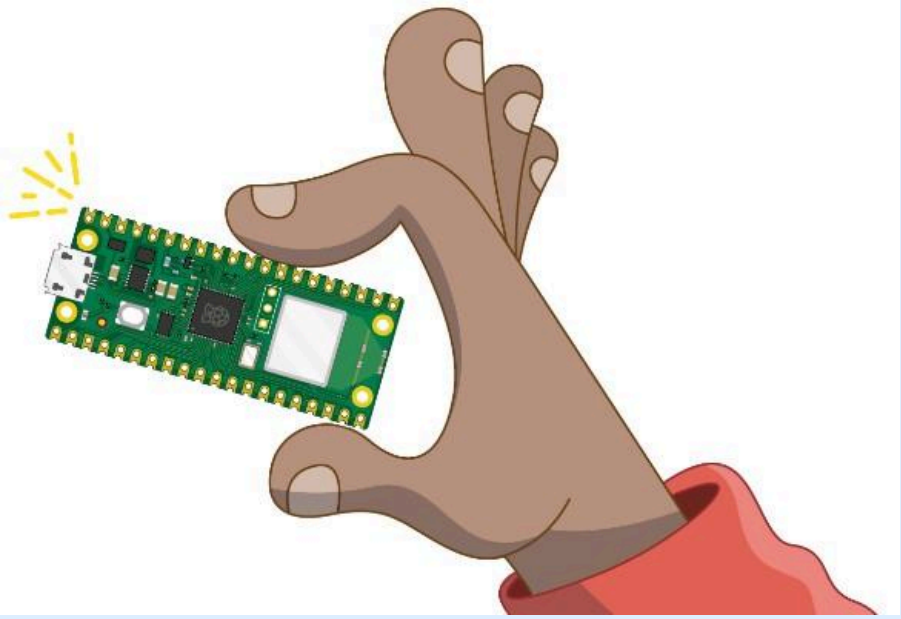
Apprendre à programmer en MicroPython avec la carte Raspberry Pi Pico W

Par Fondation Raspberry Pi - f-leb (traducteur)

Date de publication : 4 février 2023

DÉBUTANT

NDLR : **Raspberry Pi Pico W, la plateforme IoT arrive avec le Wi-Fi intégré à 6 dollars** .



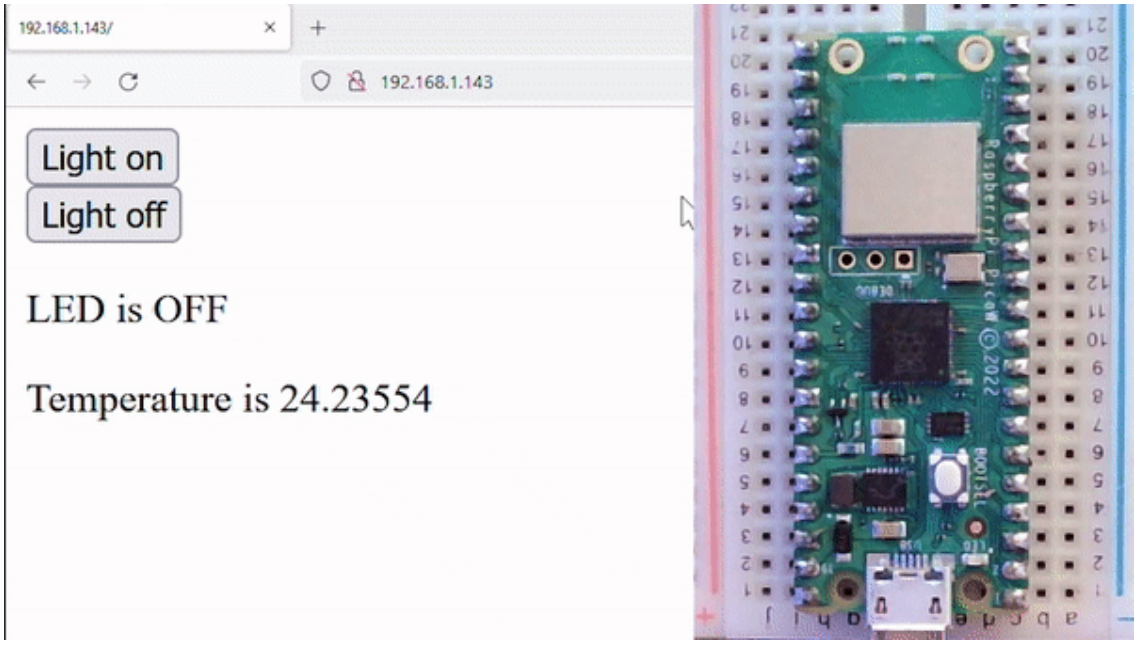
La Raspberry Pi Pico W est la nouvelle carte de la Fondation Raspberry Pi qui ajoute une connectivité Wi-Fi à la **Raspberry Pi Pico**. Dans ce tutoriel, vous apprendrez comment utiliser la Raspberry Pi Pico W, comment la connecter à un réseau Wi-Fi, et aussi comment la configurer en serveur Web pour contrôler ses sorties numériques et afficher les données fournies par des capteurs depuis un navigateur Web.

Commentez

En complément sur Developpez.com

- La fondation Raspberry Pi fait son entrée dans l'univers des microcontrôleurs avec sa nouvelle carte à 4 $ dénommée Pi Pico
- Apprendre à programmer en MicroPython avec la carte Raspberry Pi Pico

I - Introduction



Dans ce tutoriel, vous allez :

- connecter votre Raspberry Pi Pico W à un point d'accès Wi-Fi ;
- créer un serveur Web dans votre Raspberry Pi Pico W, qui servira une page Web ;
- utiliser votre page Web pour piloter la LED intégrée en surface de la carte du Raspberry Pi Pico W et restituer une donnée de température.

Vous aurez besoin :

- d'une carte Raspberry Pi Pico W et d'un câble micro-USB ;
- d'un ordinateur connecté à votre réseau ;
- de l'EDI *Thonny* pour programmer en MicroPython.

I-A - Installer Thonny

- *Thonny* est installé par défaut sur Raspberry Pi OS, mais nécessite peut-être une mise à jour vers la dernière version.
- Ouvrez un terminal, soit en cliquant sur l'icône en haut à gauche de l'écran, soit avec la combinaison de touches Ctrl+Alt+T.
- Dans la fenêtre du terminal, saisissez la commande suivante pour mettre le système d'exploitation et *Thonny* à jour :

```
sudo apt update && sudo apt upgrade -y
```

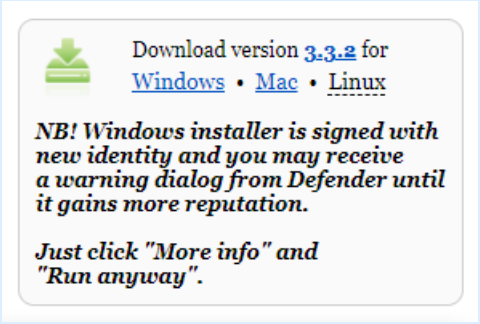
Installer *Thonny* sur un autre système d'exploitation

———

Sous Windows, macOS, et Linux, vous pouvez installer la dernière version de l'EDI *Thonny* ou mettre à jour une version existante :

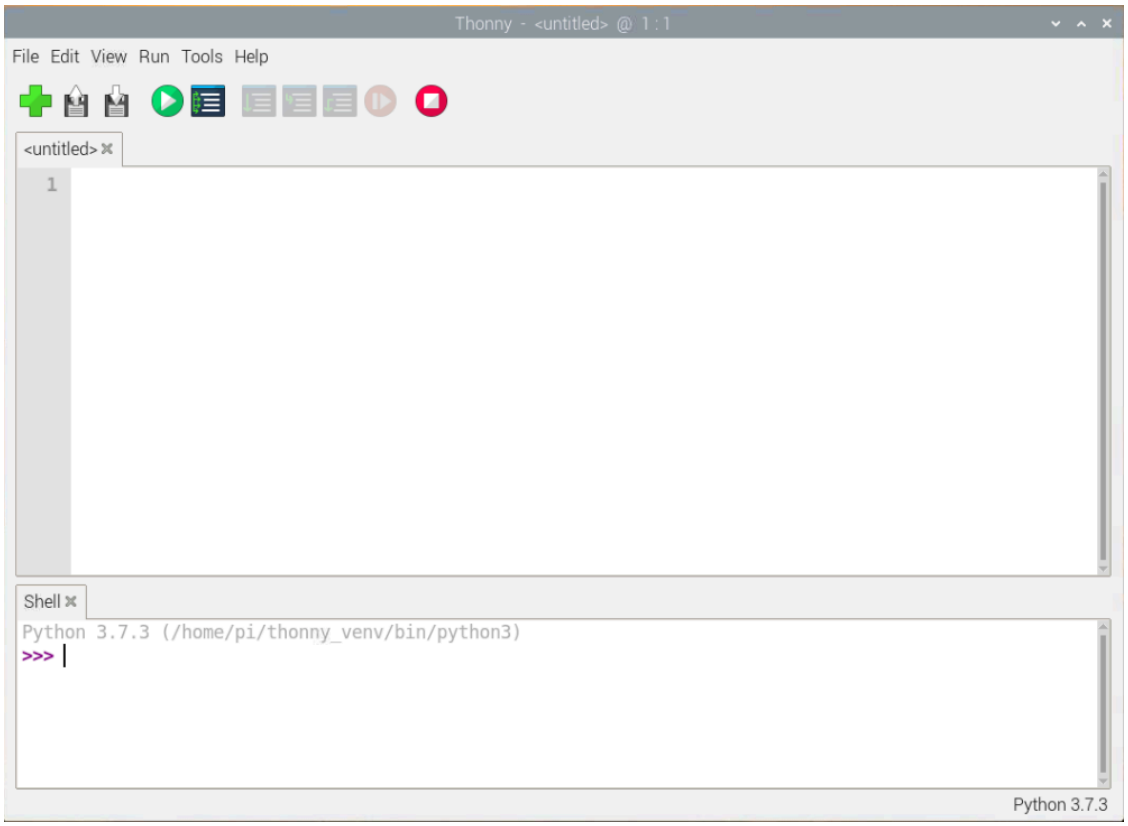
- dans un navigateur Web, rendez-vous sur le site **thonny.org** ;

