



Jul. 2022

Unidad 4:



Programación Avanzada

Objetivos de aprendizaje

Ser capaces de:

- Describir y crear módulos de Python
- Describir y crear paquetes de Python
- Trabajar con los módulos incorporados de Python
- Describir el objeto archivo de Python
- Leer y escribir en archivos de Python
- Trabajar con datos estructurados en archivos de Python

Módulos, paquetes y operaciones con archivos

Sección 1

Introducción

- El concepto de organizar el trabajo en archivos y carpetas también se aplica al programar en Python.
- Se puede organizar el código en piezas llamadas módulos, lo que facilita la agrupación de funcionalidades relacionadas.
- Una vez que has creado un módulo, es muy fácil volver a referirse a esa colección de funcionalidades y también compartir y reutilizar la funcionalidad definida en ese módulo.

Definición de módulo

- Es cualquier archivo .py que contenga código Python válido
 - Código como: definiciones de variables, funciones, clases, métodos, etc.

Importaciones

- Hay varias formas de importar y utilizar los recursos definidos en un módulo.
- Una forma de hacerlo
 - usando la palabra clave import
 - llamando a un recurso dentro de él usando la notación de punto (.)

```
>>> import modulo
>>> modulo.funcion(parametros)
```
- Otra forma es utilizar la sintaxis
 - from...import...:

```
>>> from modulo import *
>>> funcion(parametros)
```

Importaciones

- Hay varias formas de importar y utilizar los recursos definidos en un módulo.
- Una forma de hacerlo
 - usando la palabra clave import
 - llamando a un recurso dentro de él usando la notación de punto (.)

```
>>> import modulo
```

```
>>> modulo.funcion(parametros)
```

- Otra forma es utilizar la sintaxis

- from...import...:

```
>>> from modulo import *
```

```
>>> funcion(parametros)
```



Acceder a los recursos de esta manera es una mala práctica

Importaciones

- Nombrar las importaciones utilizando la palabra clave “as”

```
>>> from modulo import recurso as alias
```

```
>>> alias(parametros)
```

E1: Creando Módulos

- En este ejercicio, crearemos un módulo llamado calculadora:
 1. Crea un archivo llamado “calculadora.py”.
Aquí es donde van los recursos del módulo.
 2. Dentro del archivo, añade una función para sumar y devolver la suma de dos números
 3. A continuación, utilizando el comando Python en tu terminal en la misma carpeta en la que creaste el módulo de la calculadora, ejecutar:

```
>>> import calculadora  
>>> calculadora.sumar(8, 9)
```

Módulos y paquetes

La ruta de búsqueda de módulos

Cuando importas un módulo

- Python comprueba si hay un módulo incorporado con el nombre especificado.
- Si no encuentra ningún módulo incorporado
el intérprete buscará un archivo con el nombre del módulo y la extensión .py
en los directorios dados por la variable sys.path.

Inspeccionar el sys.path en su entorno actual

```
>>> import sys
```

```
>>> sys.path      # -> ['', '/usr/local/bin',
```

Módulos Standard de Python

- Python cuenta con muchos módulos y paquetes ya hechos.
- Estos módulos y paquetes se agrupan en bibliotecas

Nombre del módulo	Función
string	Contiene funciones para generar y operar con strings
math	Tienen funciones matematicas
unittest	Ayudar en la escritura de prueba unitarias
sys	Acceso al sistema
urllib	Trabajar con URLs y HTTP
datetime	Trabajar con fechas, horas y zonas horarias
random	Un modulo para generar numeros pseudo random
re	Para trabajar con expresiones regulares
itertools	Iteracion eficiente sobre colecciones como listas
functools	Para escribir decoradores

Módulos Standard de Python

- Python cuenta con muchos módulos y paquetes ya hechos.
- Estos módulos y paquetes se agrupan en bibliotecas
- Puedes importar cada módulo

- y utilizar la función `dir()` para ver una lista de los métodos implementados en cada uno

```
>>> import math
>>> dir(math)
[... 'pi', 'pow', 'prod',
'radians', ...']
```

Nombre del módulo	Función
string	Contiene funciones para generar y operar con strings
math	Tienen funciones matematicas
unittest	Ayudar en la escritura de prueba unitarias
sys	Acceso al sistema
urllib	Trabajar con URLs y HTTP
datetime	Trabajar con fechas, horas y zonas horarias
random	Un modulo para generar numeros pseudo random
re	Para trabajar con expresiones regulares
itertools	Iteracion eficiente sobre colecciones como listas
funtools	Para escribir decoradores

