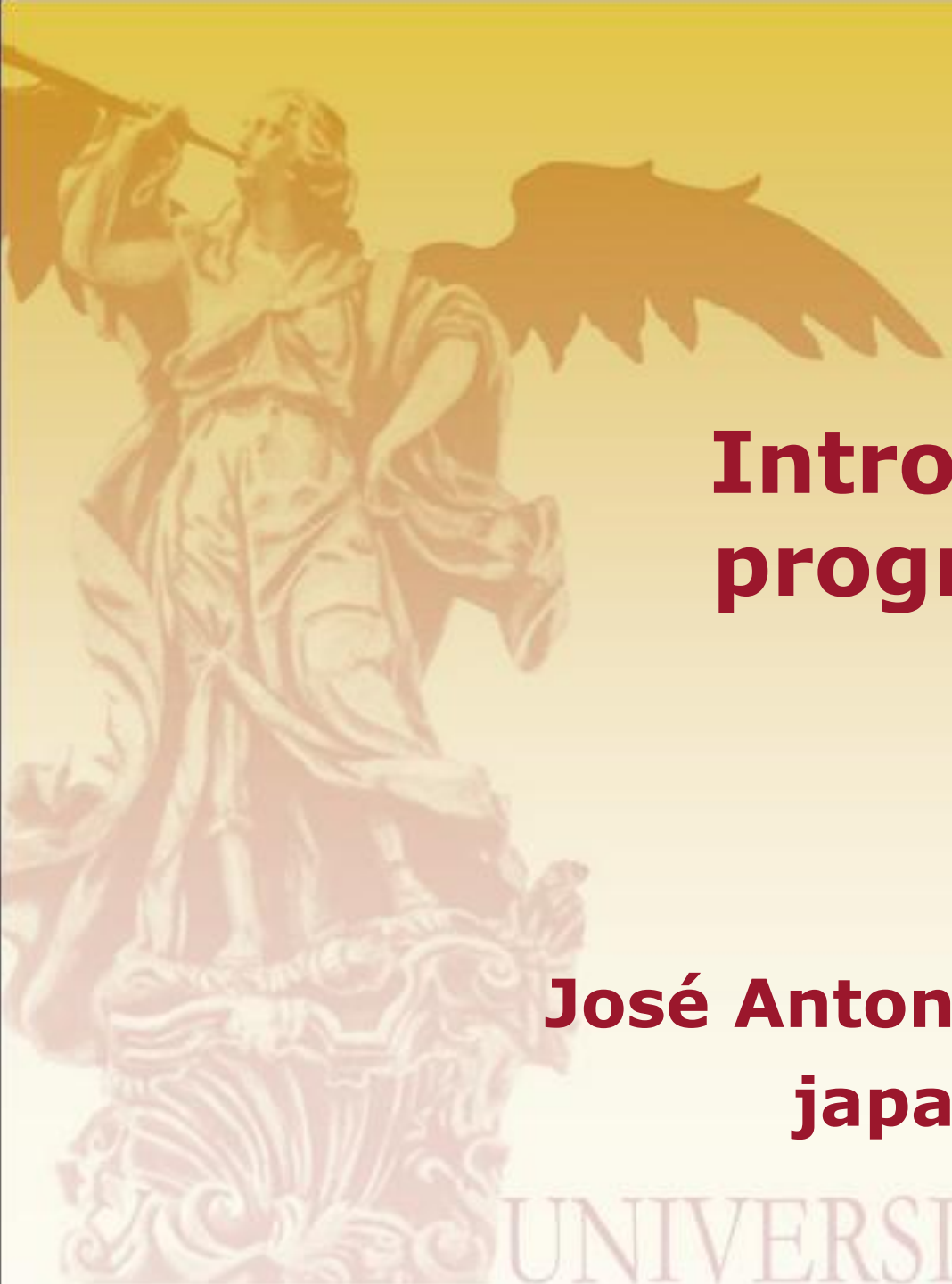


Introducción a la programación en Python

José Antonio Parejo Maestre

japarejo@us.es

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Bienvenida

- Calendario:
  - 21,22,23 y 24 de Junio de 2022

- Horario

  10:00 – 13:00 (con un descanso de 10 min. a las 11:30).