- en haut et à droite de la fenêtre du navigateur, vous devriez voir le lien de téléchargement pour Windows et macOS, ainsi que les instructions pour Linux ;
- téléchargez les fichiers nécessaires et exécutez-les pour installer *Thonny*.



I-B - Ouvrir Thonny

Ouvrez *Thonny* depuis le lanceur d'applications. Vous devriez voir une fenêtre qui ressemble à ceci :



Vous pouvez maintenant utiliser *Thonny* pour écrire du code en Python standard. Saisissez la ligne de code suivante dans la fenêtre principale, puis cliquez sur le bouton *Run* (*Thonny* vous demandera auparavant de sauvegarder le fichier).

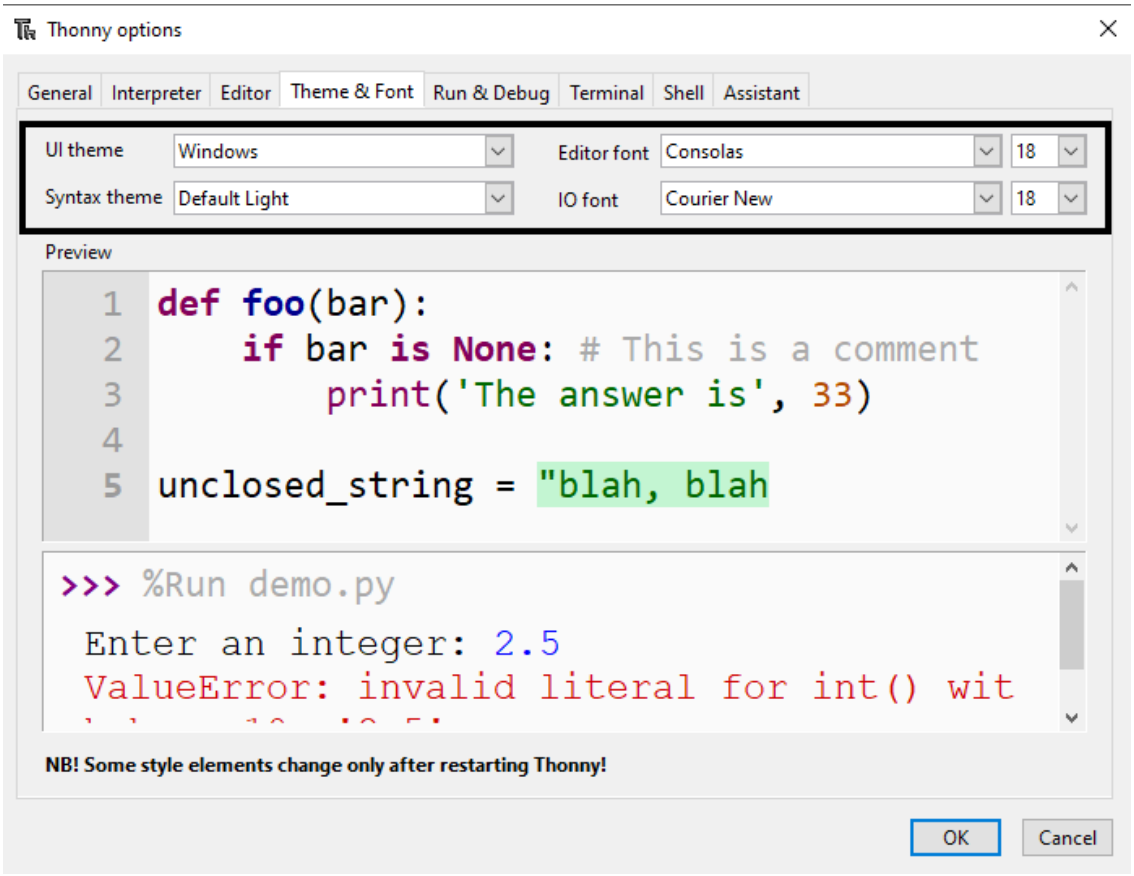
```
print('Hello World!')
```

I-C - Changer le thème d'affichage et la police de caractères dans Thonny

Thonny vous permet de changer le thème d'affichage et la police de caractères. Vous pouvez alors modifier la taille des caractères, le fond et les couleurs du texte en fonction de vos besoins.

Pour changer le thème et la police de caractères :

- cliquez sur *Tools* > *Options* ;
- cliquez sur l'onglet *Theme & Font* ;
- sélectionnez les options qui vous conviennent le mieux depuis les différentes listes déroulantes ;



- cliquez sur le bouton *OK* quand vous avez terminé.

Choisissez des polices de caractères simples et épurées. Choisir une police de caractères manuscrite par exemple peut rendre le code difficile à lire et à déboguer.

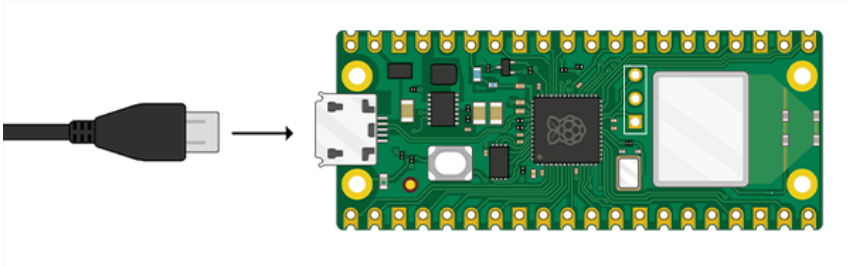
II - Préparer la carte Raspberry Pi Pico W

II-A - Installer le firmware MicroPython

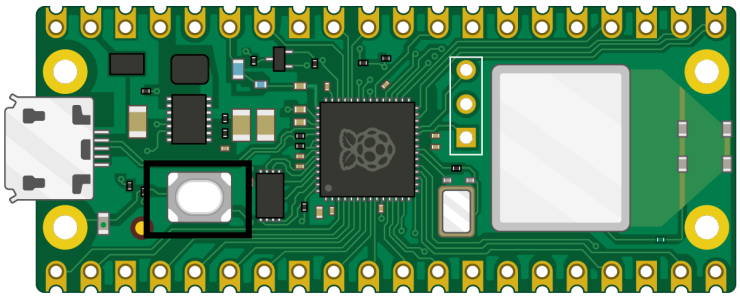
Connectez votre Raspberry Pi Pico W et installez le firmware MicroPython.

MicroPython est une version du langage de programmation Python adaptée aux microcontrôleurs, tel votre Raspberry Pi Pico W. MicroPython permet de réinvestir vos connaissances en Python pour écrire du code et interagir avec des composants électroniques.