Actividad 1: Inspección de módulos

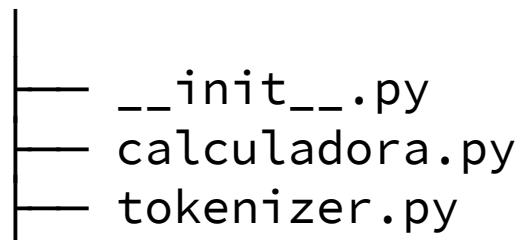
Usar las funciones incorporadas de Python para inspeccionar módulos.

1. Abre el intérprete de Python
2. Importar la biblioteca itertools.
3. Utiliza la función incorporada correspondiente para inspeccionar los recursos definidos en ella.
4. Utiliza la función incorporada correspondiente para leer cómo utilizar la biblioteca itertools.

Paquetes

- Los paquetes son colecciones de módulos.
- Cada paquete debe contener un `__init__.py` en la raíz de su directorio,
 - indica al intérprete que el directorio es un módulo.
 - el archivo no necesita contener nada
 - para construir el paquete y el espacio de nombres del paquete.
- Estructura:

`misherramientas`



Importaciones absolutas y relativas

- **importación absoluta** utiliza la ruta completa desde un proyecto hasta el módulo que se importa.
 - `import string`
- **importación relativa** utiliza la ruta relativa desde el directorio actual hasta el directorio donde se encuentra el módulo
 - `from . import calculadora`
- Siempre que sea posible, prefiera las importaciones absolutas

Actividad 2: Listado de los recursos definidos en un paquete o módulo

Escribir un script para listar todos los recursos que están disponibles en ese paquete.

1. Defina una función llamada `paquete_enumerador`.
2. Dentro de la función, utilice una sentencia `for` para listar los recursos del paquete.
3. Liste los recursos utilizando la función que acaba de crear

Actividad 3: Uso de recursos en un módulo

Escribe una función llamada `generador_numeros_aleatorios` cuyo objetivo sea generar enteros aleatorios entre 0 y 1.000

- La función debe utilizar los recursos definidos en el módulo incorporado “random”.
- Debe tomar un argumento, que es un entero “n”
- Debe devolver una lista que contenga “n” números aleatorios.
- Por ejemplo, llamando a la función con 5 se obtiene una lista de tres números aleatorios:

```
>>> generador_numeros_aleatorios(5)    #-> [539, 166, 548, 446, 238]
```

Actividad 3: Uso de recursos en un módulo

Pasos

1. Importar el módulo random.
2. Define una función llamada “generador_numeros_aleatorios”
3. Dentro de la función utilizar
 - Un bucle que use los recursos del módulo importado para generar números aleatorios.

Operaciones con archivos

- Los archivos pueden ser de muchos tipos
 - texto con extensiones .txt
 - datos organizados con extensión .csv, .json
 - ejecutables con extensiones .exe
- Trabajaremos con extensiones no propietarias
 - como los archivos .txt, .csv y .json.
- Las habilidades aprendidas serán transferibles a cualquier otro archivo que necesite ser leído con Python.

El objeto archivo

- El objeto file es la forma predeterminada y más fácil de manipular archivos en Python. Incluye un par de métodos y atributos que facilitan a los desarrolladores la lectura y escritura de archivos en el sistema de archivos.
- En Python se reconocen dos tipos principales de objetos de archivo:
 - binarios: pueden leer y escribir objetos tipo **byte**.
 - texto: pueden leer y escribir objetos de tipo **cadena de caracteres**.
- La forma más sencilla de crear un objeto archivo
 - la función open() pasandole el modo para obtener de vuelta un objeto de archivo binario o de texto

Los métodos del objeto archivo

- `file.read()`: carga el archivo completo en memoria.
- `file.readline()`: lee una sola línea del archivo en la memoria.
- `file.readlines()`: lee todas las líneas de un archivo en una lista.
- `file.write()`: escribe la salida en el archivo.
- `file.seek()`: mover la posición del objeto archivo a una determinada ubicación en el archivo.

Lectura y escritura en archivos

- El método `open()`
 - Función incorporada en Python para leer un archivo.
 - Toma dos argumentos, el nombre del archivo y un modo
 - Devuelve un objeto archivo al que se puede manipular
- El modo que se pasa a la función determina qué tipo de objeto de archivo obtendrá y qué podrá hacer con él.

