PyPortal Word of the Day Display

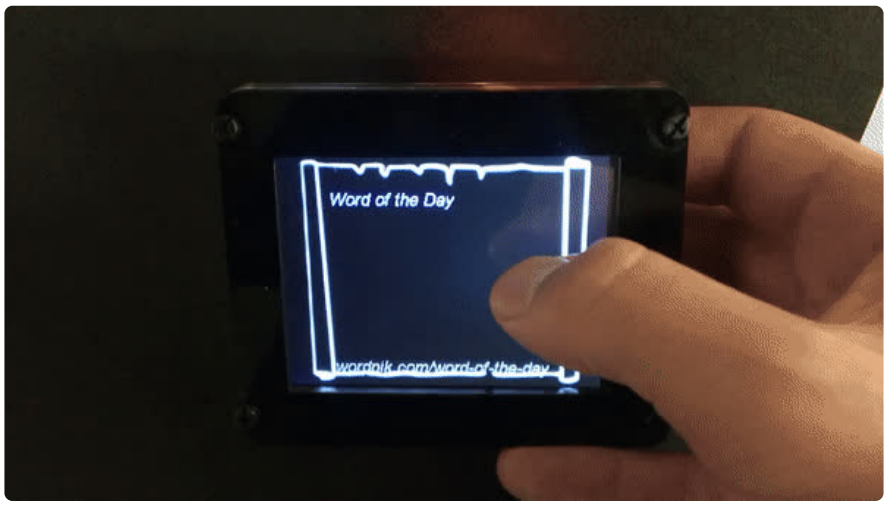
Created by Isaac Wellish

https://learn.adafruit.com/pyportal-word-of-the-day-display

Last updated on 2025-11-05 02:26:53 PM EST

Table of Contents

Overview



Expand your vocabulary by learning a new word every day with this PyPortal-powered display!

Using the Wordnik API and the Adafruit PyPortal, we'll display Wordnik's Word of the Day (https://adafru.it/Eyl), a daily curated word picked by the Wordnik website (https://adafru.it/Eym). With CircuitPython and the on-board WiFi, the PyPortal Word of the Day display dynamically loads JSON formatted data from the Wordnik API (https://adafru.it/Eyn). Each day a new word, its part of speech, definition and an example sentence will appear on the display with just a couple touches, ready for your brain to soak it up!

> The Wordnik API has a one week waiting period for free API keys

Parts

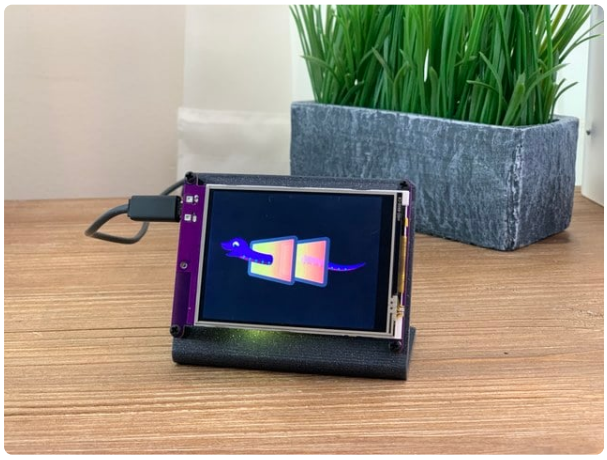
You can pick up an Adafruit PyPortal and a USB cable (if needed). If you like, you can mount the PyPortal in the Adafruit laser-cut acrylic stand. All these parts are bundled in AdaBox 011 if you'd like to buy them together.

AdaBox011 - PyPortal

Reach out beyond your desk - to the stars and beyond - with PyPortal! This ADABOX features a new, easy-to-use IoT device that allows you to customize and create your...

https://www.adafruit.com/product/4061

Instead of AdaBox 011, you can buy parts separately:



Adafruit PyPortal - CircuitPython Powered Internet Display

PyPortal, our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Make custom touch screen interface...

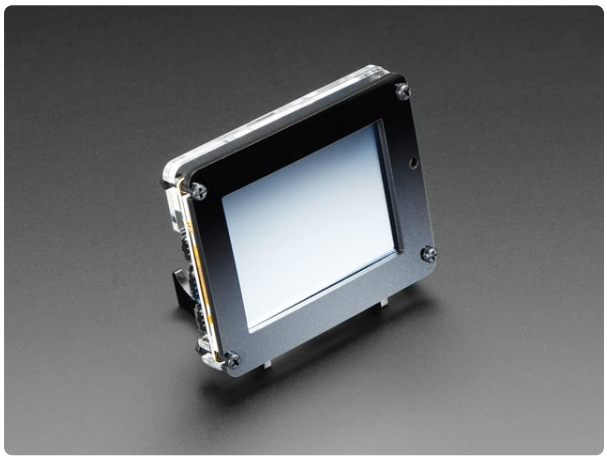
https://www.adafruit.com/product/4116

USB cable - USB A to Micro-B

This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...

https://www.adafruit.com/product/592

Optional



Adafruit PyPortal Desktop Stand Enclosure Kit

PyPortal is our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Create little pocket...

https://www.adafruit.com/product/4146

Install CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** "flash" drive to iterate.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for the PyPortal via CircuitPython.org

https://adafru.it/Egk

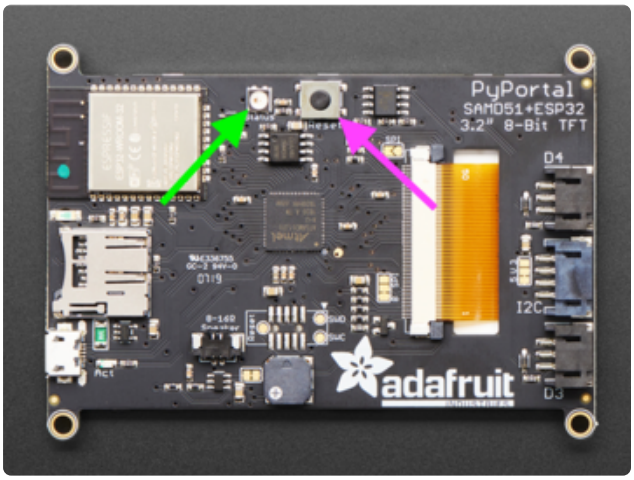
Download the latest version of CircuitPython for the PyPortal Pynt via CircuitPython.org

https://adafru.it/HFd

Click the link above to download the latest version of CircuitPython for the PyPortal.

Download and save it to your desktop (or wherever is handy).

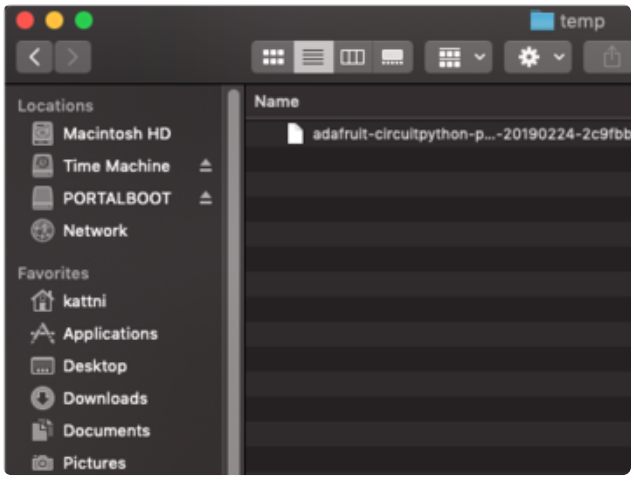
Plug your PyPortal into your computer using a known-good USB cable.