Bienvenida

- Materiales
  - Notebooks y ficheros de código fuente
  - Estas transparencias (a modo de resumen).
  - Material complementario (libros, cheatsheets, etc.)

Bienvenida

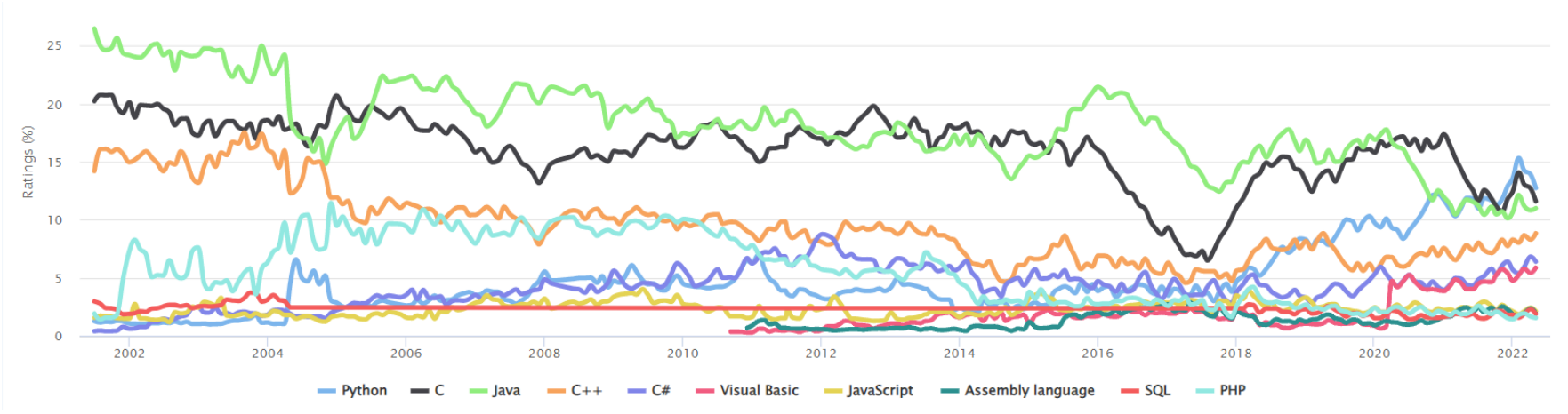
- Asunciones (pre-requisitos):
  - Nociones básicas del uso de computadoras (abrir y ejecutar ficheros, manejo de directorios y carpetas, descomprimir, etc.)
  - Nociones básicas de pensamiento computacional, la ejecución secuencial de instrucciones, etc.