Lectura y escritura en archivos

Modos disponibles:

- r Modo lectura
- w Modo escritura
- a Modo de añadir
- r+ Maneja ambos lectura y escritura

E3: Crear, leer y escribir en archivos

1. Usa el método `open()` para crear un archivo:

```
f = open('miarchivo.txt', 'w')
```

2. Escriba algo en el archivo

```
>>> f.write("Hola, mundo\n")
```

3. Mirar el contenido del archivo

4. Vuelve a escribir algo más:

```
>>> f.write("Hola, mundo otra vez")
```

5. Cierra el archivo.

```
>>> f.close()
```

Con el gestor de contexto

- Para trabajar con archivos Python cuenta con un práctico gestor de contexto
 - Ayuda a escribir un código más ordenado y a liberar recursos automáticamente
 - Cuando el código sale para que tú no tengas que hacerlo.
- La sintaxis es la siguiente:

```
>>> with open("miarchivo.txt", "r+") as f:  
...     contenido=f.read()  
...     print(contenido)
```

Actividad 4: Realizar operaciones con archivos

- Crearemos un archivo y le añadiremos datos
 1. Crear un objeto archivo, utiliza el gestor de contexto with o la función open()
 2. Escribe en el archivo "I love Python"

Si usas open() recuerda cerrar el objeto archivo.

Manejando Datos Estructurados

- En las aplicaciones del mundo real, lo más probable es que tengas que leer datos en un formato estructurado.
- Esos datos podrían representarse en un archivo CSV.
- Un archivo CSV es un archivo con valores separados por comas, normalmente organizados en columnas.

Estos datos pueden leerse fácilmente en una hoja de cálculo como Excel, y manipularse allí.

- Python proporciona una utilidad para trabajar con archivos CSV.

Manejando Datos Estructurados

- Los archivos CSV son populares porque tienen algunas ventajas sobre archivos como las hojas de cálculo
 - Son legibles y fáciles de analizar.
 - Son más pequeños, compactos y rápidos de trabajar.
 - Los archivos CSV son fáciles de generar y tienen un formato estándar.
- Otra forma de datos estructurados es JSON.
- JSON es muy útil, especialmente en Internet, para transferir fácilmente datos en forma de pares clave-valor.

Manejando Datos Estructurados

- Los datos JSON tienen este aspecto:

```
{
  "datos": [
    {
      "nombre": "Rodríguez, Eva",
      "ciudad": "Londres"
    },
    {
      "nombre": "Civera, Leandro",
      "ciudad": "Siena"
    }
  ]
}
```

- Otros formatos utilizados para transferir datos por Internet son:
XML (www.w3schools.com/xml/)
Buffer de protocolo (developers.google.com/protocol-buffers/)

Trabajar con datos CSV

- El módulo “csv” nos ayuda a trabajar con archivos CSV.
 - Hace que sea muy fácil leer los datos tabulares
 - Operar con los datos creando objetos lectores o escritores.
 - `import csv`

Lectura de un archivo CSV

```
import csv

with open('MOCK_DATA.csv', 'r') as f:
    lector_datos = csv.reader(f)
    contador_linea = 1

    for fila in lector_datos:

        if contador_linea > 1: # Saltamos la linea 1 que son los
                               # titulos de las columnas
            print(fila)

        contador_linea += 1
```


Escritura en un archivo CSV

```
import csv

with open('example.csv', 'w', newline='') as f:
    escritor_datos = csv.writer(f)
    escritor_datos.writerow(['nombre', 'edad'])
    escritor_datos.writerow(['Luis', 21])
```

Escribir un dict en el archivo CSV

- En lugar de escribir filas utilizando listas, se puede escribir diccionarios.
- Utilizar un objeto DictWriter (en lugar del objeto writer)
 - sus comportamientos son bastante similares
- Una lista de nombres de campo especificará el orden en el que se escribirán los valores en el archivo.

