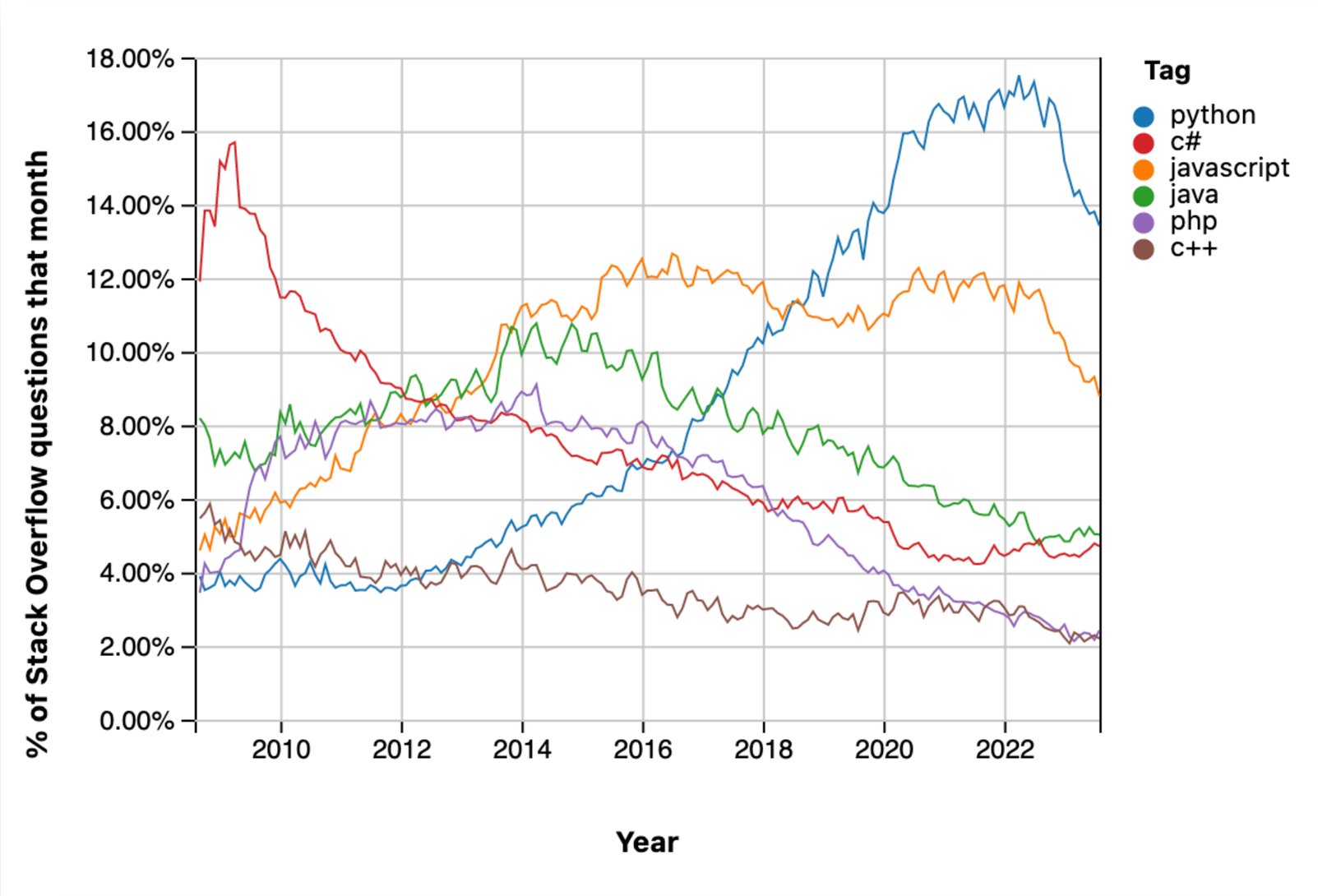
Python

Sistemas de gestión empresarial – 148FA (DAM2)

Características

- Lenguaje ~~compilado~~ interpretado
- Tipado dinámico: las variables se adaptan
- Multiplataforma
- Multiparadigma
  - Orientación a objetos
  - Programación imperativa
  - Programación funcional
- Libre

Uso de Python



Dónde se usa Python

- Aplicaciones de Inteligencia Artificial – Machine Learning, Deep Learning
- Big Data
- Aplicaciones Web: redes sociales (Instagram), almacenamiento en la nube (Dropbox)…
- Frameworks de pruebas
- Investigación
- Ciberseguridad

Entorno de desarrollo

Instalación

- Windows: descarga desde: https://www.python.org/
- MacOS:

```
brew install python3
```