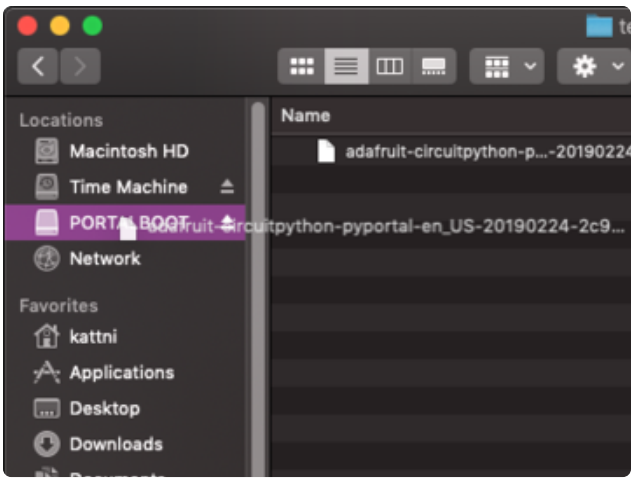


A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the **Reset** button on the top in the middle (magenta arrow) on your board, and you will see the NeoPixel RGB LED (green arrow) turn green. If it turns red, check the USB cable, try another USB port, etc. **Note:** The little red LED next to the USB connector will pulse red. That's ok!

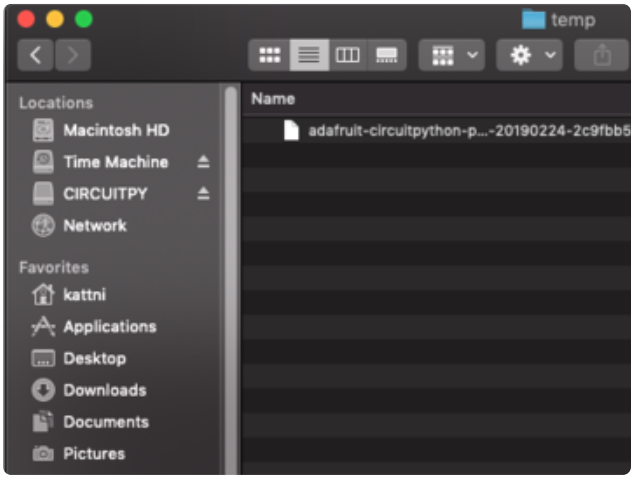
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!





You will see a new disk drive appear called **PORTALBOOT**.

Drag the **adafruit-circuitpython-pyportal-<whatever>.uf2** file to **PORTALBOOT.**



The LED will flash. Then, the **PORTALBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

If you haven't added any code to your board, the only file that will be present is **boot_out.txt**. This is absolutely normal! It's time for you to add your **code.py** and get started!

That's it, you're done! :)

PyPortal Default Files

Click below to download a zip of the files that shipped on the PyPortal or PyPortal Pynt.

PyPortal Default Files

https://adafru.it/UF-

PyPortal Pynt Default Files

https://adafru.it/UGa

PyPortal CircuitPython Setup

To use all the amazing features of your PyPortal with CircuitPython, you must first install a number of libraries. This page covers that process.

Adafruit CircuitPython Bundle

Download the Adafruit CircuitPython Library Bundle. You can find the latest release here:

Latest Adafruit CircuitPython Library Bundle

https://adafru.it/ENC

Download the **adafruit-circuitpython-bundle-*.x-mpy-*.zip** bundle zip file where ***.x MATCHES THE VERSION OF CIRCUITPYTHON YOU INSTALLED**, and unzip a folder of the same name. Inside you'll find a **lib** folder. You have two options:

- You can add the **lib** folder to your **CIRCUITPY** drive. This will ensure you have all the drivers. But it will take a bunch of space on the 8 MB disk
- Add each library as you need it, this will reduce the space usage but you'll need to put in a little more effort.

At a minimum we recommend the following libraries, in fact we more than recommend. They're basically required. So grab them and install them into **CIRCUITPY/lib** now!

- **adafruit_esp32spi** - This is the library that gives you internet access via the ESP32 using (you guessed it!) SPI transport. You need this for anything Internet

- **adafruit_requests** - This library allows us to perform HTTP requests and get responses back from servers. GET/POST/PUT/PATCH - they're all in here!
- adafruit_connection_manager - used by adafruit_requests.
- **adafruit_pyportal** - This is our friendly wrapper library that does a lot of our projects, displays graphics and text, fetches data from the internet. Nearly all of our projects depend on it!
- **adafruit_portalbase** - This library is the base library that adafruit_pyportal library is built on top of.
- **adafruit_touchscreen** - a library for reading touches from the resistive touchscreen. Handles all the analog noodling, rotation and calibration for you.
- **adafruit_io** - this library helps connect the PyPortal to our free datalogging and viewing service
- **adafruit_imageload** - an image display helper, required for any graphics!
- **adafruit_display_text** - not surprisingly, it displays text on the screen
- **adafruit_bitmap_font** - we have fancy font support, and its easy to make new fonts. This library reads and parses font files.
- **adafruit_slideshow** - for making image slideshows - handy for quick display of graphics and sound
- **neopixel** - for controlling the onboard neopixel
- **adafruit_adt7410** - library to read the temperature from the on-board Analog Devices ADT7410 precision temperature sensor (not necessary for Titano or Pynt)
- **adafruit_bus_device** - low level support for I2C/SPI
- **adafruit_fakerequests** - This library allows you to create fake HTTP requests by using local files.

Create Your settings.toml File

CircuitPython works with WiFi-capable boards to enable you to make projects that have network connectivity. This means working with various passwords and API keys. As of CircuitPython 8 (https://adafru.it/Em8), there is support for a **settings.toml** file. This is a file that is stored on your **CIRCUITPY** drive, that contains all of your secret network information, such as your SSID, SSID password and any API keys for IoT services. It is designed to separate your sensitive information from your **code.py** file so you are able to share your code without sharing your credentials.

CircuitPython previously used a **secrets.py** file for this purpose. The **settings.toml** file is quite similar.

Your **settings.toml** file should be stored in the main directory of your **CIRCUITPY** drive. It should not be in a folder.

CircuitPython **settings.toml** File

This section will provide a couple of examples of what your **settings.toml** file should look like, specifically for CircuitPython WiFi projects in general.