Escribir un dict en el archivo CSV

```
import csv

with open("gente.csv", "r+", newline='') as f:
    campos=['nombre', 'edad']
    gente_escritor = csv.DictWriter(f, fieldnames=campos)
    gente_escritor.writeheader()
    gente_escritor.writerow({'nombre': 'Jorge', 'edad': 50})
```

Actividad 5: Trabajar con archivos

1. Escriba un script para leer un archivo CSV con el formato de la **tabla1**
2. Dar salida a un nuevo archivo CSV con formato similar al de la **tabla2**
3. Los sueldos se calculan utilizando la fórmula $\text{horas_trabajadas} * 15$

Tabla1

nombre	Horas trabajadas
James Miller	36
Teresia Brown	41
Mery Laney	40

Tabla2

nombre	sueldo
James Miller	€540
Teresia Brown	€615
Mery Laney	€600

Trabajar con datos JSON

- JSON son las siglas de JavaScript Object Notation.
- Formato de datos construido para intercambiar datos de manera legible para los humanos.
- Hoy en día la mayoría de las APIs con las que se interactúa en Internet utilizan JSON.
- Python incluye un módulo “json” con funciones para
 - ayudar a analizar JSON
 - convertir objetos de Python (ejemplo, diccionarios)
- Hay dos métodos críticos del módulo json que necesitas conocer para usar JSON en Python.
- Para la codificación de JSON se utiliza `json.dumps()`

Trabajar con datos JSON

```
import json
muestra={
    "nombre":"jorge",
    "edad":30
}
# codifica a JSON
ejemplo_json = json.dumps(muestra)
print(ejemplo_json)          # -> {"nombre": "jorge", "edad": 30}
print(type(ejemplo_json))    # -> <class 'str'>

# decodificar el objeto
ejemplo_original=json.loads(ejemplo_json)
print(ejemplo_original)      # -> {'nombre': 'jorge', 'edad': 30}
print(type(ejemplo_original)) # -> <class 'dict'>
```

Manejo de Datos

Sección 3

Objetivos de aprendizaje

Al final de esta sección serás capaz de:

- Leer páginas web y recopilar datos de ellas haciendo uso de requests y BeautifulSoup
- Realizar operaciones de lectura en archivos XML y en la web utilizando una API
- Utilizar técnicas regex para extraer información útil de un corpus de texto grande y desordenado
- Recopilar datos de páginas web, archivos XML y APIs

Introducción

- Una de las habilidades más valoradas y utilizadas es la capacidad de leer y extraer datos de páginas web y bases de datos alojadas en la web
- La mayoría de las organizaciones alojan datos en la nube y la mayoría de los microservicios web de proporcionan algún tipo de API para que usuarios externos puedan acceder a sus datos
- Es necesario conocer la estructura de las páginas web y las librerías de Python

La biblioteca Requests y BeautifulSoup

- **Requests** es una API construida sobre librerías de utilidades web puras de Python que permite hacer peticiones HTTP fácil e intuitivamente

```
import requests
```

- **BeautifulSoup** es uno de los paquetes de análisis de HTML más populares.
 - Analiza el contenido HTML
 - Construye un árbol detallado de todas las etiquetas y marcas

```
import BeautifulSoup
```

Web Scrapping

- Remover las capas de HTML/CSS/JavaScript para extraer la información que nos interesa, como un objeto que Python pueda procesar



Web Scrapping - Pasos en general

1. Importa la biblioteca `requests`
2. Utiliza el método `get` de requests para obtener respuesta de una URL
3. Comprobar el `estado` de la petición mediante código 200 de la respuesta
4. Comprobar la `codificación` de la página web
5. `Decodificar` el contenido del objeto de respuesta
6. `Tratar` la mezcla de etiquetas de marcado HTML, `texto` y nombres de elementos/propiedades

el método `text` de BeautifulSoup se puede para extraer texto


Web Scrapping - Pasos en general

1. Importar `requests`
2. Utiliza `get` para obtener respuesta de una URL `r=requests.get("URL")`
3. Comprobar el `estado` de la respuesta `if r.status_code==200`
4. Comprobar la `codificación` de la p. web `r.encoding`
5. `Decodificar` la respuesta `contenido=r.content.decode(r.encoding)`
6. `Tratar` el texto `sopa=BeautifulSoup(contenido, 'html.parser')`
`texto=sopa.text`

E1: Extraer texto desde una sección en una web

- En la página de inicio de Wikipedia hay una sección llamada “Recurso del día” que muestra un extracto de un artículo destacado de hoy, y es necesario extraer el texto de esta sección

Recurso del día



Castillo de Chambord

El **Castillo de Chambord** es el mayor de los **castillos del Loira**. Fue construido entre 1519 y 1539 como pabellón de caza para el rey **Francisco I de Francia** a partir del diseño original de **Domenico da Cortona**, aunque se cree que **Leonardo da Vinci** también participó en su diseño final.