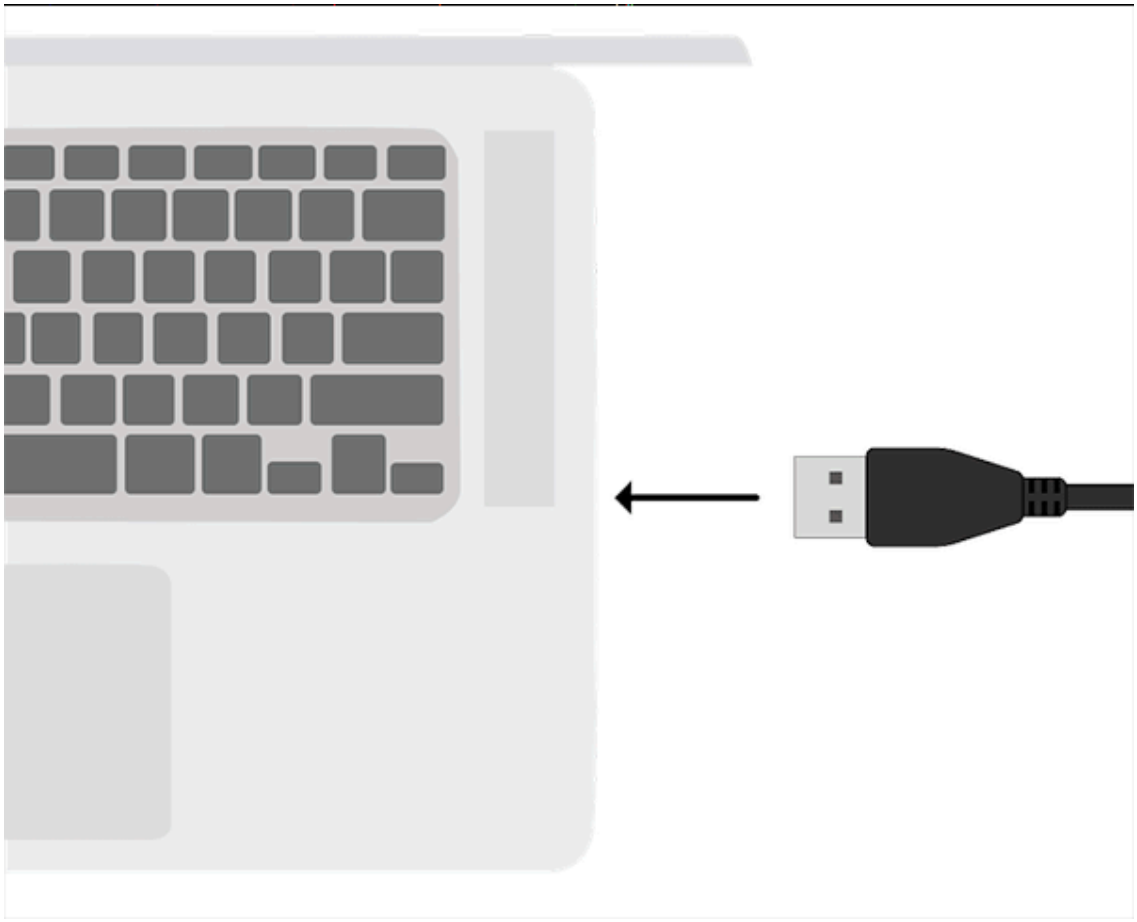
- Téléchargez la dernière version du firmware Raspberry Pi Pico W firmware à l'adresse **https://rpf.io/pico-w-firmware**.
- Connectez le côté micro-USB du câble à la carte Raspberry Pi Pico W.



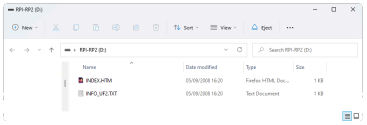
- Maintenez appuyé le bouton **BOOTSEL** de votre Raspberry Pi Pico W.



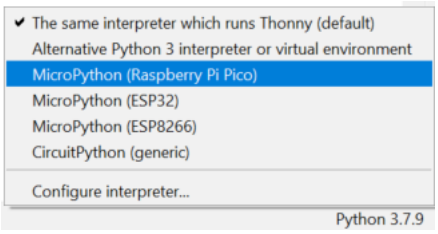
- Connectez l'autre extrémité du câble USB à votre ordinateur de bureau, ordinateur portable, ou Raspberry Pi.



- Le gestionnaire de fichiers devrait s'ouvrir, avec la Raspberry Pi Pico W vue comme un disque externe. Faites « glisser et déposer » le fichier du firmware que vous venez de télécharger dans la fenêtre du gestionnaire de fichiers. Votre Raspberry Pi Pico devrait se déconnecter et la fenêtre du gestionnaire de fichiers devrait se fermer.



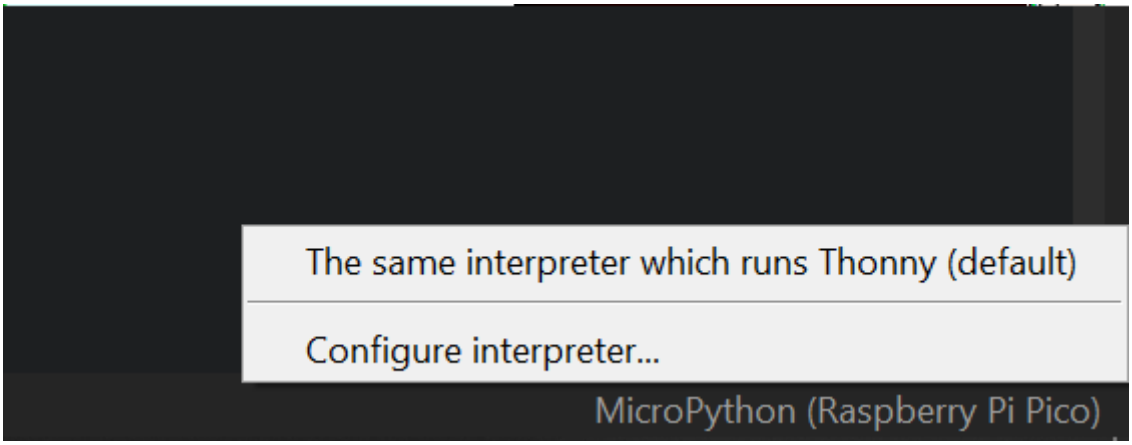
- Ouvrez l'éditeur *Thonny*.
- Observez le texte dans le coin en bas à droite de la fenêtre de *Thonny*. Il vous montrera la version de Python utilisée. À cette étape, vous devez sélectionner l'option *MicroPython (Raspberry Pi Pico)*.



II-B - Problèmes rencontrés

II-B-1 - Je ne sais pas si le firmware est installé et je ne peux pas me connecter à ma carte Pico.

Assurez-vous d'avoir correctement relié votre carte Raspberry Pi Pico W à votre ordinateur avec le câble micro-USB. Cliquez sur la liste en bas à droite de la fenêtre de *Thonny*. Un menu *popup* doit surgir, avec la liste des interpréteurs Python disponibles.



Si vous ne voyez pas la carte Pico dans la liste (comme sur l'image ci-dessus), vous devez reconnecter votre Raspberry Pi Pico W pendant que vous maintenez le bouton BOOTSEL appuyé afin que la carte soit reconnue comme un lecteur externe. Puis réinstallez le firmware en suivant la procédure décrite précédemment.

II-B-2 - Le firmware est installé, mais je ne peux toujours pas me connecter à ma carte Pico.

Peut-être que votre câble est endommagé ou que vous n'utilisez pas le bon type de câble micro-USB. Certains câbles sont conçus uniquement pour alimenter le dispositif et non pour transmettre des données. Commencez par changer de câble si rien ne fonctionne.

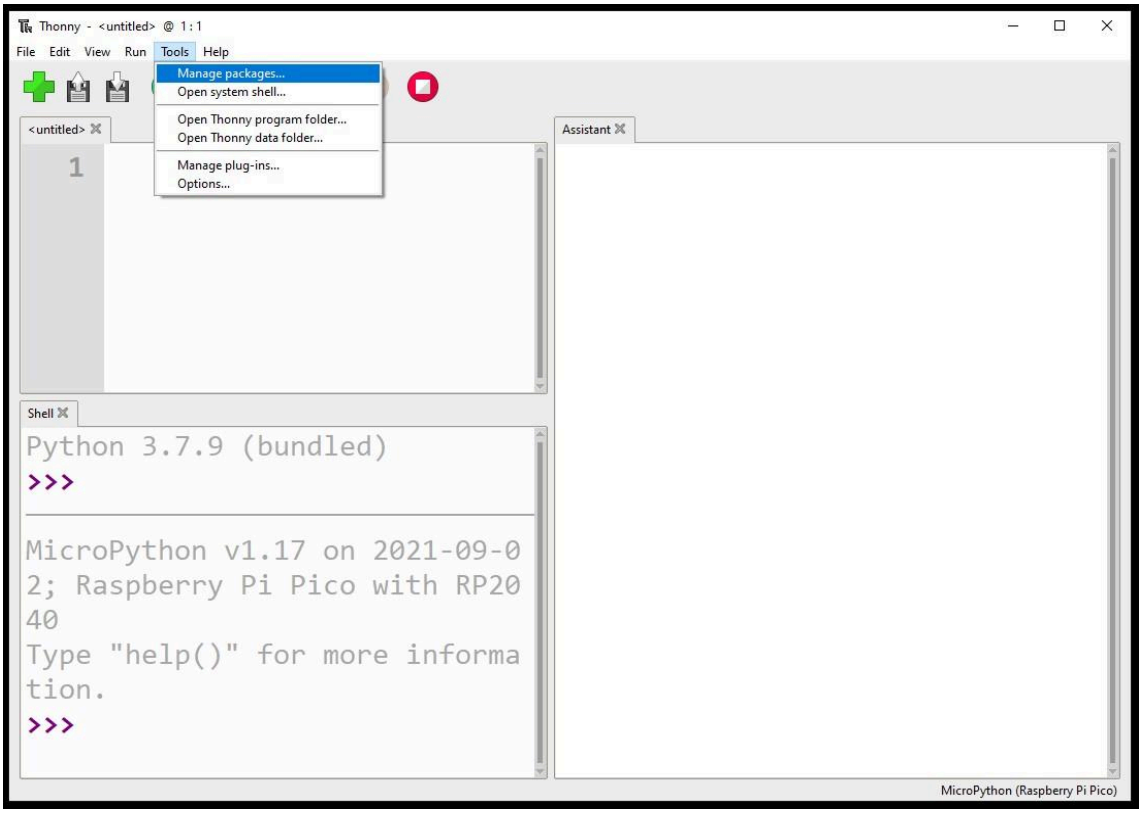
Si votre Pico ne veut toujours pas se connecter, il se peut que votre carte soit elle-même endommagée et donc incapable de se connecter…

