Web: **https://www.aemet.es/es/serviciosclimaticos/datosclimatologicos/efemerides\_extremos**

**Resumen del flujo:**

1. Seleccionamos comunidad autónoma: (<select id="ccaa\_selector">)
2. **Pulsar el botón "Buscar"** (para que se cargue la página con las estaciones)
3. **Esperar a que se cargue el desplegable de estaciones** (<select id="estacion">)
4. Seleccionamos estación.
5. Pulsamos el primer botón "Buscar" del formulario frm1.
6. Se carga una nueva página con selección de:
   * Meses → seleccionamos "Anual" (value="13").
   * Variables → seleccionamos "todos".
7. Pulsamos el segundo botón "Buscar" (mismo <input type="submit">, pero en nueva página).
8. Esperamos la tabla, scrapeamos los datos, y extraemos valor y fecha.

driver.get("https://www.aemet.es/es/serviciosclimaticos/datosclimatologicos/efemerides\_extremos")

**Explicacion uso Selenium**

**driver = webdriver.Chrome()  
  
url = "https://www.aemet.es/es/serviciosclimaticos/datosclimatologicos/efemerides\_extremos"  
driver.get(url)**Inicia el navegador Chrome y abre la URL principal del servicio de efemérides de AEMET.

**wait = WebDriverWait(driver, 10)**Crea un 'esperador explícito' con un tiempo máximo de 10 segundos. Se usa para esperar a que aparezca un elemento en el DOM.

**wait.until(EC.presence\_of\_element\_located((By.ID, "ccaa\_selector")))**Espera hasta que el elemento con id='ccaa\_selector' esté presente en el DOM. Este elemento es el desplegable de comunidades autónomas.

**selector\_comunidades = Select(driver.find\_element(By.ID, "ccaa\_selector"))**Localiza el elemento <select> y lo convierte en un objeto Select para acceder a sus opciones fácilmente: **Es el desplegable**

**num\_comunidades = len(selector\_comunidades.options)**Cuenta cuántas opciones hay en el desplegable. Hay una vacía y luego todas las CCAA.

**datos = []**Inicializa una lista vacía para almacenar los datos que se extraerán más adelante.

**for i in range(1, num\_comunidades):**Itera sobre todas las comunidades, comenzando desde 1 para saltar la opción vacía.

**comunidad = selector\_comunidades.options[i]**Obtiene el objeto <option> correspondiente a la comunidad actual.

**cod\_ccaa = comunidad.get\_attribute("value")**Extrae el valor del atributo 'value', necesario para seleccionar la comunidad.

**nombre\_ccaa = comunidad.text.strip()**Obtiene el texto visible de la opción, por ejemplo 'Madrid' para guardarlo en la lista datos[]

**selector\_comunidades.select\_by\_value(cod\_ccaa)  
Selecciona la comunidad autónoma actual en el desplegable.**

**formulario = driver.find\_element(By.NAME, "frm1")**Localiza el formulario principal en el DOM.

**boton\_buscar\_comunidad = formulario.find\_element(By.CSS\_SELECTOR, "input[type='submit'].form\_submit")  
Encuentra el botón 'Buscar'** dentro del formulario mediante su clase y tipo.

**boton\_buscar\_comunidad.click()  
Simula un click** en el botón para enviar el formulario y cargar las estaciones de la comunidad seleccionada.

Se carga la página de selección de estaciones

**wait.until(EC.presence\_of\_element\_located((By.ID, "estacion")))**

Espera hasta que el elemento con id=**'estación'**  esté presente en el DOM. Este elemento es el desplegable de las estaciones

**selector\_estaciones = Select(driver.find\_element(By.ID, "estacion"))**

Localiza el elemento <select> y lo convierte en un objeto Select para acceder a sus opciones fácilmente: **Es el desplegable**

**num\_estaciones = len(selector\_estaciones.options) # numero de estaciones**

Cuenta cuántas opciones hay en el desplegable. Hay una vacía y luego todas las estaciones.

Repetimos lo mismo que hicimos para las comunidades, pero para las estaciones, al final de cada iteración, volvemos a la página de selección de estaciones:

**driver.get(url)  
wait.until(EC.presence\_of\_element\_located((By.ID, "ccaa\_selector")))  
selector\_comunidades = Select(driver.find\_element(By.ID, "ccaa\_selector"))  
selector\_comunidades.select\_by\_value(cod\_ccaa)  
formulario = driver.find\_element(By.NAME, "frm1")  
formulario.find\_element(By.CSS\_SELECTOR, "input[type='submit'].form\_submit").click()**

Ya hemos vuelto a cargar la pagina de las estaciones con el mismo código de comunidad autónoma **wait.until(EC.presence\_of\_element\_located((By.ID, "estacion")))  
selector\_estaciones = Select(driver.find\_element(By.ID, "estacion"))**