Archivo

Portales

Artes: Arquitectura – Cine – Danza – Literatura – Música – Música clásica – Pintura – Teatro

Ciencias sociales: Comunicación – Deporte – Derecho – Economía – Filosofía – Lingüística – Psicología – Sociología

- 12 de julio: Día Nac
- 12 de julio: Día Nac
- 11 de julio: [Día Mun](#)
- 11 de julio: Día Naci

Véase también: [Categ](#)

Otros eventos actua

Efemérides

12 de julio

- **1922** (*hace 100 años* británico (f. 1956).
- **1962** (*hace 60 años* **Stones** actúan por l
- **1997** (*hace 25 años* francés (n. 1927).
- **1997** (*hace 25 años* Hayao Miyazaki.
- **1997** (*hace 25 años* la paz en 2014 (*en l*

11 de julio 12 de

1. Buscando palabras y utilizando índices

Inicio del texto
que estamos interesados



Recurso del día



Castillo de Chambord

El **Castillo de Chambord** es el mayor de los **castillos del Loira**. Fue construido entre 1519 y 1539 como pabellón de caza para el rey **Francisco I de Francia** a partir del diseño original de **Domenico da Cortona**, aunque se cree que **Leonardo da Vinci** también participó en su diseño final.

Archivo

Portales

Artes: Arquitectura – Cine – Danza – Literatura – Música – Música clásica – Pintura – Teatro

Ciencias sociales: Comunicación – Deporte – Derecho – Economía – Filosofía – Lingüística – Psicología – Sociología

- 12 de julio: Día Nac
- 12 de julio: Día Nac
- 11 de julio: [Día Mun](#)
- 11 de julio: Día Naci

Véase también: [Categori](#)

Otros eventos actua

Efemérides

12 de julio

- **1922** (*hace 100 años* británico (f. 1956).
- **1962** (*hace 60 años* **Stones** actúan por l
- **1997** (*hace 25 años* francés (n. 1927).
- **1997** (*hace 25 años* Hayao Miyazaki.
- **1997** (*hace 25 años* la paz en 2014 (*en l*

11 de julio 12 de

Fin del texto que estamos interesados

2. Uso de técnicas avanzadas BS4

Texto a extraer de aquí

como
ortona,

Efemérides

12 de julio

- **1922** (hace 100 años): Nace **Michael Ventris**, lingüista británico (f. 1956).
- **1962** (hace 60 años): En Londres (Reino Unido), **The Rolling Stones** actúan por primera vez.
- **1997** (hace 25 años): Fallece **François Furet**, historiador francés (n. 1927).
- **1997** (hace 25 años): Se estrena la película **La princesa Mononoke**, del japonés Hayao Miyazaki.
- **1997** (hace 25 años): Nace **Malala Yousafzai**, activista pakistaní, premio Nobel de la paz en 2014 (en la imagen).

11 de julio 12 de julio 13 de julio ...

Otros proyectos de la Fundación Wikimedia

 **Commons**
Imágenes y multimedia

 **Wikcionario**
Diccionario libre



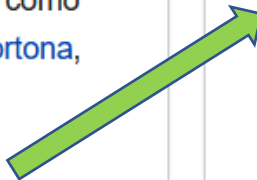
2. Uso de técnicas avanzadas BS4

Texto a extraer de aquí



Inicio del texto
que estamos interesados

como
ortona,



Efemérides

12 de julio

- **1922** (hace 100 años): Nace **Michael Ventris**, lingüista británico (f. 1956).
- **1962** (hace 60 años): En Londres (Reino Unido), **The Rolling Stones** actúan por primera vez.
- **1997** (hace 25 años): Fallece **François Furet**, historiador francés (n. 1927).
- **1997** (hace 25 años): Se estrena la película **La princesa Mononoke**, del japonés Hayao Miyazaki.
- **1997** (hace 25 años): Nace **Malala Yousafzai**, activista pakistaní, premio Nobel de la paz en 2014 (en la imagen).



11 de julio

12 de julio

13 de julio

...

Fin del texto que estamos
interesados



Otros proyectos de la Fundación Wikimedia



Commons

Imágenes y multimedia



Wikcionario

Diccionario libre

2. Uso de técnicas avanzadas BS4

Queremos empezar aquí inmediatamente después de la fecha

12 de julio

1922 (hace 100 años): Nace Michael Ventris, lingüista británico (f. 1956).