II-C - Installer la bibliothèque Picozero

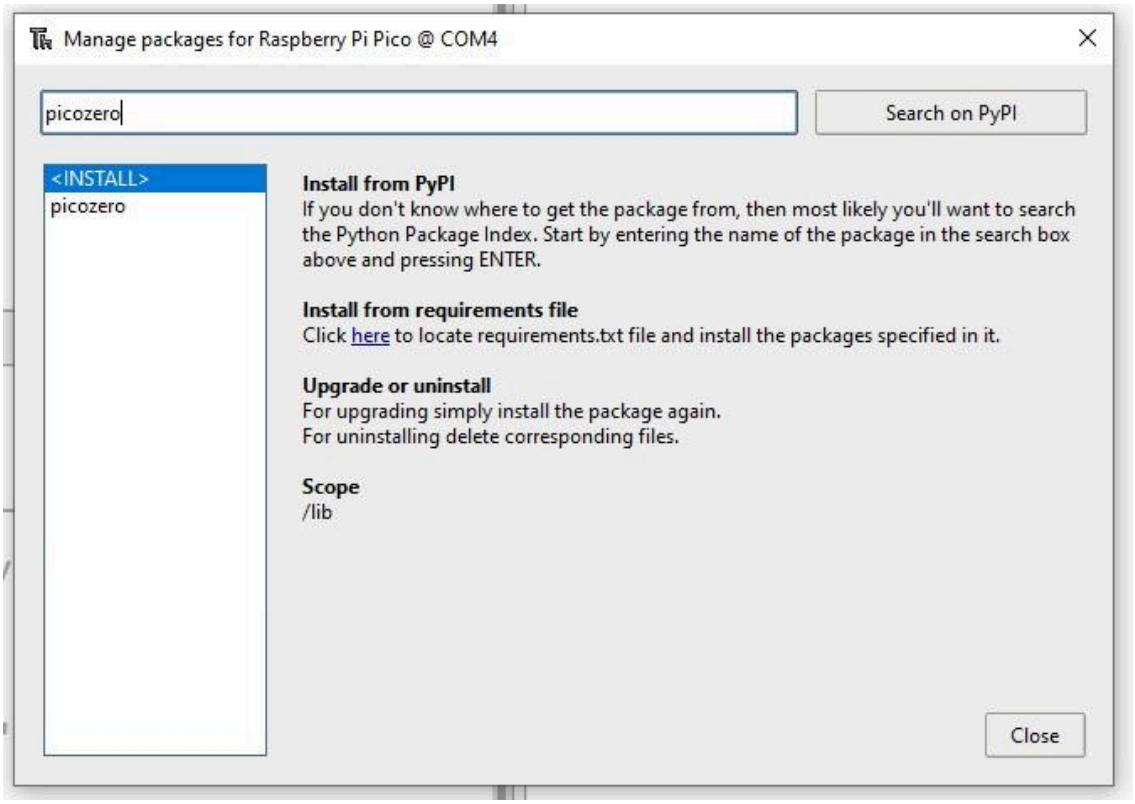
II-C-1 - Installer en ligne

Pour les nouveaux venus sur Raspberry Pi Pico, *picozero* est une bibliothèque MicroPython facile à prendre en main.

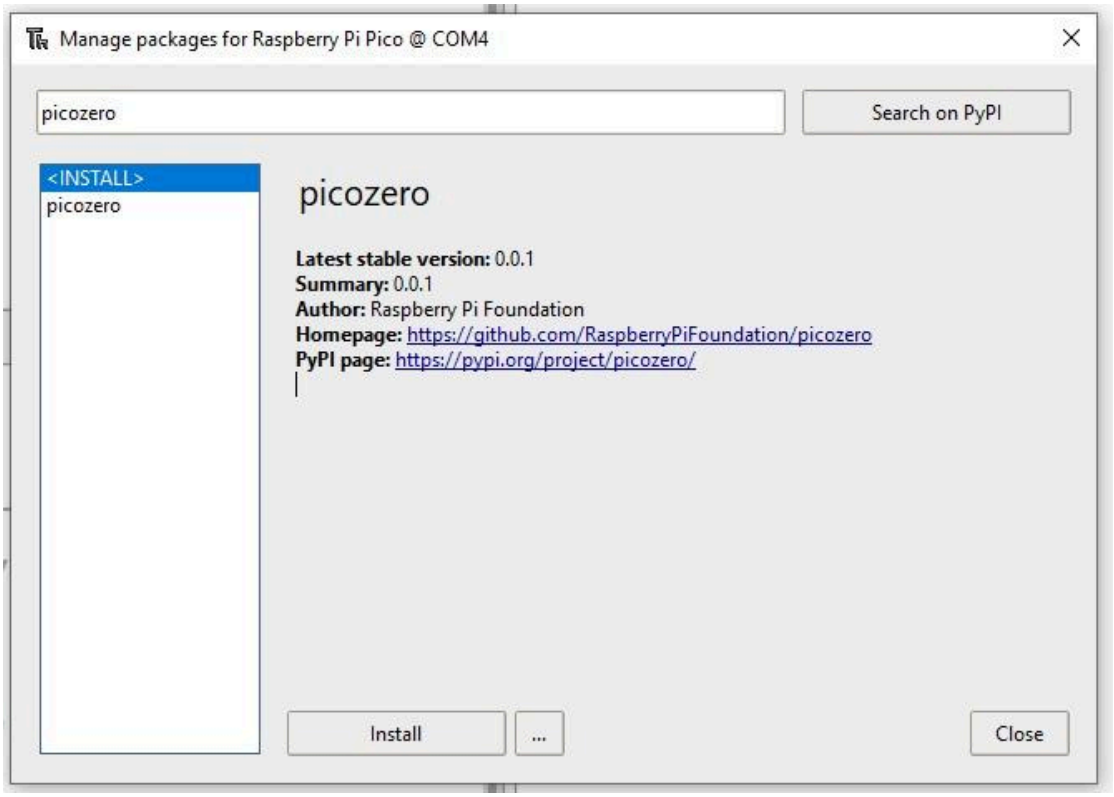
- Pour compléter ce projet, vous devez installer cette bibliothèque *picozero* en chargeant le paquetage. Dans *Thonny*, sélectionnez *Tools* > *Manage packages*.



- Dans le champ texte de la fenêtre *Manage packages for Raspberry Pi Pico*, saisissez *picozero* et cliquez sur le bouton *Search on PyPi*.



- Cliquez sur *picozero* dans la liste des résultats. Puis cliquez sur *Install*.



Quand l'installation est terminée, fermez la fenêtre, quittez puis ouvrez à nouveau *Thonny*.

Si vous avez des difficultés pour installer la bibliothèque *picozero*, vous pouvez télécharger le fichier de la bibliothèque et le sauvegarder pour votre Raspberry Pi Pico W.

II-C-2 - Installer hors-ligne

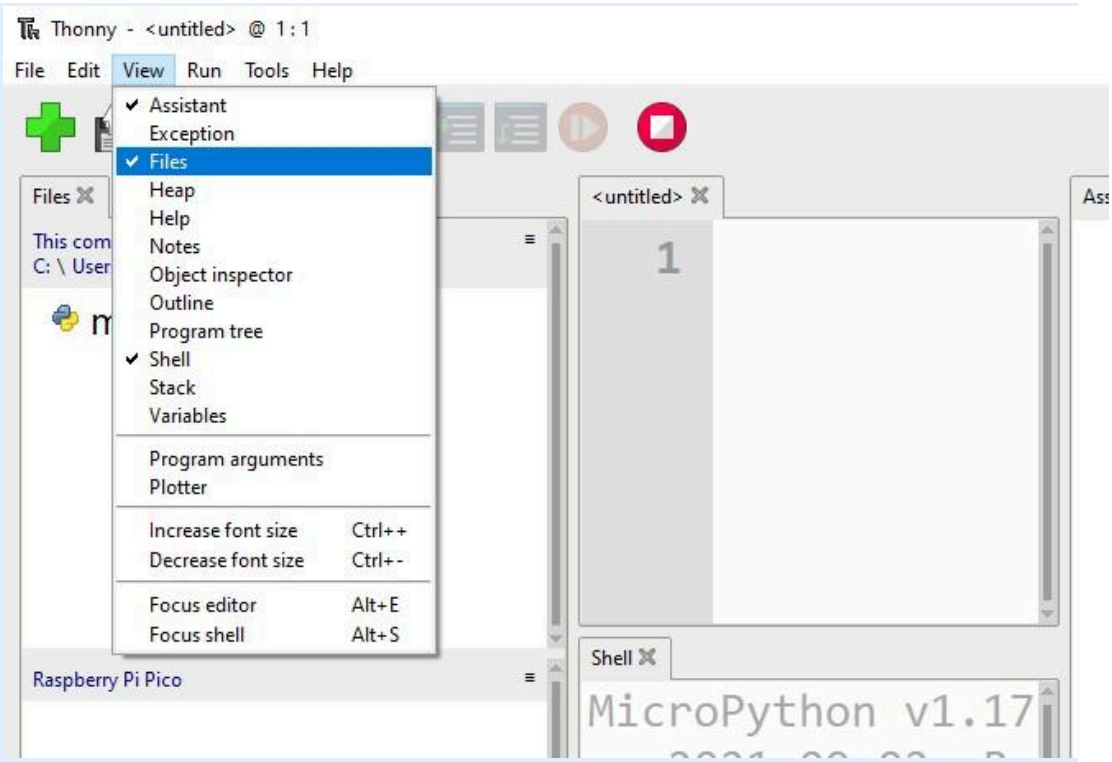
Si vous n'avez pas accès à Internet sur l'ordinateur où est reliée votre carte Raspberry Pi Pico, ou si vous n'avez pas les autorisations nécessaires pour installer des paquetages depuis *Thonny*, il reste un moyen de mettre en œuvre la bibliothèque *picozero*.