Cargamos el selector de estaciones porque vamos a incrementar en la iteración a la siguiente estación

Una vez termiando el bucle de las estaciones, volvemos a la página de las comunidades autónomas y obtenemos el selector para seguir avanzando en las iteraciones

**driver.get(url)  
wait.until(EC.presence\_of\_element\_located((By.ID, "ccaa\_selector")))  
selector\_comunidades = Select(driver.find\_element(By.ID, "ccaa\_selector"))**

**Explicación uso beautifulsoup**

Para analizar los datos de la tabla con beautifulsoup, no todos los campos tienen el mismo formato:

# Extraer tabla con BeautifulSoup  
soup = BeautifulSoup(driver.page\_source, "html.parser")  
tabla = soup.find("table", class\_="tabla\_datos\_sinpadd")  
filas = tabla.find\_all("tr")

Buscamos la tabla que tiene los datos climáticos (después de haber inspeccionado la página con F12)

tabla = soup.find("table", class\_="tabla\_datos\_sinpadd")

Esto busca el elemento <table> que contiene los datos, gracias a su class="tabla\_datos\_sinpadd".

filas = tabla.find\_all("tr")

Esto busca **todas las filas** <tr> (table row) — tanto del <thead> como del <tbody>.  
Por eso luego usamos filas[1:] para **saltar la cabecera** (que tiene los <th>).

for fila in filas[1:]:  
 columnas = fila.find\_all("td")

En cada fila real de datos, usamos .find\_all("td") para obtener sus **dos columnas**:

* td[0] → nombre de la variable
* td[1] → valor con fecha
* if len(columnas) == 2:  
   variable = columnas[0].get\_text(strip=True)  
   valor\_bruto = columnas[1].get\_text(strip=True)

Guardamos el valor de la variable y el contenido que vamos a analizar con expresiones regulares

|  |  |
| --- | --- |
| Máx. núm. de días de lluvia en el mes | 21 (abr. 1946) |
| Máx. núm. de días de nieve en el mes | 2 (dic. 1949) |
| Máx. núm. de días de tormenta en el mes | 7 (may. 1962) |
| Prec. máx. en un día (l/m2) | 98.3 (11 sept. 1951) |
| Prec. mensual más alta (l/m2) | 193.5 (dic. 1943) |
| Prec. mensual más baja (l/m2) | 0.0 (dic. 1979) |
| Racha máx. viento: velocidad y dirección (km/h) | Vel 119, Dir 230 (18 mar. 1971 12:11) |
| Tem. máx. absoluta (°C) | 39.6 (10 jul. 1979) |
| Tem. media de las máx. más alta (°C) | 30.9 (ago. 1949) |
| Tem. media de las mín. más baja (°C) | 5.7 (feb. 1956) |
| Tem. media más alta (°C) | 27.0 (ago. 1975) |
| Tem. media más baja (°C) | 9.5 (feb. 1956) |
| Tem. mín. absoluta (°C) | 0.2 (09 feb. 1935) |

**Variables de la tabla:**  
Máx. núm. de días de lluvia en el mes: Valor, mes y año

Máx. núm. de días de nieve en el mes: Valor, mes y año

Máx. núm. de días de tormenta en el mes: Valor, mes y año

Prec. máx. en un día (l/m2): Valor, dia, mes , año

Prec. mensual más alta (l/m2):Valor, mes año

Prec. mensual más baja (l/m2): Valor, mes año

Racha máx. viento: velocidad y dirección (km/h): valor velocidad, valor direccion, dia, mes, año, hora

Tem. máx. absoluta (°C): valor, dia, mes, año

Tem. media de las máx. más alta (°C): Valor, mes, año

Tem. media de las mín. más baja (°C): Valor, mes, año

Tem. media más alta (°C): Valor, mes, año

Tem. media más baja (°C): valor, mes, año

Tem. mín. absoluta (°C): valor, dia, mes, año   
  
Codificación tablas vientos.  
<https://www.aemet.es/documentos/es/conocermas/recursos_en_linea/publicaciones_y_estudios/publicaciones/Manual_de_uso_de_terminos_met_2023/Manual_de_uso_de_terminos_met_2023.pdf>  
https://www.aemet.es/documentos/es/conocermas/recursos\_en\_linea/publicaciones\_y\_estudios/publicaciones/Manual\_de\_uso\_de\_terminos\_met\_2023/Manual\_de\_uso\_de\_terminos\_met\_2023.pdf

**Explicacion de las expresione regulares:**  
**Para datos de viento Formato en tabla: Vel 126, Dir 230 (30 dic. 1981 14:15)**

Nos interesa quedarnos con varias variables:

* Velocidad (km/h),
* Dirección en grados que pasaremos a los 8 puntos de la rosa de los vientos según las tablas de la aemet
* Mes y Año. El día y la hora no nos interesarán.