1962 (hace 60 años): En Londres (Reino Unido), The Rolling Stones actúan por primera vez.

1997 (hace 25 años): Fallece François Furet, historiador francés (n. 1927).

1997 (hace 25 años): Se estrena la película La princesa Mononoke, del japonés Hayao Miyazaki.

1997 (hace 25 años): Nace Malala Yousafzai, activista pakistaní, premio Nobel de la paz en 2014 (en la imagen).

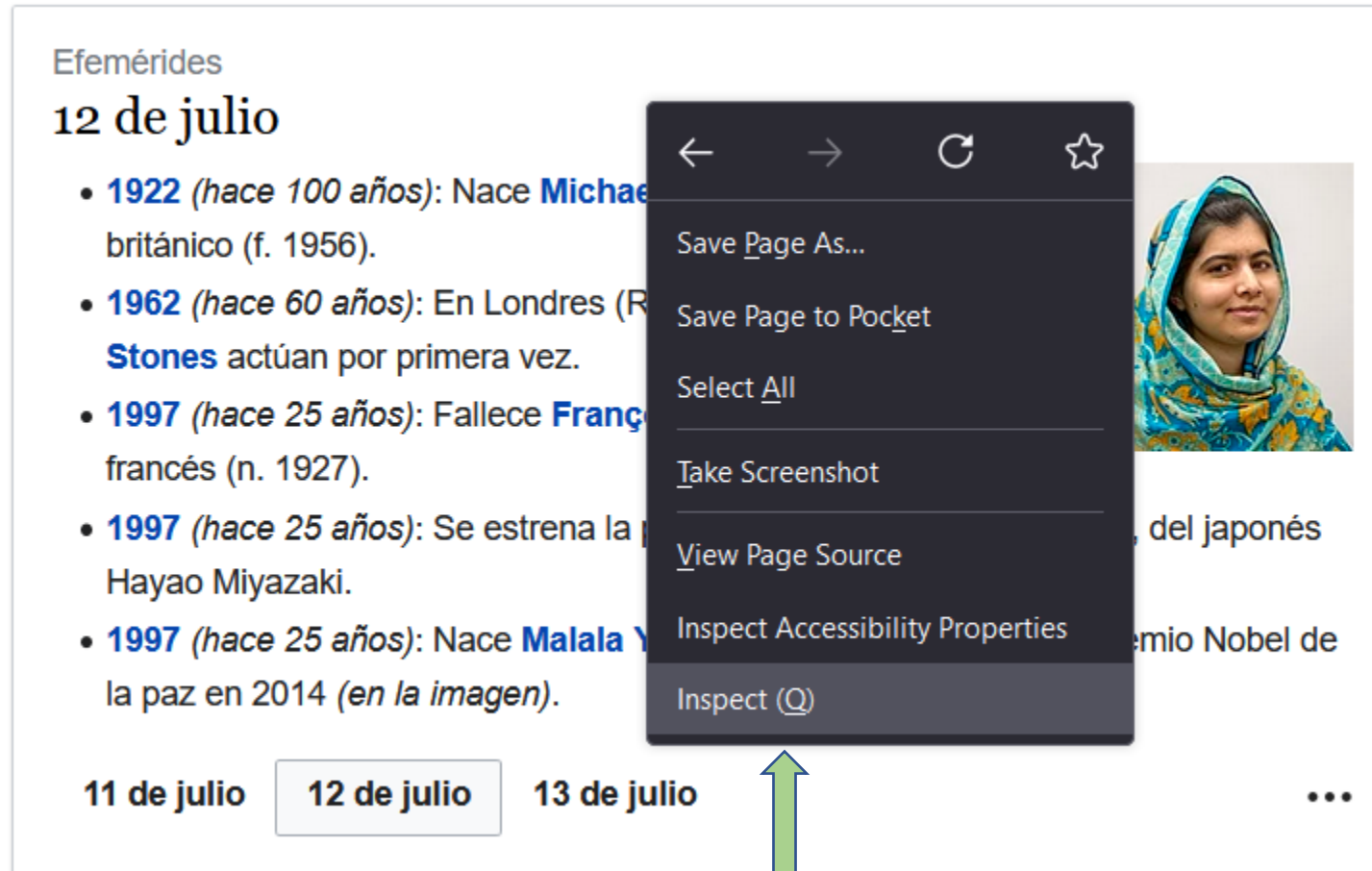
11 de julio

12 de julio

13 de julio

No hay forma de detenernos exactamente

2. Uso de técnicas avanzadas BS4



Clic derecho sobre el elemento y escoger la opción “Inspect”