Il vous faut tout de même un ordinateur connecté à Internet pour télécharger le fichier de la bibliothèque et le sauvegarder sur une clé USB.

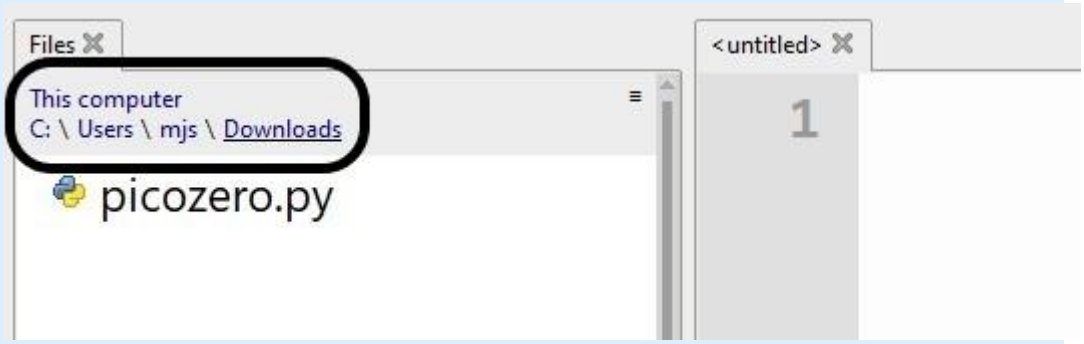
- Récupérez le fichier picozero.py depuis le **dépôt GitHub de picozero** avec un navigateur Web.
- Faites un clic droit sur la page du code, et sélectionnez *Enregistrer sous…*
- Choisissez un répertoire de destination et conservez le nom du fichier picozero.py.

Option 1 – Transfert des fichiers à partir du gestionnaire de fichiers de *Thonny*

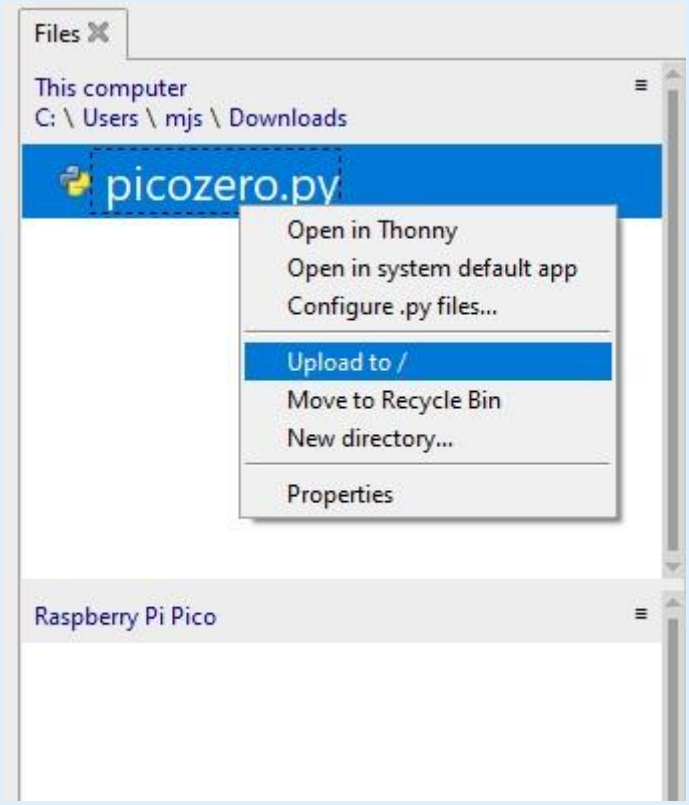
- Connectez votre Raspberry Pi Pico à l'ordinateur avec un câble micro-USB.
- Ouvrez *Thonny*, puis cochez l'option *Files* dans le menu *View*.



- Dans la fenêtre *Files*, naviguez dans les répertoires de votre ordinateur et sélectionnez le chemin où vous avez sauvegardé le fichier picozero.py.



- Faites un clic droit sur le fichier picozero.py et sélectionnez l'option *Upload to /* dans le menu contextuel.

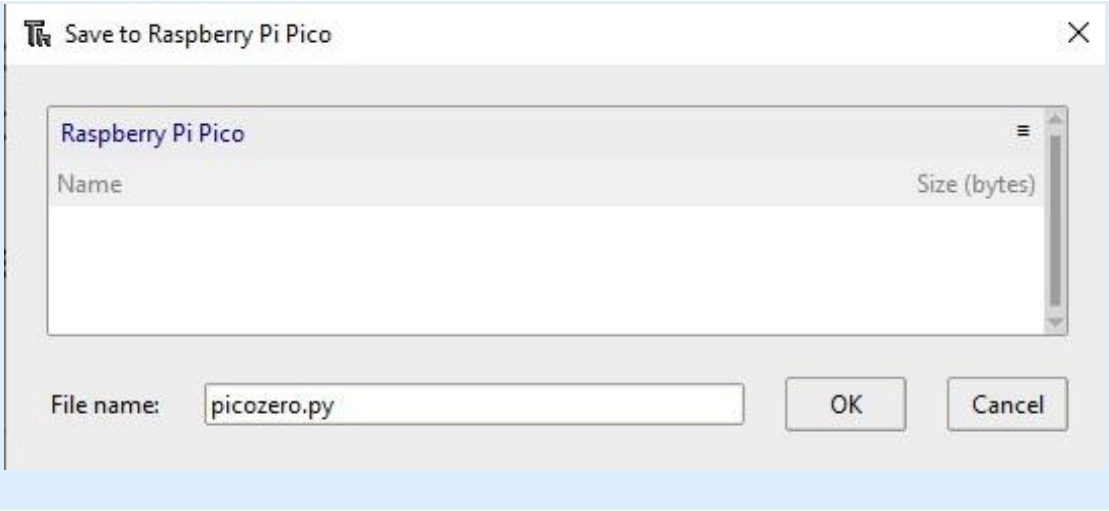


- Vous devriez maintenant apercevoir une copie du fichier picozero.py transféré dans votre carte Raspberry Pi Pico.

Option 2 – Copier et coller le code du fichier dans *Thonny*

- Sélectionnez tout le texte dans le fichier picozero.py grâce à la combinaison de touches Ctrl + a du clavier, puis faites un *copier* avec Ctrl + c.
- Ouvrez *Thonny*, cliquez dans la fenêtre *untitled* et faites un Ctrl + v pour coller le contenu du fichier picozero.py.
- Utilisez Ctrl + s pour sauvegarder le fichier, et cliquez sur le bouton *Raspberry Pi Pico* quand vous sera posée la question du lieu de sauvegarde.

- Nommez le fichier picozero.py et cliquez sur le bouton *OK*.



III - Connecter la carte Raspberry Pi Pico W au réseau local WLAN

Vous allez maintenant apprendre à programmer en MicroPython pour connecter votre carte Raspberry Pi Pico W à un réseau local (WLAN) en Wi-Fi.

```
Shell
MicroPython v1.18-659-g7afd6452f-dirty on 2022-06-15; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 Waiting for connection...
 Waiting for connection...
 Waiting for connection...
 Connected on 192.168.1.143

>>>
```

⚠ Les identifiant et mot de passe doivent rester privés et être conservés en toute sécurité. À cette étape, les identifiant et mot de passe pour accéder à votre réseau Wi-Fi seront ajoutés dans le code du fichier Python. Assurez-vous de ne pas partager le code avec quelqu'un à qui vous ne voudriez pas dévoiler les accès à votre réseau.

Pour vous connecter à un réseau Wi-Fi, vous devez connaître son identifiant SSID et son mot de passe. On peut les trouver sur l'étiquette du routeur, bien que normalement vous devriez au moins avoir changé le mot de passe par défaut.

Dans *Thonny*, importez les bibliothèques nécessaires pour se connecter au réseau Wi-Fi, lire la température du capteur interne et allumer la LED intégrée en surface de la carte.

web_server.py
```
1. import network
2. import socket
3. from time import sleep
4. from picozero import pico_temp_sensor, pico_led
5. import machine
```

Sauvegardez le code maintenant dans votre carte Raspberry Pi Pico W.