The most minimal **settings.toml** file must contain your WiFi SSID and password, as that is the minimum required to connect to WiFi. Copy this example, paste it into your **settings.toml**, and update:

- `your_wifi_ssid`
- `your_wifi_password`

```
CIRCUITPY_WIFI_SSID = "your_wifi_ssid"
CIRCUITPY_WIFI_PASSWORD = "your_wifi_password"
```

Many CircuitPython network-connected projects on the Adafruit Learn System involve using Adafruit IO. For these projects, you must also include your Adafruit IO username and key. Copy the following example, paste it into your settings.toml file, and update:

- `your_wifi_ssid`
- `your_wifi_password`
- `your_aio_username`
- `your_aio_key`

```
CIRCUITPY_WIFI_SSID = "your_wifi_ssid"
CIRCUITPY_WIFI_PASSWORD = "your_wifi_password"
ADAFRUIT_AIO_USERNAME = "your_aio_username"
ADAFRUIT_AIO_KEY = "your_aio_key"
```

Some projects use different variable names for the entries in the **settings.toml** file. For example, a project might use `ADAFRUIT_AIO_ID` in the place of `ADAFRUIT_AIO_USERNAME`. **If you run into connectivity issues, one of the first things to check is that the names in the settings.toml file match the names in the code.**

Not every project uses the same variable name for each entry in the **settings.toml** file! Always verify it matches the code.

settings.toml File Tips

Here is an example **settings.toml** file.

```
# Comments are supported
CIRCUITPY_WIFI_SSID = "guest wifi"
CIRCUITPY_WIFI_PASSWORD = "guessable"
CIRCUITPY_WEB_API_PORT = 80
CIRCUITPY_WEB_API_PASSWORD = "passw0rd"
test_variable = "this is a test"
thumbs_up = "\U0001f44d"
```

In a **settings.toml** file, it's important to keep these factors in mind:

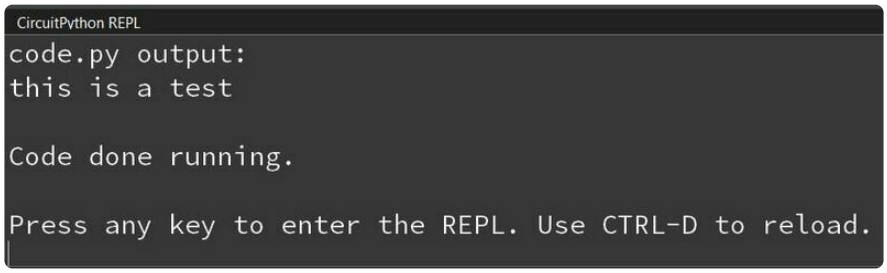
- Strings are wrapped in double quotes; ex: `"your-string-here"`
- Integers are **not** quoted and may be written in decimal with optional sign (`+1`, `-1`, `1000`) or hexadecimal (`0xabcd`).
    - Floats (decimal numbers), octal (`0o567`) and binary (`0b11011`) are not supported.
- Use `\u` escapes for weird characters, `\x` and `\ooo` escapes are not available in **.toml** files
    - Example: `\U0001f44d` for (thumbs up emoji) and `\u20ac` for € (EUR sign)
- Unicode emoji, and non-ASCII characters, stand for themselves as long as you're careful to save in "UTF-8 without BOM" format

When your **settings.toml** file is ready, you can save it in your text editor with the **.toml** extension.

Accessing Your **settings.toml** Information in **code.py**

In your **code.py** file, you'll need to `import` the `os` library to access the **settings.toml** file. Your settings are accessed with the `os.getenv()` function. You'll pass your settings entry to the function to import it into the **code.py** file.

```
import os

print(os.getenv("test_variable"))
```



In the upcoming CircuitPython WiFi examples, you'll see how the **settings.toml** file is used for connecting to your SSID and accessing your API keys.

Internet Connect!

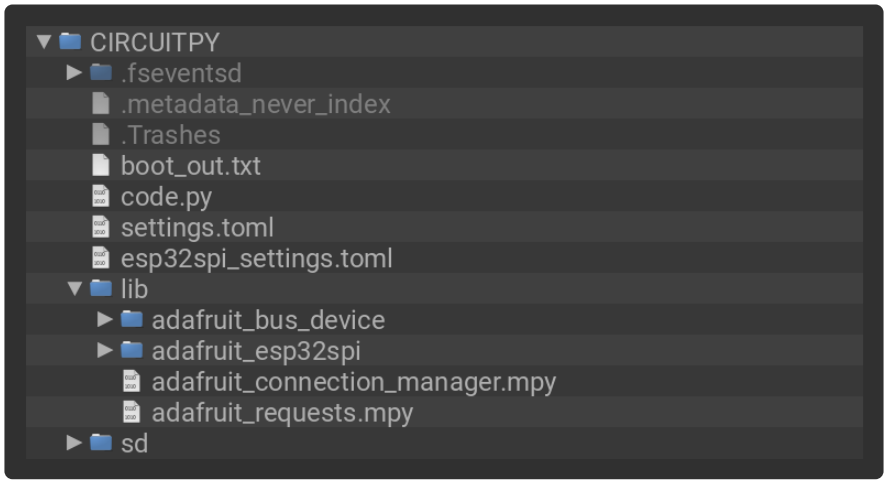
Connect to WiFi

OK, now that you have your **settings.toml** file set up - you can connect to the Internet.

To do this, you need to first install a few libraries, into the lib folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, open the directory **examples/** and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your **CIRCUITPY** drive.

Your **CIRCUITPY** drive should now look similar to the following image:



Update to CircuitPython 9.2.x or later to use this example.

```
# SPDX-FileCopyrightText: 2019 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

from os import getenv

import adafruit_connection_manager
import adafruit_requests
import board
import busio
from digitalio import DigitalInOut

from adafruit_esp32spi import adafruit_esp32spi

# Get wifi details and more from a settings.toml file
# tokens used by this Demo: CIRCUITPY_WIFI_SSID, CIRCUITPY_WIFI_PASSWORD
ssid = getenv("CIRCUITPY_WIFI_SSID")
password = getenv("CIRCUITPY_WIFI_PASSWORD")

print("ESP32 SPI webclient test")
```

```
TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_URL = "http://wifitest.adafruit.com/testwifi/sample.json"


# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an AirLift Shield:
# esp32_cs = DigitalInOut(board.D10)
# esp32_ready = DigitalInOut(board.D7)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

# If you have an externally connected ESP32:
# NOTE: You may need to change the pins to reflect your wiring
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# Secondary (SCK1) SPI used to connect to WiFi board on Arduino Nano Connect RP2040
if "SCK1" in dir(board):
    spi = busio.SPI(board.SCK1, board.MOSI1, board.MISO1)
else:
    spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

pool = adafruit_connection_manager.get_radio_socketpool(esp)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(esp)
requests = adafruit_requests.Session(pool, ssl_context)

if esp.status == adafruit_esp32spi.WL_IDLE_STATUS:
    print("ESP32 found and in idle mode")
print("Firmware vers.", esp.firmware_version)
print("MAC addr:", ":".join("%02X" % byte for byte in esp.MAC_address))

for ap in esp.scan_networks():
    print("\t%-23s RSSI: %d" % (ap.ssid, ap.rssi))

print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(ssid, password)
    except OSError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", esp.ap_info.ssid, "\tRSSI:", esp.ap_info.rssi)
print("My IP address is", esp.ipv4_address)
print("IP lookup adafruit.com: %s" %
esp.pretty_ip(esp.get_host_by_name("adafruit.com")))
print("Ping google.com: %d ms" % esp.ping("google.com"))

# esp._debug = True
print("Fetching text from", TEXT_URL)
r = requests.get(TEXT_URL)
print("-" * 40)
print(r.text)
print("-" * 40)
r.close()

print()
print("Fetching json from", JSON_URL)
r = requests.get(JSON_URL)
print("-" * 40)
```

```
print(r.json())
print("-" * 40)
r.close()

print("Done!")
```

And save it to your board, with the name **code.py**.

Don't forget you'll also need to create the **settings.toml** file as seen above, with your WiFi ssid and password.