2. Uso de técnicas avanzadas BS4

Este es el bloque que contiene el texto “`main-idt`”



2. Uso de técnicas avanzadas BS4

```
▼ <div id="main-itd" class="main-box">  
  <div id="Efemérides" class="main-box-section">Efemérides</div>  
  ▶ <h2 class="main-header main-box-header ext-discussiontools-init-section">... </h2>  
  ▶ <div class="floatright">... </div>  
  ▼ <ul>  
    ▶ <li>... </li>  
    ▶ <li>... </li>  
    ▶ <li>... </li>  
    ▶ <li>... </li>  
    ▶ <li>... </li>  
  </ul>  
  <div style="clear: both;"></div>  
  ▶ <div class="main-footer">... </div> flex  
</div>
```

← Este es el bloque **<div>** que contiene los ****
Tiene el id **"main-itd"**

← Este es el bloque **** que contiene el texto que buscamos

XML (Lenguaje de Marcado Extensible)

- A diferencia de otros lenguajes de marcado
 - Permite definir tus propias etiquetas

```
<mensaje>  
  <saludo>  
    Hola, mundo  
  </saludo>  
</mensaje>
```

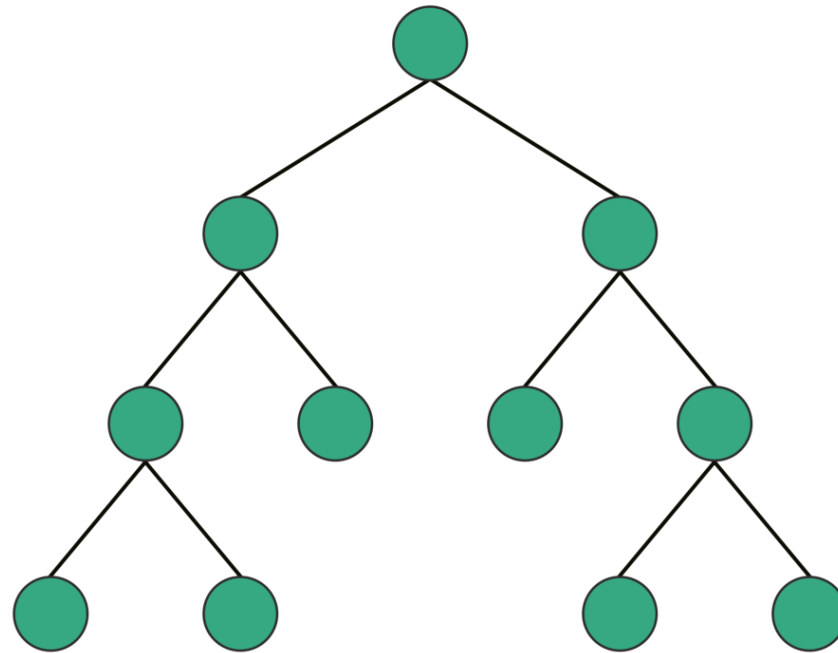
```
<usuario></usuario>  
<usuario id='6'></usuario>  
<usuario id='6'>Sistema</usuario>
```

XML (Lenguaje de Marcado Extensible)

Árbol de elementos

1 Padre

0+ Hijos





Programación F(uncional)

Sección 3

Contenido

- ¿Qué es la programación funcional y por qué interesa?
- `lambda`
- `map` y `filter`
- Iteradores/Generadores
- Decoradores

Funciones de clase

```
def funcion(arg):  
    return arg
```

```
type(funcion)      # => <class 'function'>  
id(funcion)        # => 2151832346688  
print(funcion)     # => <function funcion at 0x1F50333E040>  
  
fun = funcion  
id(fun)            # => 2151832346688  
print(fun)         # => <function funcion at 0x1F50333E040>  
isinstance(echo, object)      # => True
```

Paradigmas de programación

- **Procedimental:** el programa es una secuencia de instrucciones que indican al ordenador lo que debe hacer.
 - Ej.: C, Pascal, Unix shell.
- **Orientada a Objetos:** colecciones de objetos con un atributos distintivos que se pueden acceder y a menudo modificar por medio de diferentes métodos.
 - Ej.: Java, Smalltalk.
- **Declarativo:** describe el problema que hay que resolver y la implementación del lenguaje se encarga de los detalles.
 - Ej.: SQL, Prolog.
- **Funcional:** los programas se descomponen en conjuntos de funciones, cada una de las cuales toma entradas y produce salidas sin estados internos.
 - Ej.: Haskell, OCaml.