Complétez le code avec la définition de l'identifiant SSID et du mot de passe de votre réseau.

web_server.py
```
7. ssid = 'NAME OF YOUR WIFI NETWORK'
8. password = 'YOUR SECRET PASSWORD'
```

Créez maintenant une fonction de connexion à votre réseau local WLAN. Vous devez pour cela définir un objet WLAN, activer le réseau sans fil et fournir l'identifiant SSID et le mot de passe à votre objet.

web_server.py
```
12. def connect():
13.     #Connect to WLAN
14.     wlan = network.WLAN(network.STA_IF)
15.     wlan.active(True)
16.     wlan.connect(ssid, password)
```

Si vous avez déjà connecté un dispositif à un réseau Wi-Fi, vous devez savoir que la connexion n'est pas instantanée. Votre dispositif envoie des requêtes à votre routeur Wi-Fi pour se connecter, et lorsque le routeur répond, s'ensuivent des échanges suivant un protocole pour établir la connexion. Pour faire cela en Python, vous pouvez envoyer en boucle des requêtes chaque seconde, jusqu'à ce que la connexion soit établie.

web_server.py
```
12. def connect():
13.     #Connect to WLAN
14.     wlan = network.WLAN(network.STA_IF)
15.     wlan.active(True)
16.     wlan.connect(ssid, password)
17.     while wlan.isconnected() == False:
18.         print('Waiting for connection...')
19.         sleep(1)
```

Maintenant, affichez la configuration WLAN et testez l'ensemble. Vous devez appeler votre fonction. Les appels sont généralement situés en fin de programme. Comme la connexion Wi-Fi peut persister, même si le programme est stoppé, vous pouvez ajouter un bloc try/except qui réinitialise la carte Raspberry Pi Pico W quand le script est interrompu.

web_server.py
```
12. def connect():
13.     #Connect to WLAN
14.     wlan = network.WLAN(network.STA_IF)
15.     wlan.active(True)
16.     wlan.connect(ssid, password)
17.     while wlan.isconnected() == False:
18.         print('Waiting for connection...')
19.         sleep(1)
20.     print(wlan.ifconfig())
21.
22. try:
23.     connect()
24. except KeyboardInterrupt:
25.     machine.reset()
```

Test : sauvegardez et lancez l'exécution du programme. Vous devriez voir des messages semblables à l'exemple ci-dessous dans la console, mais il est très probable que les adresses IP soient différentes chez vous.

```
Waiting for connection...
Waiting for connection...
Waiting for connection...
Waiting for connection...
Waiting for connection...
('192.168.1.143', '255.255.255.0', '192.168.1.254', '192.168.1.254')
```

La carte Raspberry Pi Pico W ne veut pas se connecter.

- Vérifiez que vos identifiant et mot de passe sont corrects.
- Si vous vous connectez sur le réseau d'un établissement scolaire ou d'une entreprise, il y a sans doute des restrictions d'accès aux dispositifs non autorisés.
- Débranchez votre Raspberry Pi Pico W de votre ordinateur pour l'éteindre, puis rebranchez-le. Cela peut résoudre un problème ponctuel, tentez alors une nouvelle connexion au réseau.

Vous n'avez pas besoin de toutes les informations renvoyées par wlan.ifconfig(). L'information clé dont vous avez besoin est l'adresse IP de votre carte Raspberry Pi Pico W, qui est la première information renvoyée dans la liste. Vous pouvez passer par une fstring pour afficher l'adresse IP. En plaçant un f devant la chaîne de caractères, le contenu des variables encadrées par des accolades {} sera affiché.

web_server.py
```
12. def connect():
13.     #Connect to WLAN
14.     wlan = network.WLAN(network.STA_IF)
15.     wlan.active(True)
16.     wlan.connect(ssid, password)
17.     while wlan.isconnected() == False:
18.         print('Waiting for connection...')
19.         sleep(1)
20.     ip = wlan.ifconfig()[0]
21.     print(f'Connected on {ip}')
22.
23.
24. try:
25.     connect()
26. except KeyboardInterrupt:
27.     machine.reset()
```

Lors de l'appel de la fonction, celle-ci peut maintenant renvoyer l'adresse IP prise par votre carte Raspberry Pi Pico W.

web_server.py
```
12. def connect():
13.     #Connect to WLAN
14.     wlan = network.WLAN(network.STA_IF)
15.     wlan.active(True)
16.     wlan.connect(ssid, password)
17.     while wlan.isconnected() == False:
18.         print('Waiting for connection...')
19.         sleep(1)
20.     ip = wlan.ifconfig()[0]
21.     print(f'Connected on {ip}')
22.     return ip
23.
24.
25. try:
26.     ip = connect()
27. except KeyboardInterrupt:
28.     machine.reset()
```

Sauvegardez votre projet.

IV - Ouvrir un socket

À cette étape, vous allez profiter de la connexion au réseau local WLAN pour ouvrir un *socket*.

Un ***socket*** est la façon dont un **serveur** peut écouter un **client** qui veut se connecter à lui. La page Web que vous êtes en train de consulter est hébergée sur un serveur. Ce serveur a un *socket* ouvert qui attend que votre navigateur Web se connecte, et à ce moment-là, le contenu

de la page Web est envoyé à votre ordinateur. Dans votre cas, la carte Raspberry Pi Pico W sera le serveur, et le client sera tout navigateur Web sur un autre ordinateur.

Pour ouvrir un *socket*, vous devez renseigner l'adresse IP et un numéro de port. Les numéros de port sont utilisés par les ordinateurs pour déterminer où les requêtes doivent être envoyées. Par exemple, le port 80 est normalement utilisé pour le trafic Web. Le jeu *Stardew Valley* utilise le port 24642 quand vous faites une partie en mode multijoueur. Comme vous configurez un serveur Web, vous utiliserez donc le port 80.

Créez une nouvelle fonction `open_socket` qui sera appelée pour ouvrir un *socket*. Elle sera écrite juste au-dessus du bloc `try`/`except`. Commencez par donner au *socket* une adresse IP et un numéro de port.

web_server.py
```
25. def open_socket(ip):
26.     # Open a socket
27.     address = (ip, 80)
28.
29.
30. try:
31.     connect()
32. except KeyboardInterrupt:
33.     machine.reset()
```

Maintenant, créez votre *socket* et demandez-lui d'écouter les requêtes sur le port 80. N'oubliez pas d'appeler votre fonction en fin de programme.

web_server.py
```
25. def open_socket(ip):
26.     # Open a socket
27.     address = (ip, 80)
28.     connection = socket.socket()
29.     connection.bind(address)
30.     connection.listen(1)
31.     print(connection)
32.
33. try:
34.     ip = connect()
35.     open_socket(ip)
36. except KeyboardInterrupt:
37.     machine.reset()
```

Test : exécutez votre programme, vous devriez voir s'afficher le genre de messages ci-dessous.

```
>>> %Run -c $EDITOR_CONTENT
Waiting for connection...
Waiting for connection...
Waiting for connection...
Waiting for connection...
Waiting for connection...
Connected on 192.168.1.143
<socket state=1 timeout=-1 incoming=0 off=0>
```

`socket state=1` signifie que votre *socket* est actif.

Enfin, remplacez le `print` par un `return` pour renvoyer la connexion sous la forme d'objet `socket`.

web_server.py
```
25. def open_socket(ip):
26.     # Open a socket
27.     address = (ip, 80)
28.     connection = socket.socket()
29.     connection.bind(address)
30.     connection.listen(1)
31.     return connection
```

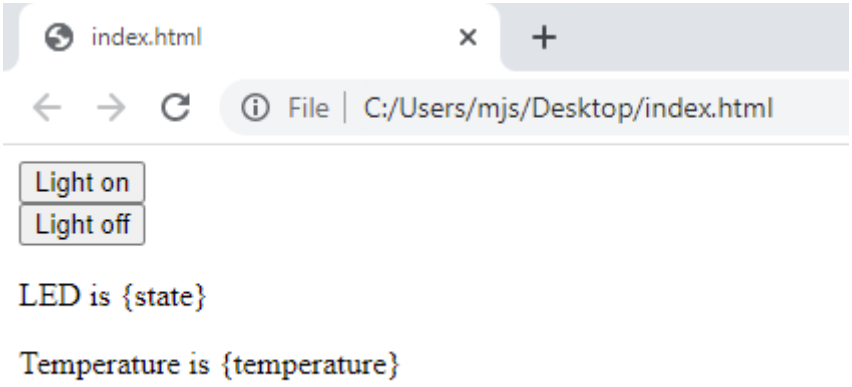
web_server.py
```
32.
33.
34. try:
35.     ip = connect()
36.     connection = open_socket(ip)
37. except KeyboardInterrupt:
38.     machine.reset()
```

Votre carte Raspberry Pi Pico W est maintenant à l'écoute des tentatives de connexion à son adresse IP sur le port 80. Elle est donc prête à servir en retour du code HTML au client Web connecté avec un navigateur qui restituera la page Web à l'utilisateur.

Sauvegardez votre projet.