In a serial console, you should see something like the following. For more information about connecting with a serial console, view the guide Connecting to the Serial Console (https://adafru.it/Bec).

```
&gt;&gt;&gt; import wifitest
ESP32 SPI webclient test
ESP32 found and in idle mode
Firmware vers. 1.7.5
MAC addr: 24:C9:DC:BD:0F:3F
    HomeNetwork             RSSI: -46
    HomeNetwork             RSSI: -76
    Fios-12345              RSSI: -92
    FiOS-AB123              RSSI: -92
    NETGEAR53               RSSI: -93
Connecting to AP...
Connected to HomeNetwork    RSSI: -45
My IP address is 192.168.1.245
IP lookup adafruit.com: 104.20.39.240
Ping google.com: 30 ms
Fetching text from http://wifitest.adafruit.com/testwifi/index.html
----------------------------------------
This is a test of Adafruit WiFi!
If you can read this, its working :)
----------------------------------------

Fetching json from http://wifitest.adafruit.com/testwifi/sample.json
----------------------------------------
{'fun': True, 'company': 'Adafruit', 'founded': 2005, 'primes': [2, 3, 5], 'pi':
3.14, 'mixed': [False, None, 3, True, 2.7, 'cheese']}
----------------------------------------
Done!
```

Going over the example above, here's a breakdown of what the program is doing:

- Initialize the ESP32 over SPI using the SPI port and 3 control pins:

```
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

#...

else:
```

```
    spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)
```

- Get the socket pool and the SSL context, and then tell the `adafruit_requests` library about them.

```
pool = adafruit_connection_manager.get_radio_socketpool(esp)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(esp)
requests = adafruit_requests.Session(pool, ssl_context)
```

- Verify an ESP32 is found, checks the firmware and MAC address

```
if esp.status == adafruit_esp32spi.WL_IDLE_STATUS:
    print("ESP32 found and in idle mode")
print("Firmware vers.", esp.firmware_version)
print("MAC addr:", ":".join("%02X" % byte for byte in esp.MAC_address))
```

- Perform a scan of all access points it can see and print out the name and signal strength.

```
for ap in esp.scan_networks():
    print("\t%-23s RSSI: %d" % (ap.ssid, ap.rssi))
```

- Connect to the AP we've defined here, then print out the local IP address. Then attempt to do a domain name lookup and ping google.com to check network connectivity. (Note sometimes the ping fails or takes a while; this isn't a big deal.)

```
print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(ssid, password)
    except OSError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", esp.ap_info.ssid, "\tRSSI:", esp.ap_info.rssi)
print("My IP address is", esp.ipv4_address)
print(
    "IP lookup adafruit.com: %s" %
esp.pretty_ip(esp.get_host_by_name("adafruit.com"))
)
```

Now we're getting to the really interesting part of the example program. We've written a library for web fetching web data, named adafruit_requests (https://adafru.it/FpW). It is a lot like the regular Python library named requests (https://adafru.it/1af4). This library allows you to send HTTP and HTTPS requests easily and provides helpful methods for parsing the response from the server.

- Here is the part of the example program is fetching text data from a URL.

```
TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"  # Further up in the
program

# ...

print("Fetching text from", TEXT_URL)
r = requests.get(TEXT_URL)
print('-' * 40)
print(r.text)
print('-' * 40)
r.close()
```

- Finally, here the program is fetching some JSON data. The `adafruit_requests` library will parse the JSON into a Python dictionary whose structure is the same as the structure of the JSON.

```
JSON_URL = "http://wifitest.adafruit.com/testwifi/sample.json"   # Further up in the
program

# ...

print("Fetching json from", JSON_URL)
r = requests.get(JSON_URL)
print('-' * 40)
print(r.json())
print('-' * 40)
r.close()
```

Advanced Requests Usage

Want to send custom HTTP headers, parse the response as raw bytes, or handle a response's http status code in your CircuitPython code?

We've written an example to show advanced usage of the requests module below.

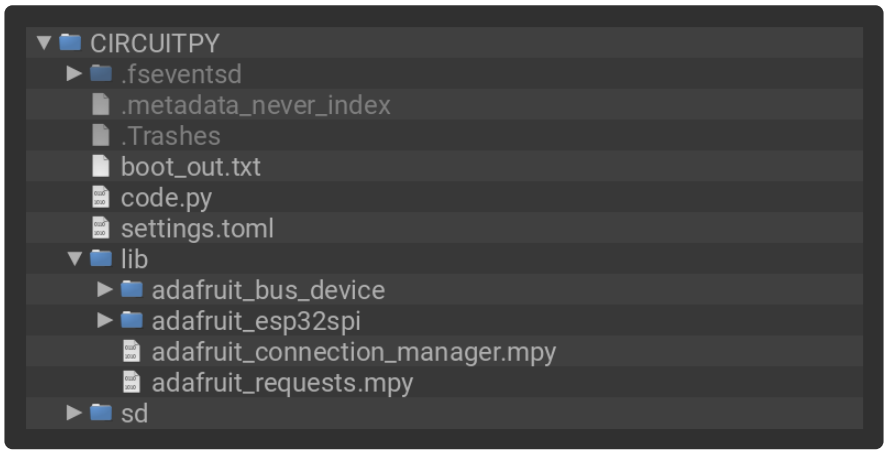
To use with CircuitPython, you need to first install a few libraries, into the lib folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, open the directory **examples/** and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your **CIRCUITPY** drive.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import os

import adafruit_connection_manager
import board
import busio
from adafruit_esp32spi import adafruit_esp32spi
from digitalio import DigitalInOut

import adafruit_requests

# Get WiFi details, ensure these are setup in settings.toml
ssid = os.getenv("CIRCUITPY_WIFI_SSID")
password = os.getenv("CIRCUITPY_WIFI_PASSWORD")

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
radio = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not radio.is_connected:
    try:
        radio.connect_AP(ssid, password)
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
print("Connected to", str(radio.ap_info.ssid, "utf-8"), "\tRSSI:",
radio.ap_info.rssi)

# Initialize a requests session
pool = adafruit_connection_manager.get_radio_socketpool(radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(radio)
requests = adafruit_requests.Session(pool, ssl_context)

JSON_GET_URL = "https://httpbin.org/get"

# Define a custom header as a dict.
headers = {"user-agent": "blinka/1.0.0"}

print("Fetching JSON data from %s..." % JSON_GET_URL)
with requests.get(JSON_GET_URL, headers=headers) as response:
    print("-" * 60)
```

```
    json_data = response.json()
    headers = json_data["headers"]
    print("Response's Custom User-Agent Header: {0}".format(headers["User-Agent"]))
    print("-" * 60)

    # Read Response's HTTP status code
    print("Response HTTP Status Code: ", response.status_code)
    print("-" * 60)
```

Your **CIRCUITPY** drive should now look similar to the following image:



WiFi Manager

The way the examples above connect to WiFi works but it's a little finicky. Since WiFi is not necessarily so reliable, you may have disconnects and need to reconnect. For more advanced uses, we recommend using the `WiFiManager` class. It will wrap the connection/status/requests loop for you - reconnecting if WiFi drops, resetting the ESP32 if it gets into a bad state, etc.

Here's a more advanced example that shows using the `WiFiManager` and also how to fetch the current time from a web source.

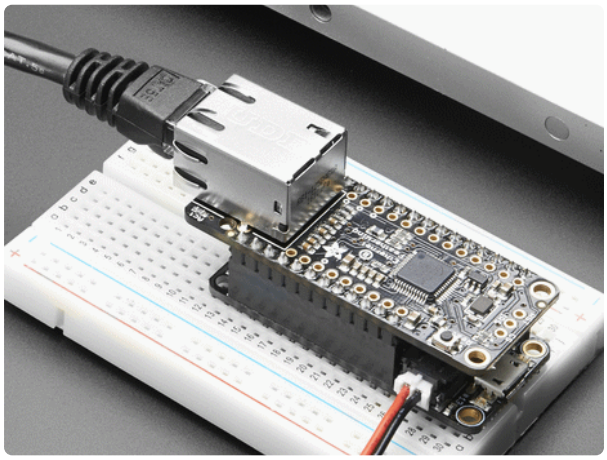
```
# SPDX-FileCopyrightText: 2019 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
from os import getenv

import board
import busio
import neopixel
import rtc
from digitalio import DigitalInOut

from adafruit_esp32spi import adafruit_esp32spi
from adafruit_esp32spi.adafruit_esp32spi_wifimanager import WiFiManager

