
```

clé

Affichage des **clés** d'un dictionnaire

```
print(dico_fruits.keys())
```

résultat

```
dict.keys( ['poire', 'prune', 'pomme', 'tomate'] )
```

Tuto exercice: led5.py

Python

Les nouvelles instructions utilisées.

dico.**items()** Permet de parcourir un dictionnaire à l'aide d'une boucle **for**

Exemple:

```
dico_fruits = {'pomme':23, 'poire':45, 'prune':66, 'tomate':14, 'fraise':3}
for clef, valeur in dico_fruits.items():
....print(clef, valeur)
```

résultat

```
pomme 23
poire 45
tomate 14
fraise 3
prune 66
```

Tuto exercice: led5.py

Python Les nouvelles instructions utilisées.

dict.get(key, default=None) Permet de saisir la clef dans un dictionnaire.

Exemple: construction d'un histogramme
chaque numéro est pris comme clé

```
msg = ''1025AFFF00356879422BCEFF4568291"
dico_histo = { }

for numero in msg:
....dico_histo[numero] = dico_histo.get(numero, 0) + 1

print(dico_histo)
```

clé

Valeur nulle si elle n'existe pas encore

Incrémentation de la valeur

Résultat

{'1' : 2, '0' : 3, '2' : 4, '5' : 3, 'F' : 5, 'A' : 1, '6' : 2, 'B' : 1, '8' : 2, 'C' : 1, '7' : 1, '4' : 2, '9' : 2, 'E' : 1, '3' : 1}

Tuto exercice: led5.py

Python

Les nouvelles instructions utilisées.

Exemple: gestion d'un menu

```
choix = input("Choisisser la fonction que vous voulez: \n "
              "l3n: LED verte à 3 niveaux d'intensité \n "
              "lbp: LED verte éclaire par pas de 0.1\n "
              "lbm: LED verte s'éteint par pas de 0.1\n "
              ‘’q: pour arrêter la boucle. ")
```

Gestion avec des **if, else**

```
if choix == "l3n":
….led_3n()
elif choix == "lbp":
….led_plus()
elif choix == "lbm":
….led_moins(0.1)
elif choix == “q":
….fin()
```

Gestion avec un **dictionnaire**

```
dico_fct={‘l3n’:led_3n, ‘lbp’:led_plus, ‘lbm’:led_moins, ‘q’:fin}

dico_fct.get(choix, fct_erreur)()
```

Si la valeur de choix est inconnue **alors** c'est la fct_erreur qui est appelée

Lorsque **get()** est apellé, Python vérifie que la **clé** demandée existe dans le dictionnaire **dict_fct**

Si oui, **alors**
get() renvoie la valeur de cette **clé**.

Si la **clé** n'existe pas **alors**
get() renvoie la valeur qui est présente dans le deuxième argument de **get()** ➔ fct_erreur

Tuto exercice: led5.py

Travaux Pratique

Modifier le programme pour :

1. gérer le menu par un dictionnaire de fonctions avec l'instruction **.get()** à la place des if; elif; else
   indice s'inspirer de la diapo: numéro

2. où faut-il positionner l'instruction **quit()** pour quitter le programme.

3. passer en paramètre une durée constante pour chaque fonction de 1s

4. ajouter une fonction led_3n qui allume la led verte avec un rapport cyclique de 0.0 ; 0.5 ; 1.0

   ajouter une fonction led_brille_plus qui allume la led verte progressivement par pas de 0.1

   ajouter une fonction led_brille_moins qui éteint la led verte progressivement par pas de 0.1

5. Avec quelle LED faut-il ajouter une fonction led_clignote qui utilise l'instruction **.pulse()** ?

Glossaire

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