V - Créer une page Web

Vous allez maintenant créer une page Web que le serveur, qui s'exécute sur votre carte Raspberry Pi Pico W, va envoyer au navigateur client. Vous commencerez par tester votre page Web sur votre ordinateur, pour vérifier qu'elle s'affiche correctement. Ensuite, vous inclurez le code HTML au script Python pour que ce soit la carte Raspberry Pi Pico W qui serve la page Web.



Une page Web peut être décrite dans un simple fichier texte, dans un langage reconnu par un navigateur qui saura restituer la page et même y apporter un peu d'interactivité. Bien que *Thonny* ne soit pas conçu pour écrire du code HTML, on pourra tout de même l'employer à cette occasion. Cependant, vous pouvez utiliser votre éditeur de texte préféré si vous le souhaitez, que ce soit *VSCode*, *TextEdit* ou *Notepad*.

Dans votre éditeur de texte ou dans *Thonny*, créez un nouveau fichier. Vous pouvez le nommer comme vous le voulez, mais il est d'usage de prendre le nom standard `index.html` pour cette première page d'accueil. Le nom du fichier doit se terminer avec l'extension `.html`. Si vous utilisez *Thonny*, sauvegardez votre fichier avec l'option *This computer*.

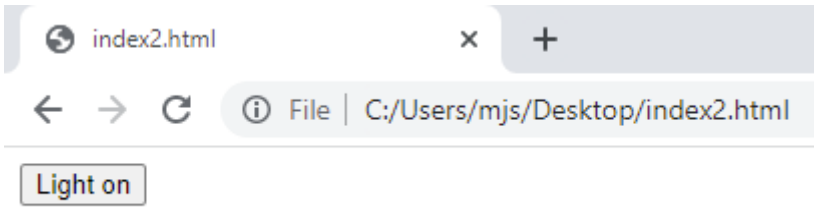
Le squelette minimal d'un code HTML standard inclura les lignes suivantes :

index.html
```
1. <!DOCTYPE html>
2. <html>
3. <body>
4. </body>
5. </html>
```

Puis, vous rajoutez le code du bouton qui servira à allumer la LED intégrée de la carte.

index.html
```
<!DOCTYPE html>
<html>
  <body>
    <form action="./lighton">
      <input type="submit" value="Light on" />
    </form>
  </body>
</html>
```

Sauvegardez votre fichier et retrouvez son emplacement dans le gestionnaire de fichiers. Quand vous double-cliquez sur le fichier, il doit s'ouvrir dans votre navigateur Web par défaut. Voici à quoi devrait ressembler la page Web, ici dans *Google Chrome* :



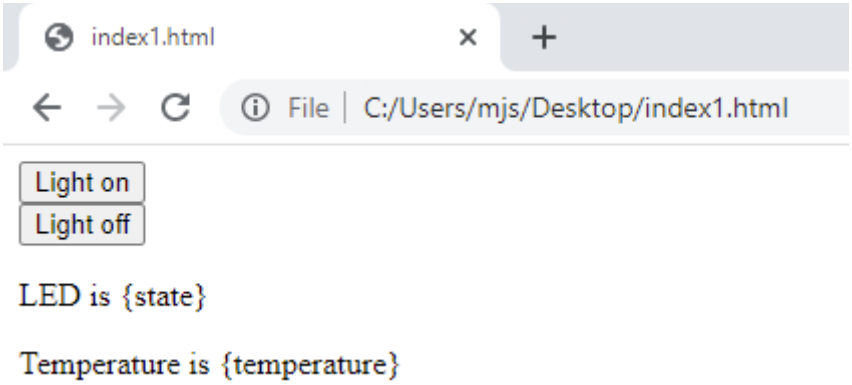
Rajoutez le code du second bouton pour éteindre la LED.

index.html
```
<!DOCTYPE html>
<html>
  <body>
    <form action="./lighton">
      <input type="submit" value="Light on" />
    </form>
    <form action="./lightoff">
      <input type="submit" value="Light off" />
    </form>
  </body>
</html>
```

Pour terminer cette page Web, vous pouvez y ajouter des données supplémentaires comme l'état de la LED ou la température interne de votre Raspberry Pi Pico W.

index.html
```
<!DOCTYPE html>
<html>
  <body>
    <form action="./lighton">
      <input type="submit" value="Light on" />
    </form>
    <form action="./lightoff">
      <input type="submit" value="Light off" />
    </form>
     <p>LED is {state}</p>
     <p>Temperature is {temperature}</p>
  </body>
</html>
```

La page Web devrait maintenant ressembler à ceci :



Maintenant que le rendu est satisfaisant, vous pouvez inclure le code dans votre script Python. Dans *Thonny*, commencez par retourner dans votre code Python.

Créez une nouvelle fonction Python nommée `webpage` avec deux paramètres : `temperature` et `state`.

web_server.py
```
34. def webpage(temperature, state):
35.     #Template HTML
```

Vous pouvez alors stocker dans une variable le code HTML que vous avez écrit et testé. Passer par les fstrings permettra d'insérer le contenu des variables `temperature` et `state` dans les emplacements réservés de la chaîne de caractères.

web_server.py
```
34. def webpage(temperature, state):
35.     #Template HTML
36.     html = f"""
37.             <!DOCTYPE html>
38.             <html>
39.             <form action="./lighton">
40.             <input type="submit" value="Light on" />
41.             </form>
42.             <form action="./lightoff">
43.             <input type="submit" value="Light off" />
44.             </form>
45.             <p>LED is {state}</p>
46.             <p>Temperature is {temperature}</p>
47.             </body>
48.             </html>
49.             """
```

Vous pouvez maintenant faire en sorte de renvoyer la chaîne de caractères du code HTML en sortant de la fonction.

web_server.py
```
34. def webpage(temperature, state):
35.     #Template HTML
36.     html = f"""
37.             <!DOCTYPE html>
38.             <html>
39.             <form action="./lighton">
40.             <input type="submit" value="Light on" />
41.             </form>
42.             <form action="./lightoff">
43.             <input type="submit" value="Light off" />
44.             </form>
45.             <p>LED is {state}</p>
46.             <p>Temperature is {temperature}</p>
47.             </body>
48.             </html>
49.             """
```

web_server.py
```
50.     return str(html)
```

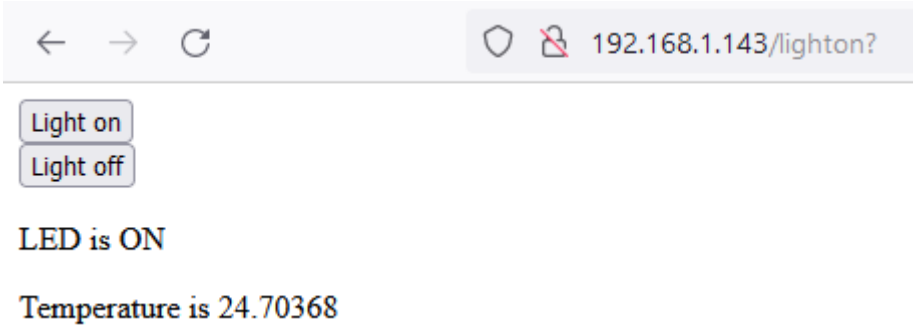
Sauvegardez votre projet.

Vous ne pouvez pas encore tester ce code, car il ne permet pas de servir le code HTML au client. Ce sera l'objectif dans la prochaine étape.

Ce code HTML très simple que vous venez d'écrire sera stocké dans votre script MicroPython et pourra être renvoyé au navigateur Web qui en fera la demande sur votre réseau, comme n'importe quelle page Web stockée dans un serveur quelque part dans le monde. Une différence importante est que seuls les dispositifs connectés à votre réseau Wi-Fi pourront accéder à cette page Web et contrôler votre carte Raspberry Pi Pico W. Cette page est une simple démonstration des possibilités du système, mais il faudra approfondir vos connaissances sur la conception de sites Internet.

VI - Servir une page Web

Dans ce chapitre, vous allez configurer votre serveur Web pour qu'un client puisse s'y connecter, contrôler la LED et consulter la température.



Créez une fonction qui va démarrer le serveur Web, grâce à l'objet `connection` récupéré en paramètre. Les variables `state` et `temperature` doivent être initialisées. L'état de la LED est initialement `'OFF'`, et la valeur de température à 0, il faut donc penser à éteindre la LED au démarrage du serveur.

web_server.py
```
53. def serve(connection):
54.     #Start a web server
55.     state = 'OFF'
56.     pico_led.off()
57.     temperature = 0
```

Quand votre navigateur réclame la connexion à votre Raspberry Pi Pico W, la connexion doit être acceptée. Après cela, les données qui sont envoyées depuis votre navigateur Web doivent être découpées en paquets de taille spécifique (ici, 1024 octets). Vous devez aussi savoir de quoi est faite la requête du navigateur Web — réclame-t-il une simple page ? Ou réclame-t-il une page qui n'existe pas ?