# Get wifi details and more from a settings.toml file
```

```
# tokens used by this Demo: CIRCUITPY_WIFI_SSID, CIRCUITPY_WIFI_PASSWORD
ssid = getenv("CIRCUITPY_WIFI_SSID")
password = getenv("CIRCUITPY_WIFI_PASSWORD")

print("ESP32 local time")

TIME_API = "http://worldtimeapi.org/api/ip"

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)

# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# Secondary (SCK1) SPI used to connect to WiFi board on Arduino Nano Connect RP2040
if "SCK1" in dir(board):
    spi = busio.SPI(board.SCK1, board.MOSI1, board.MISO1)
else:
    spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

"""Use below for Most Boards"""
status_pixel = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.2)
"""Uncomment below for ItsyBitsy M4"""
# status_pixel = dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1,
brightness=0.2)
"""Uncomment below for an externally defined RGB LED (including Arduino Nano
Connect)"""
# import adafruit_rgbled
# from adafruit_esp32spi import PWMOut
# RED_LED = PWMOut.PWMOut(esp, 26)
# GREEN_LED = PWMOut.PWMOut(esp, 27)
# BLUE_LED = PWMOut.PWMOut(esp, 25)
# status_pixel = adafruit_rgbled.RGBLED(RED_LED, BLUE_LED, GREEN_LED)

wifi = WiFiManager(esp, ssid, password, status_pixel=status_pixel)

the_rtc = rtc.RTC()

response = None
while True:
    try:
        print("Fetching json from", TIME_API)
        response = wifi.get(TIME_API)
        break
    except OSError as e:
        print("Failed to get data, retrying\n", e)
        continue

json = response.json()
current_time = json["datetime"]
the_date, the_time = current_time.split("T")
year, month, mday = (int(x) for x in the_date.split("-"))
the_time = the_time.split(".")[0]
hours, minutes, seconds = (int(x) for x in the_time.split(":"))

# We can also fill in these extra nice things
year_day = json["day_of_year"]
week_day = json["day_of_week"]
is_dst = json["dst"]

now = time.struct_time((year, month, mday, hours, minutes, seconds, week_day,
year_day, is_dst))
print(now)
the_rtc.datetime = now
```

```
while True:
    print(time.localtime())
    time.sleep(1)
```

Further Information

For more information on the basics of doing networking in CircuitPython, see this guide:



Networking in CircuitPython
By Anne Barela

https://learn.adafruit.com/networking-in-circuitpython

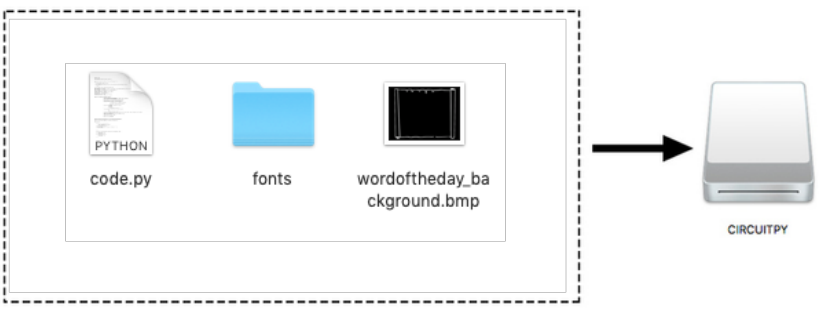
Code PyPortal with CircuitPython

CircuitPython Code

In the embedded code element below, click on the **Download Project Bundle** button, and save the .zip archive file to your computer.

Then, uncompress the .zip file, it will unpack to a folder named **PyPortal_Word_of_the_Day**.

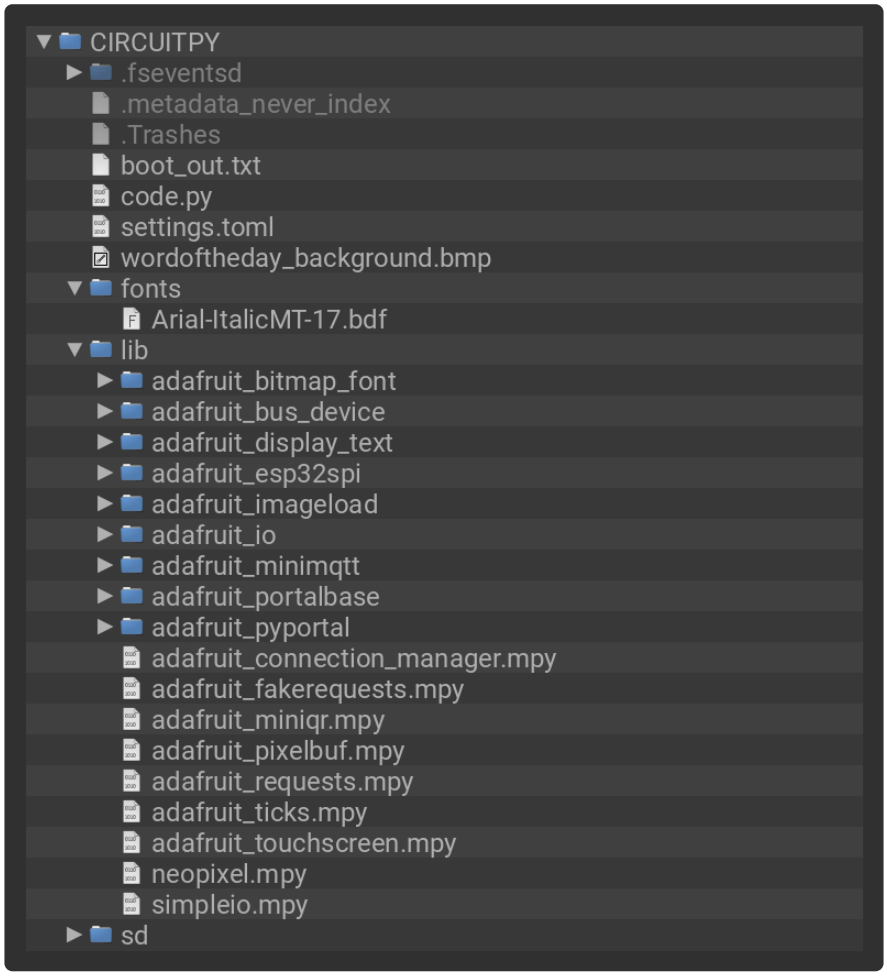
Copy the contents of the **PyPortal_Word_of_the_Day** directory to your PyPortal **CIRCUITPY** drive.