Comprobación:

```
python --version
```

Identación

- En Python es muy importante la identación (las tabulaciones) para escribir el código correctamente
- Se trata de la sangría, como si estuviéramos hablando de un documento de Word, la cual va a identificar el fin de las sentencias if, los bucles, etc.
- En el caso de Python, la identación es obligatoria, ya que de ella, dependerá su estructura

```
x = 5
if x == 5:
    # tabulación de 4 espacios
    print("El valor de x es 5.")
```

Comentarios

- Comentarios de una sola línea:

```
# Es un comentario de una línea
```

- Comentarios de más de una línea

```
'''
Este es un comentario de más
de una línea
'''
```

Variables

- Variables:
  - Sin acabar con ;
  - No hay :
  - Tipado dinámico
  - Posibilidad de hacer casting
- CONSTANTES (No existen como tal): sólo se distinguen por la convención de nombre —> en mayúsculas

```
MI_VARIABLE_CONSTANTE = 12
```

Hello world

- Crea un fichero helloWorld.py y añade:

helloWorld.py

```
1  name=input("Enter your name: ")
2  print("Hello "+name)
```

- Ejecuta el script vía línea de comandos e IDE.

Tipos de variables

```
edad = 24 # número entero (integer)
precio = 112.9 # número de punto flotante (float)
titulo = ‘Aprende Python desde cero’ # cadena de texto (string)
test = True # booleano
```

Casting de tipos

casting.py

```
1  num1=int(input("Enter the first number: "))
2  num2=int(input("Enter the second number: "))
3  result=num1+num2;
4  print("This is the result",result)
```

Operadores aritméticos

Símbolo	Significado	Ejemplo	Resultado
+	Suma	a = 5 + 5	10
-	Resta	a = 9 - 7	2
-	Negación	a = -3	-3
*	Multiplicación	a = 4 * 5	20
**	Exponente	a = 2 ** 4	16
/	División	a = 12.5 / 2	6.25
//	División entera	a = 12.5 / 2	6.0
%	Módulo	a = 18 % 4	2

Operadores relacionales

Símbolo	Significado	Ejemplo	Resultado
!	Igual que	3 == 2	False
!=	Distinto que	coche != moto	True
<	Menor que	4 < 16	True
>	Mayor que	8 > 7	True
<=	Menor o igual que	2 <= 2	True
>=	Mayor o igual que	4 >= 5	False

Operadores lógicos

Operador	Ejemplo	Explicación	Resultado
and	3 == 7 and 7 < 15	False and False	False
and	9 < 12 and 12 > 7	True and True	True
and	9 < 12 and 12 > 15	True and False	False
or	12 == 12 or 15 < 7	True or False	True
or	7 > 5 or 9 < 12	True or True	True
xor	4 == 4 xor 9 > 3	True o True	False
xor	4 == 4 xor 9 < 3	True o False	True

Estructuras de control

- Condicionales:

```
numero = 5
if numero < 3:
    print("Es menor que 3")
elif numero < 6:
    print("El número está entre el 3 y el 5")
else:
    print("Es mayor o igual a 6")
```

Estructuras de control

- Bucle While:

```
contador = 0
while(contador < 5):
    # Se ejecutará mientras la variable contador sea menor a 5.
    contador = contador+1
    print("Iteración número",contador)
print ("¡Fin!")
```

Existen las sentencias break y continue.

Estructuras de control

- Bucle While con else:

```
count = 0
while(count < 5):
    count = count+1
    print("Iteración número {}".format(count))
else:
    print("Bucle while finalizado")
```

Estructuras de control

- Bucle for:

```
numeros = [4,8,2,7,1,9,3,5]
total = 0

for n in numeros:
    total += n
    if total > 10
        break
```

```
alumnos = ["Ane", "Mikel", "Unai", "Lorea"]
for alumno in alumnos:
    print(alumno)
```

```
alumnos = ["Ane", "Mikel", "Unai", "Lorea"]
for alumno in alumnos:
    print(alumno)
```

Estructuras de control

- Funcion range(): La función range([start,] stop [, step]) devuelve una secuencia de números. Es por ello que se utiliza de forma frecuente para iterar:

```
print("Números del 5 al 10")
for i in range(5,  10):
    print(i,  end=', ')
# 5,  6,  7,  8,  9,

print("Números impares del 1 al 10")
for i in range(1,  10,  2):
    print(i,  end=', ')
# 1,  3,  5,  7,  9,
```

```
alumnos = ["Ane", "Mikel", "Unai", "Lorea"]
for i in range(len(alumnos)):
        print(alumnos[i])
```