Why Functional Programming Matters

John Hughes
The University, Glasgow

Abstract

As software becomes more and more complex, it is more and more important to structure it well. Well-structured software is easy to write and to debug, and provides a collection of modules that can be reused to reduce future programming costs. In this paper we show that two features of functional languages in particular, higher-order functions and lazy evaluation, can contribute significantly to modularity. As examples, we manipulate lists and trees, program several numerical algorithms, and implement the alpha-beta heuristic (an algorithm from Artificial Intelligence used in game-playing programs). We conclude that since modularity is the key to successful programming, functional programming offers important advantages for software development.

<https://www.cs.kent.ac.uk/people/staff/dat/miranda/whyfp90.pdf>

Programación funcional - Ejemplo

Procedimental - "flujo del programa"

```
def obtener_pares(arr):  
    ret_list = []  
    for elem in arr:  
        if elem % 2 == 1:  
            ret_list.append(elem)  
    return ret_list
```

Funcional - "los programas son conjuntos de funciones"

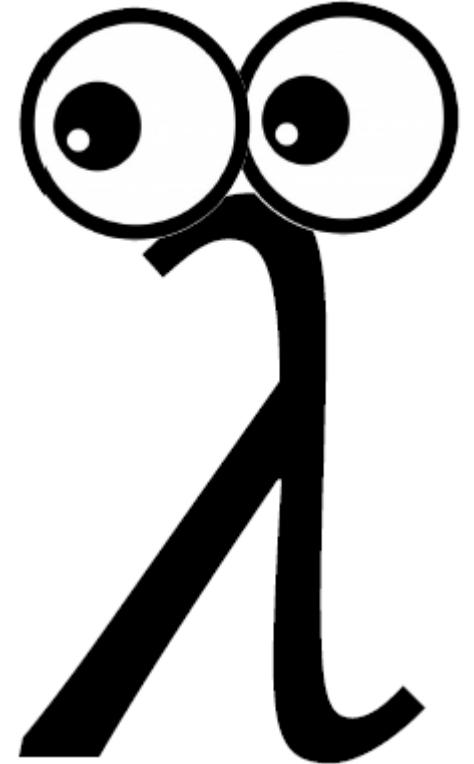
```
def obtener_pares(arr):  
    return list(filter(lambda elem: elem % 2 == 0, arr))
```

¿Por qué la programación funcional?

- **Simplificar la depuración** - línea por línea, por lo que es más fácil encontrar puntos de fallo.
 - Menos variables diseñadas exclusivamente
- **Menos código y más limpio** - ¡como en el ejemplo anterior!
- **Modular** - las funciones suelen ser reutilizables, por lo que es más rápido codificar, probar y depurar.

Funciones Lambdas λ

- ¡Funciones más pequeñas y bonitas!



Lambdas: Funciones anónimas en línea

- Funciones anónimas, sobre la marcha, que pueden ser pasadas como parámetros a otras funciones.

```
>>> lambda params: expression
```

Comprueba si el primer elemento de un par (tupla) es mayor que el segundo

```
>>> lambda tup: tup[0] > tup[1]
```

Las lambdas pueden personalizar el funcionamiento de las funciones.

Lambdas

De esta lista de pares

```
parejas=[(3,2),(-1,5),(4,4)]
```

Encontrar el máximo, por el valor del segundo elemento

```
max(parejas, key=lambda tup: tup[1])  
(-1, 5)
```

Ordenar la colección de pares por el valor del segundo elemento

```
sorted(parejas, key=lambda tup: tup[1])  
[(3, 2), (4, 4), (-1, 5)]
```

Lambdas

- El objetivo de una lambda es ser utilizada dentro de una llamada a una función, y luego ser descartada.
 - Si lo vinculamos a un nombre para ser llamado así mismo en el futuro
¡También podría definirse como una función!

```
trip=lambda x:3*x
```

```
trip          # <function <lambda> at 0x000002212F36E040>
```

```
trip(4)       # -> 12
```