```
# SPDX-FileCopyrightText: 2019 Isaac Wellish for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
This example uses the Wordnik API to display Wordnik's Word of the Day.
Each day a new word, its part of speech, and definition
will appear automatically on the display. Tap the screen to start
as well as to switch between the word's definition and an example sentence.
"""
import os

import board
from adafruit_pyportal import PyPortal

# Set up where we'll be fetching data from
DATA_SOURCE = "https://api.wordnik.com/v4/words.json/wordOfTheDay?api_key=" +
os.getenv(
    "WORDNIK_TOKEN"
)
PART_OF_SPEECH = ["definitions", 0, "partOfSpeech"]
DEF_LOCATION = ["definitions", 0, "text"]
EXAMPLE_LOCATION = ["examples", 0, "text"]
CAPTION = "wordnik.com/word-of-the-day"
DEFINITION_EXAMPLE_ARR = [DEF_LOCATION, EXAMPLE_LOCATION]
# defintion and example array variable initialized at 0
definition_example = 0

# determine the current working directory
# needed so we know where to find files
cwd = ("/" + __file__).rsplit("/", 1)[0]

# Initialize the pyportal object and let us know what data to fetch and where
# to display it
pyportal = PyPortal(
    url=DATA_SOURCE,
    status_neopixel=board.NEOPIXEL,
    default_bg=cwd + "/wordoftheday_background.bmp",
    text_font=cwd + "/fonts/Arial-ItalicMT-17.bdf",
    text_position=(
        (50, 30),  # word location
        (50, 50),  # part of speech location
        (50, 135),
    ),  # definition location
    text_color=(0x8080FF, 0xFF00FF, 0xFFFFFF),
    text_wrap=(0, 0, 28),  # characters to wrap for text
    text_maxlen=(180, 30, 115),  # max text size for word, part of speech and def
    caption_text=CAPTION,
    caption_font=cwd + "/fonts/Arial-ItalicMT-17.bdf",
    caption_position=(50, 220),
    caption_color=0x808080,
)

print("loading...")  # print to repl while waiting for font to load
pyportal.preload_font()  # speed things up by preloading font
pyportal.set_text("\nWord of the Day")  # show title

while True:
    if pyportal.touchscreen.touch_point:
        try:
            # set the JSON path here to be able to change between definition and
example
            # pylint: disable=protected-access
            pyportal._json_path = (
                ["word"],
                PART_OF_SPEECH,
                DEFINITION_EXAMPLE_ARR[definition_example],
            )
```