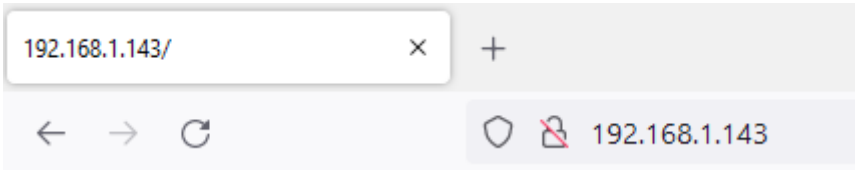
Vous souhaitez garder le serveur Web actif et à l'écoute en permanence, pour qu'un client puisse s'y connecter à tout moment. Vous pouvez faire cela grâce à une boucle infinie `while True :`. Ajouter les cinq lignes de code qui suivent pour que la requête puisse être acceptée, avec un `print()` pour voir le contenu de la requête. Ajoutez un appel à votre fonction `serve` avec les autres appels de fonction en fin de programme.

web_server.py
```
53. def serve(connection):
54.     #Start a web server
55.     state = 'OFF'
```

web_server.py
```
    pico_led.off()
    temperature = 0
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        print(request)
        client.close()


try:
    ip = connect()
    connection = open_socket(ip)
    serve(connection)
except KeyboardInterrupt:
    machine.reset()
```

Test : lancez l'exécution de votre programme et saisissez l'adresse IP dans la barre d'adresse du navigateur Web de votre ordinateur.



Vous devriez voir ce genre de messages dans la console de *Thonny*.

```
>>> %Run -c $EDITOR_CONTENT
Waiting for connection...
Waiting for connection...
Waiting for connection...
Connected on 192.168.1.143
b'GET / HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
 rv:101.0) Gecko/20100101 Firefox/101.0\r\nAccept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\nAccept-Language: en-GB,en;q=0.5\r\nAccept-Encoding:
 gzip, deflate\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\n\r\n'
b'GET /favicon.ico HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0;
 Win64; x64; rv:101.0) Gecko/20100101 Firefox/101.0\r\nAccept: image/avif,image/webp,*/*\r
\nAccept-Language: en-GB,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r
\nReferer: http://192.168.1.143/\r\n\r\n'
```

Ensuite, vous devez envoyer le code HTML au navigateur Web client.

web_server.py
```
def serve(connection):
    #Start a web server
    state = 'OFF'
    pico_led.off()
    temperature = 0
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        print(request)
        html = webpage(temperature, state)
        client.send(html)
        client.close()


try:
    ip = connect()
    connection = open_socket(ip)
    serve(connection)
except KeyboardInterrupt:
```

web_server.py
```
73.     machine.reset()
```

Après avoir relancé le programme, rafraîchissez la page. Cliquez à nouveau sur les boutons. Dans *Thonny*, vous devriez voir deux types de requêtes affichés dans la console :

```
b'GET /lighton? HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0;
 Win64; x64; rv:101.0) Gecko/20100101 Firefox/101.0\r\nAccept: text/html,application/xhtml
+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\nAccept-Language: en-GB,en;q=0.5\r
\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\nReferer: http://192.168.1.143/\r
\nUpgrade-Insecure-Requests: 1\r\n\r\n'
```

et :

```
b'GET /lightoff? HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0;
 Win64; x64; rv:101.0) Gecko/20100101 Firefox/101.0\r\nAccept: text/html,application/xhtml
+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\nAccept-Language: en-GB,en;q=0.5\r
\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\nReferer: http://192.168.1.143/
lighton?\r\nUpgrade-Insecure-Requests: 1\r\n\r\n'
```

Vous notez les informations /lighton? et /lightoff? à l'intérieur des requêtes. On peut les utiliser pour contrôler l'état de la LED en surface de votre carte Raspberry Pi Pico W. Fractionnez la chaîne de caractères de la requête en sous-chaînes (split) et récupérez le deuxième élément dans la liste des sous-chaînes. La chaîne de caractères constituant la requête ne peut pas être fractionnée dans tous les cas, il vaut donc mieux gérer cela dans un bloc try/except.

Si la deuxième sous-chaîne est /lighton?, il faut allumer la LED. Si c'est un /lightoff?, il faut l'éteindre.

web_server.py
```
53. def serve(connection):
54.     #Start a web server
55.     state = 'OFF'
56.     pico_led.off()
57.     temperature = 0
58.     while True:
59.         client = connection.accept()[0]
60.         request = client.recv(1024)
61.         request = str(request)
62.         try:
63.             request = request.split()[1]
64.         except IndexError:
65.             pass
66.         if request == '/lighton?':
67.             pico_led.on()
68.         elif request =='/lightoff?':
69.             pico_led.off()
70.         html = webpage(temperature, state)
71.         client.send(html)
72.         client.close()
```

Exécutez à nouveau votre programme. Cette fois, lorsque vous rafraîchissez la page dans votre navigateur et que vous cliquez sur les boutons, la LED doit s'allumer ou s'éteindre.

Vous pouvez également informer l'utilisateur de la page Web de l'état de la LED.

web_server.py
```
53. def serve(connection):
54.     #Start a web server
55.     state = 'OFF'
56.     pico_led.off()
57.     temperature = 0
58.     while True:
59.         client = connection.accept()[0]
60.         request = client.recv(1024)
61.         request = str(request)
```

web_server.py
```
62.         try:
63.             request = request.split()[1]
64.         except IndexError:
65.             pass
66.         if request == '/lighton?':
67.             pico_led.on()
68.             state = 'ON'
69.         elif request =='/lightoff?':
70.             pico_led.off()
71.             state = 'OFF'
72.         html = webpage(temperature, state)
73.         client.send(html)
74.         client.close()
```

À l'exécution du programme, après rafraîchissement de la page Web, le texte sur l'état de la LED devrait maintenant changer.

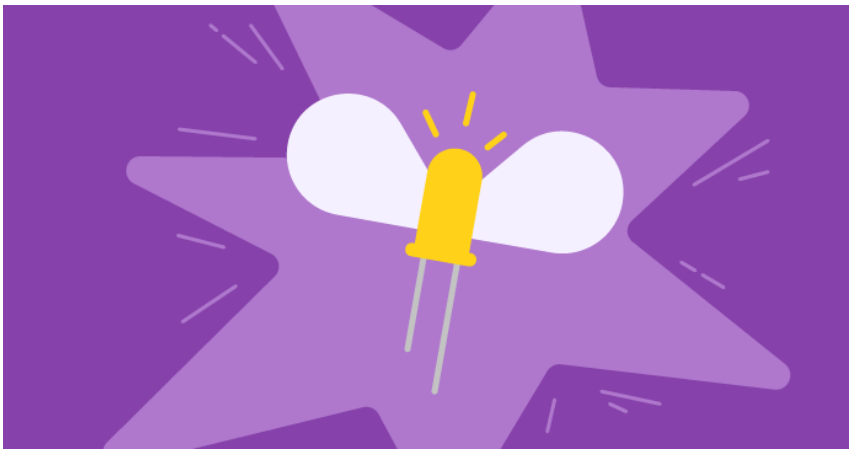
Vous pouvez ensuite utiliser le capteur de température interne pour obtenir une température approximative du microprocesseur, et ainsi l'afficher sur votre page Web.

web_server.py
```
53. def serve(connection):
54.     #Start a web server
55.     state = 'OFF'
56.     pico_led.off()
57.     temperature = 0
58.     while True:
59.         client = connection.accept()[0]
60.         request = client.recv(1024)
61.         request = str(request)
62.         try:
63.             request = request.split()[1]
64.         except IndexError:
65.             pass
66.         if request == '/lighton?':
67.             pico_led.on()
68.             state = 'ON'
69.         elif request =='/lightoff?':
70.             pico_led.off()
71.             state = 'OFF'
72.         temperature = pico_temp_sensor.temp
73.         html = webpage(temperature, state)
74.         client.send(html)
75.         client.close()
```

Test : vous pouvez maintenir votre Raspbberry Pi Pico W dans le creux de votre main pour faire grimper sa température. La nouvelle valeur de température doit s'afficher à chaque fois que vous rafraîchissez la page Web dans votre navigateur.

VII - Et la suite ?

Si vous regardez le document **Introduction to Raspberry Pi Pico: LEDs, buzzers, switches, and dials**, vous trouverez plein d'idées pour améliorer votre serveur Web sur Raspberry Pi Pico W.



Dans le projet **LED firefly**, vous apprendrez à brancher une LED externe à une carte Raspberry Pi Pico et vous pourrez ainsi modifier la façon dont vous faites clignoter votre luciole grâce à une interface Web.

Dans le projet **Party popper**, vous apprendrez comment contrôler une LED RVB et un buzzer grâce à un interrupteur. Vous pourriez compléter ce projet avec un formulaire Web pour décider de la couleur de la LED et du son joué par le buzzer.

Dans le projet **Beating heart**, vous apprendrez à utiliser un potentiomètre pour contrôler la pulsation d'une LED, mais le potentiomètre pourrait aussi servir à modifier l'apparence de votre page Web.

VIII - Notes de la rédaction de Developpez.com

Cet article est une traduction du guide **Getting started with your Raspberry Pi Pico W** proposé par la **Fondation Raspberry Pi**.

Nous remercions les membres de la rédaction de **Developpez.com** pour le travail de traduction et de relecture qu'ils ont effectué, en particulier : **f-leb**, **LittleWhite** et **escartefigue**.