match = re.search(r"Vel\s+(\d+),\s+Dir\s+(\d+)\s+\(\d{1,2}\s+(\w+)[.)]?\s+(\d{4})", valor\_bruto)

| **Parte del patrón** | **¿Qué hace?** | **Ejemplo que coincide** | **¿Se captura?** |
| --- | --- | --- | --- |
| Vel\s+ | Coincide literalmente con "Vel " seguido de uno o más espacios | "Vel " | ❌ No |
| (\d+) | Captura uno o más dígitos → la velocidad del viento | 153 | ✅ Sí (grupo 1) |
| ,\s+ | Coincide con coma y espacios: ", Dir " | , | ❌ No |
| Dir\s+ | Coincide con "Dir " | "Dir " | ❌ No |
| (\d+) | Captura la dirección del viento en grados | 230 | ✅ Sí (grupo 2) |
| \s+\( | Coincide con espacios y paréntesis de apertura | ( | ❌ No |
| \d{1,2} | Detecta el día, pero no lo guarda | 13 | ❌ No |
| \s+(\w+) | Captura el mes, una palabra como "ene" | ene | ✅ Sí (grupo 3) |
| [.)]? | Acepta un punto o paréntesis de cierre, opcional | . o ) o nada | ❌ No |
| \s+(\d{4}) | Captura el año, 4 dígitos | 1969 | ✅ Sí (grupo 4) |

vel, dir\_grados, mes, anio = match.groups()  
orientacion = **grados\_a\_orientacion**(dir\_grados) 🡪 Definidos al principio del código

**Para datos NO VIENTO:**  
match = re.search(r"([\-\d.]+)\s\*\((?:\d{1,2}\s+)?(\w+)[.)]?\s+(\d{4})", valor\_bruto)

**Los posibles formatos en tabla son:**

* en algunos casos el **valor será**: 78.1 (19 oct. 2023)
* **En otros:** 30.1 (jul. 2022)

Sólo almacenamos el mes y año, no nos interesa el día, por eso en la re. Introducimos: (?:\d{1,2}\s+)?

* (?:...) → Es un **grupo sin captura**, es decir, lo usa el patrón pero **no lo guarda**.
* \d{1,2} → Busca un número de 1 o 2 dígitos (el **día**).
* \s+ → Uno o más espacios después del día.
* ? → **Todo ese bloque es opcional**, es decir: puede estar o no.

La expresión completa sera:  
*([\-\d.]+)\s\*\((?:\d{1,2}\s+)?(\w+)[.)]?\s+(\d{4})*

1. **([\-\d.]+)** → Captura el valor (positivo o negativo, con decimales)
2. \**s\*\(** → Ignora los espacios antes del paréntesis de apertura
3. **(?:\d{1,2}\s+)?** → **Opcional**: día seguido de espacio
4. **(\w+)** → Captura el mes
5. **[.)]?** → Permite que el mes termine con punto, o no
6. **\s+(\d{4})** → Captura el año (cuatro cifras)

valor, mes, anio = match.groups()

| **Parte del patrón** | **¿Qué hace?** | **Ejemplo que coincide** | **¿Se captura?** |
| --- | --- | --- | --- |
| ([\-\d.]+) | Captura un número que puede tener signo negativo y decimales | 30.1, -7.2 | ✅ Sí (grupo 1) |
| \s\* | Permite cero o más espacios antes del paréntesis | *(espacios opcionales)* | ❌ No |
| \( | Coincide con un paréntesis de apertura literal | ( | ❌ No |
| (?:\d{1,2}\s+)? | Grupo **no capturable**: uno o dos dígitos (el **día**) seguido de espacio, **opcional** | 19 | ❌ No |
| (\w+) | Captura una palabra → el **mes** abreviado | jul, oct | ✅ Sí (grupo 2) |
| [.)]? | Coincide con un punto o paréntesis de cierre, **opcional** | ., ) | ❌ No |
| \s+ | Uno o más espacios antes del año | *(espacio)* | ❌ No |
| (\d{4}) | Captura cuatro dígitos → el **año** | 2022, 2023 | ✅ Sí (grupo 3) |

Ejemplos.  
Con estos inputs:

| **Input** | **Resultado de match.groups()** |
| --- | --- |
| "30.1 (jul. 2022)" | ('30.1', 'jul', '2022') |
| "78.1 (19 oct. 2023)" | ('78.1', 'oct', '2023') |
| "-5.2 (ene. 1985)" | ('-5.2', 'ene', '1985') |

**Guardamos los datos en un csv**

# Guardar en CSV  
**df = pd.DataFrame(datos)  
df.to\_csv("efemerides\_aemet\_completo.csv", index=False, encoding="utf-8-sig")  
print("\n✅ Proceso finalizado. Archivo guardado como 'efemerides\_aemet\_completo.csv'.")**

No incluimos el índice del data frame.