```
            value = pyportal.fetch()
            print("Response is", value)
        except RuntimeError as e:
            print("Some error occured, retrying! -", e)
        # Change between definition and example
        if definition_example == 0:
            definition_example = 1
        elif definition_example == 1:
            definition_example = 0
```

This is what the final contents of the **CIRCUITPY** drive will look like:



The Wordnik API

Wordnik is where we'll be grabbing our word of the day data from to display on the PyPortal.

A bit about Wordnik from their website (https://adafru.it/EyY):

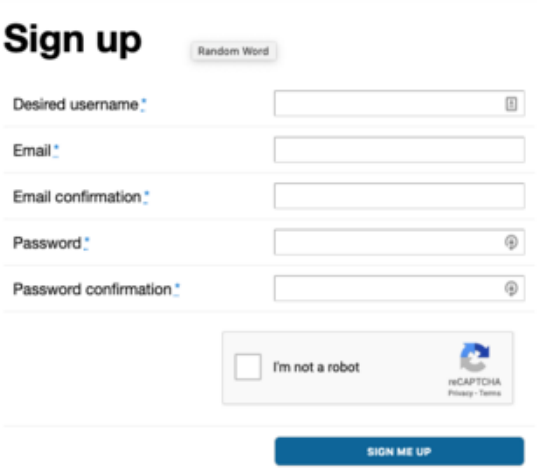
> Wordnik is the world's biggest online English dictionary (https://adafru.it/EyZ), by number of words.

> Wordnik is a 501(c)(3) nonprofit organization, and our mission is to find and share as many words of English as possible with as many people as possible.

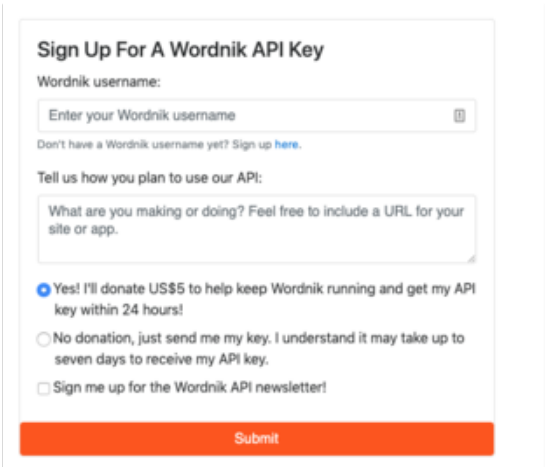
In order to use Wordnik's API and access their word data, we will need what's called an API key.

> The Wordnik API has a one week waiting period for free API keys

How to obtain an API key from Wordnik



To get an API key first make an account on the Wordnik website by clicking here (https://adafru.it/Ey-).



Next, head here (https://adafru.it/Eyn) to sign up for an API key.
If you donate $5 or more you can get the API key the same day.
You can also get a key for free **but will have to wait one week to get it.**

Using the API key

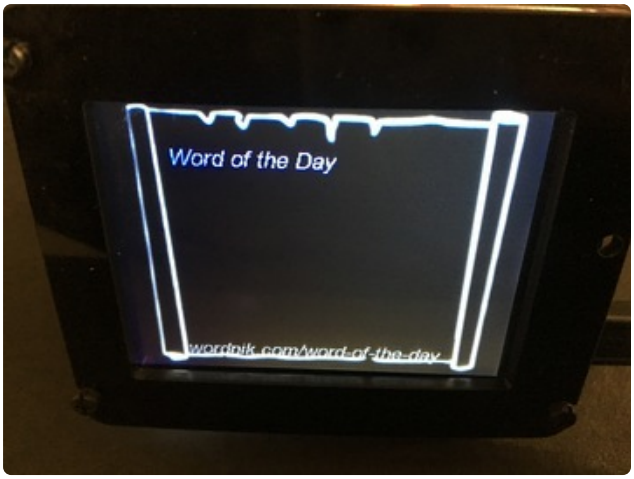
After you've gotten your key from Wordnik in your email, it's time to add it to your **settings.toml** file:

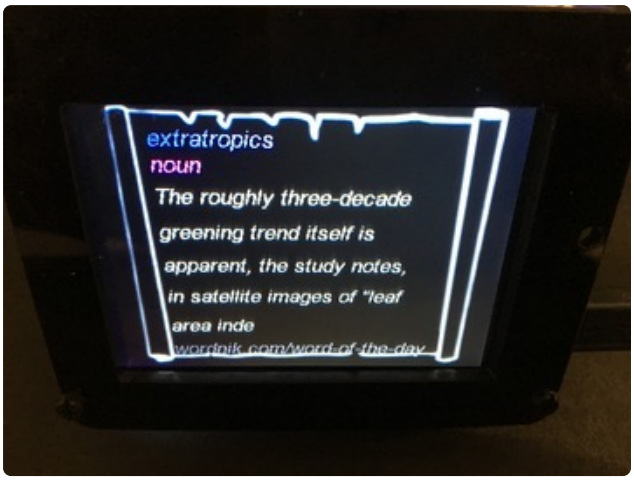
```
CIRCUITPY_WIFI_SSID = "your_wifi_ssid"
CIRCUITPY_WIFI_PASSWORD = "your_wifi_password"
WORDNIK_TOKEN = "HUGE_LONG_WORDNIK_API_KEY"
```

If you run into any errors, such as "ImportError: no module named `adafruit_display_text.label`" be sure to update your libraries to the latest release bundle!

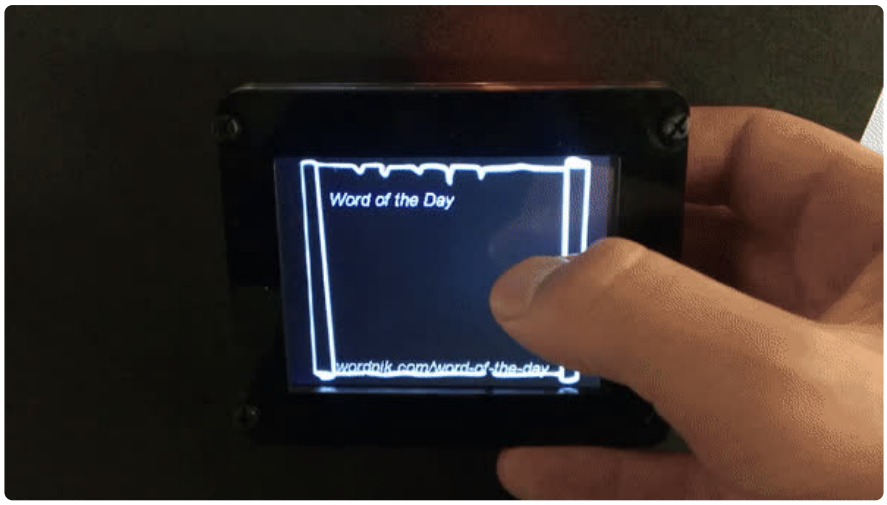
Title Sequence

When the PyPortal is turned on or reset, the following events occur.





Display the title of the app, "Word of the Day" at the top of the display and the source "Wordnik.com/wordoftheday" at the bottom
When screen is touched, display the word, part of speech, and definition with the source at the bottom.
When screen is touched again, display the word, part of speech, and an example sentence with the word.

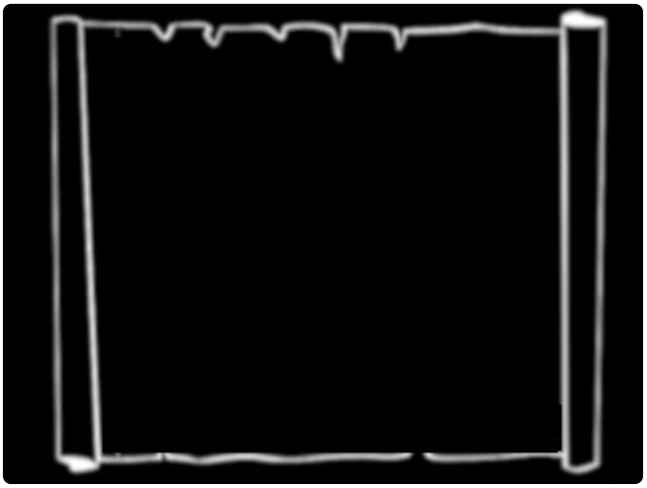


How It Works

The PyPortal Word of the Day is doing a couple of nifty things to deliver your insightful vocabulary expanding experience!

Background

First, it displays a bitmap graphic as the screen's background. This is a 320 x 240 pixel RGB 16-bit raster graphic in **.bmp** format. I found this image on Pixabay (https://adafru.it/Ez0) by user CIker-Free-Vector-Images (https://adafru.it/Ez1) and turned it into a black and white scroll for ultimate word of the day background!



Font

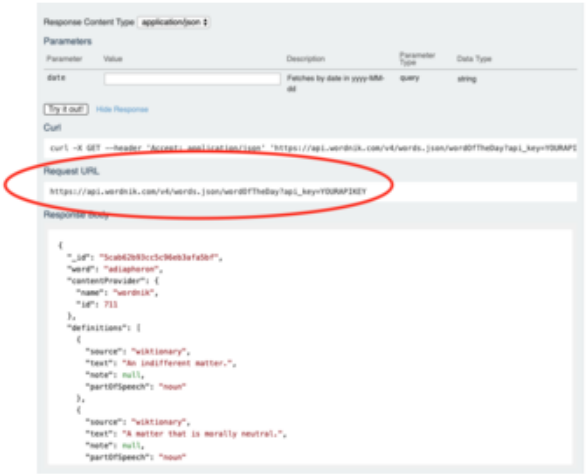
First we see "Word of the Day" appear on the screen along with the source at the bottom.

The fonts used here are bitmap fonts made from the Arial Italic typeface. You can learn more about converting type in this guide (https://adafru.it/E7E).

Parsing JSON from the Web

The neat part is that the text is not coming from a file on the device, but rather it is grabbed from a website!

From the Wordnik API's documentation (https://adafru.it/Ez2) for the word of the day, click the "Try it out!" button.



A link can be found in the Request URL section.
Try copy and pasting the below link into your browser and replace "YOURAPIKEY" with key you were given form Wordnik.
When this link is entered into a browser the JSON data for that word shows up!

```
https://api.wordnik.com/v4/words.json/wordOfTheDay?api_key=YOURAPIKEY
```

You may see a large mess of numbers, letters and symbols. This is JSON data!

Head here (https://adafru.it/Eb5) to beautify the code so we can actually read it!

"Beautified" code:

```
{
  "_id": "5cab62b93cc5c96eb3afa5bf",
  "word": "adiaphoron",
  "contentProvider": {
    "name": "wordnik",
    "id": 711
  },
```

```
  "definitions": [
    {
      "source": "wiktionary",
      "text": "An indifferent matter.",
      "note": null,
      "partOfSpeech": "noun"
    },
    {
      "source": "wiktionary",
      "text": "A matter that is morally neutral.",
      "note": null,
      "partOfSpeech": "noun"
    },
    {
      "source": "wiktionary",
      "text": "Something neither forbidden nor commanded by scripture.",
      "note": null,
      "partOfSpeech": "noun"
    },
    {
      "source": "century",
      "text": "In theology and ethics, a thing indifferent; a tenet or practice
which may be considered non-essential.",
      "note": null,
      "partOfSpeech": "noun"
    }
  ],
  "publishDate": "2019-04-12T03:00:00.000Z",
  "examples": [
    {
      "url": "http://api.wordnik.com/v4/mid/
e1284319bcc8eed2b40528c240f6d69376b90187bd37ec0240e9907527a238de12ff8f433fbda52a4181ae99f4ade6
      "title": "Historical Introductions to the Symbolical Books of the Evangelical
Lutheran Church",
      "text": "For if the condition is good, the adiaphoron, too, is good, and its
observance is commanded.",
      "id": 1090967175
    },
    {
      "url": "http://api.wordnik.com/v4/mid/
6542bbb01a6a8314704c23814788a11d6197fd480404393c5fbd590e84fe27d6ad9ae69016e66722bfa1d945c00d2a
      "title": "Apology of the Augsburg Confession",
      "text": "For Augustine, in a long discussion refutes the opinion of those who
thought that concupiscence in man is not a fault but an adiaphoron, as color of the
body or ill health is said to be an adiaphoron [as to have a black or a white body
is neither good nor evil].",
      "id": 1158602656
    }
  ],
  "pdd": "2019-04-12",
  "note": "The word 'adiaphoron' comes from a Greek word meaning 'indifferent'.",
  "htmlExtra": null
}
```


If we look through the JSON file, we'll see a **key** called `word`, and two others called `definitions` and `examples` each with a sub-tree below it hierarchically called `0`. Within the `definitions` key we see the two **key value** pairs that we want: `text` that has the contents of the word's definition, `"An indifferent matter"` and `"partOfSpeech"` with the contents of `"noun"`. In the `examples` key we want `text` which has the contents of `"For if the condition is good, the adiaphoron, too, is good, and its observance is commanded."`

The raw JSON for these **key : value** pairs look like this:

```
"word": "adiaphoron""text": "An indifferent matter""partOfSpeech": "noun""text":
"For if the condition is good, the adiaphoron, too, is good, and its observance is
commanded."
```

Our CircuitPython code is able to grab and parse this data using these variables:

```
PART_OF_SPEECH = ['definitions', 0, 'partOfSpeech']
DEF_LOCATION = ['definitions', 0, 'text']
EXAMPLE_LOCATION = ['examples', 0, 'text']
```

*word data is called directly in the `while` loop

Parsing local JSON files

If you would like to avoid pulling data from a web page or maybe you don't feel like waiting for an API key, you can use a "local" JSON file to pull data from.

To implement this local data sourcing method, create a new file and name it `local.txt`. Populate this file with the JSON data that you would like to use. For example, you could use the JSON data provided above and make sure the format of the data is the same. Save this file on the CIRCUITPY drive in the root.

You should not need to change anything in your code.

And that's it! The JSON data will now be pulled from this local file!

PyPortal Constructor

When we set up the `pyportal` constructor, we are providing it with these things:

- `url` to query
- `json_path` to traverse and find the key:value pair we need
- `status_neopixel` pin
- `default_bg` path and name to display the background bitmap
- `text_font` path and name to the font used for displaying the follower count value
- `text_position` on the screen's x/y coordinate system
- `text_color`
- `caption_text` to display statically -- in this case the name of the repo
- `caption_font`
- `caption_position`

- `caption_color`

Fetch

With the pyportal set up, we can then use `pyportal.fetch()` to do the query and parsing of the three pieces of Wordnik data and then display them on screen along with the caption text on top of the background image.

Customization

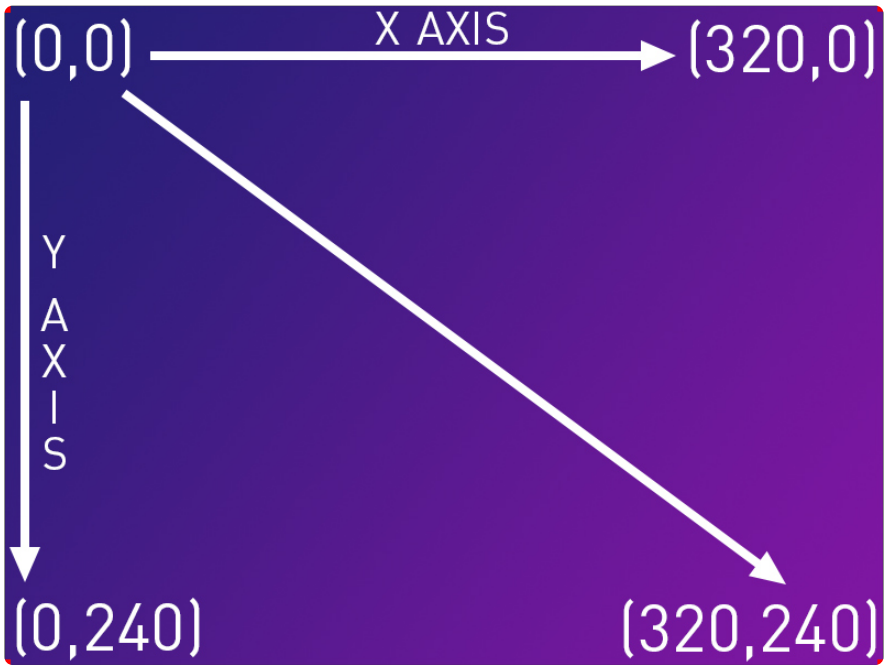
You can customize this project to make it your own and point to different website APIs as the source of your JSON data, as well as adjust the graphics and text.

Text Position

Depending on the design of your background bitmap and the length of the text you're displaying, you may want to reposition the text and caption. You can do this with the `text_position` and `caption_position` options.

The PyPortal's display is 320 pixels wide and 240 pixels high. In order to refer to those positions on the screen, we use an x/y coordinate system, where x is horizontal and y is vertical.

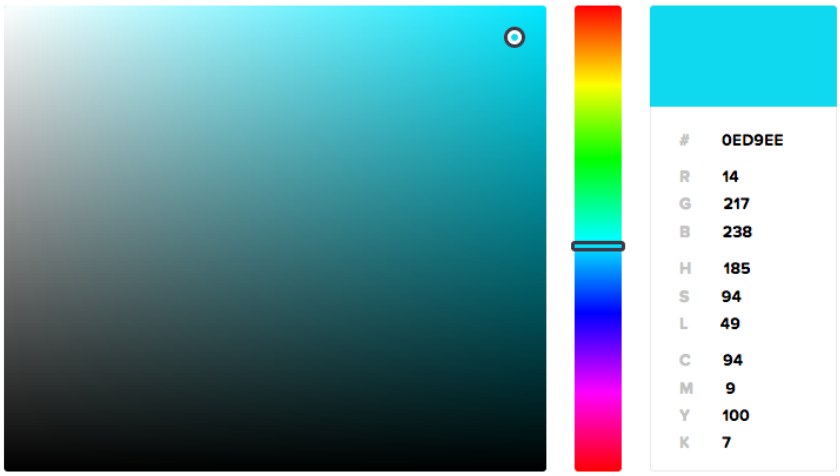
The origin of this coordinate system is the upper left corner. This means that a pixel placed at the upper left corner would be (0,0) and the lower right corner would be (320, 240).



Text Color

Another way to customize Word of the Day is to adjust the color of the text. The line `text_color=0xFFFFFF` in the constructor shows how. You will need to use the hexidecimal value for any color you want to display.

You can use something like https://htmlcolorcodes.com/ (https://adafru.it/Eb7) to pick your color and then copy the hex value, in this example it would be `0x0ED9EE`



Background Image

If you would like to create your own background, awesome! You'll want to save the file with these specifications:

- 320 x 240 pixels
- 16-bit RGB color
- Save file as .bmp format

You can then copy the .bmp file to the root level of the **CIRCUITPY** drive. Make sure you refer to this new filename in the pyportal constructor line:

```
default_bg=cwd+ "/wordoftheday_background.bmp"
```

Change that line to use the new filename name, such as:

```
default_bg=cwd+"/my_new_background.bmp"
```

Main loop

```
while True:
```

In the **repeating** main loop, the following actions are performed:

1. Check to see if the screen was **touched**
2. If screen was touched, display the word, part of speech, and definition with the source at the bottom.
3. If the screen was touched again, display the word, part of speech, and an example sentence with the word.
4. Any further screen touches will continue to switch between displaying the word's definition and example.

Going Further

How can the app be modified to give even more knowledge on the words?

The Wordnik API has [many more features](https://adafru.it/Ez2) (https://adafru.it/Ez2) including:

- Audio pronunciations
- Hyphenations
- Scrabble scores