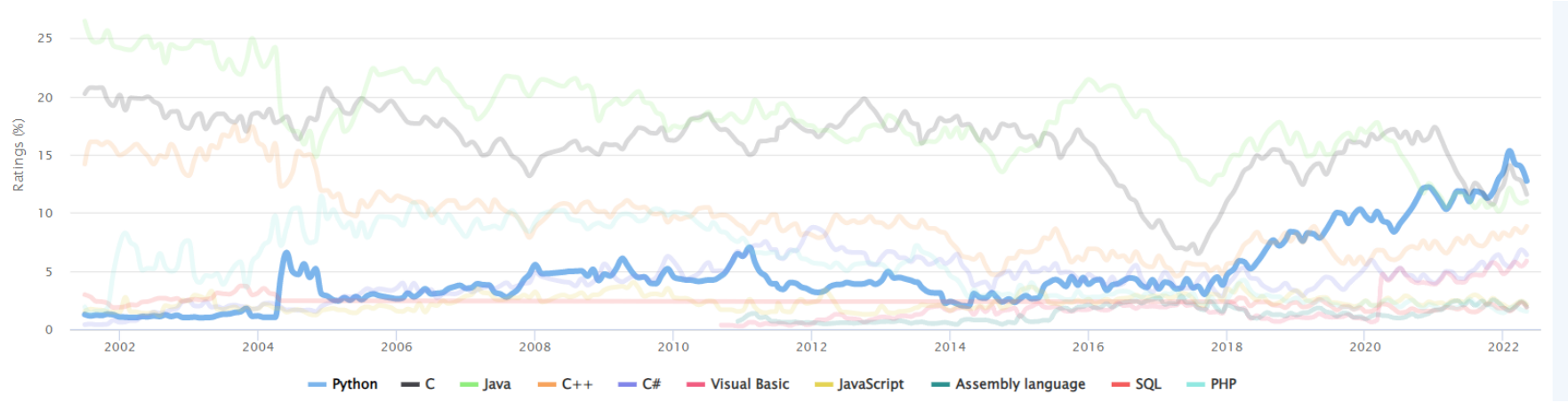
Introducción

Python: el lenguaje de programación más popular del mundo

https://youtu.be/UNSoPa-XQN0?t=145

Tiobe programming languages popularity index (mayo 2022)





https://www.tiobe.com/tiobe-index/

Introducción

Razones por las que Python se ha vuelto tan popular

- **Sintaxis sencilla**, que trata de parecerse al lenguaje natural
- Muy **eficiente** comparado con otros lenguajes similares
- **Multiplataforma:** Soporta distintos entornos de ejecución
- Tiene una **comunidad activa** detrás que ha creado una cantidad ingente de **paquetes / librerías**
- Está bien dotado para la computación numérica, el análisis y tratamiento de datos, y la inteligencia artificial

Introducción

- Guido van Rossum ideó el lenguaje a finales de los 80 y comenzó su implementación en dic. de 1989.
- En febrero de 1991 publicó la primera versión pública, la versión 0.9.0.
- La versión 1.0 se publicó en enero de 1994, la versión 2.0 se publicó en octubre de 2000 y la versión 3.0 se publicó en diciembre de 2008.
- Es posible tener instalados en el ordenador varias versiones de Python pero se recomienda instalar siempre la última versión disponible.

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Configuración del entorno de ejecución

- Elementos del entorno de ejecución de python:
  - **Distribución de Python**:
    La instalación del intérprete del lenguaje y los archivos necesarios para que los programas simples escritos en Python se ejecuten en nuestro ordenador
  - **Gestor de paquetes/librerías**:

    Nos permitirá usar el código creado por otros usuarios de manera cómoda, y compartir ese código entre varios de nuestros proyectos si crear n copias.
  - **Entorno de Desarrollo / Entorno de ejecución de Notebooks**

    Nos permitirá escribir nuestros programas cómodamente, ejecutarlos, visualizar el resultado de su ejecución, y encontrar los errores que podamos haber cometido (depurar).

Configuración del entorno de ejecución

- **Distribuciones y versiones de Python:**
  - Python tiene una cadencia de publicación anual. Así, Python 3.9 se publicó en octubre de 2020, Python 3.10 se publicó en octubre de 2021 y Python 3.11 se publicará en octubre de 2022.
  - La transición de Python 2 a Python 3 resultó mucho más costosa de lo esperado, debido a que Python 3 introdujo muchos cambios en el lenguaje y obligaba a reescribir prácticamente todos los programas (aunque se han creado herramientas para ayudar en ese proceso).
  - El intérprete de lenguaje de Python puede lanzarse con comando “python” o “python3”

Gestor de paquetes/librerías

- Los lenguajes de programación modernos usan librerías para reutilizar el código.
- El Python Package Index (PyPI) es el repositorio de software oficial para paquetes del lenguaje, ofreciendo un catálogo exhaustivo de todos los paquetes de Python de código abierto.
- PIP es un sistema de gestión de paquetes utilizado para instalar paquetes del PyPI y viene instalado por defecto
- Instalar un paquete con pip es tan fácil como escribir en la línea de commandos: *pip install nombre-paquete*
- Podemos crear archivos de requisitos para que se instalen todos de una vez con el commando: *pip install –r requiistos.txt*

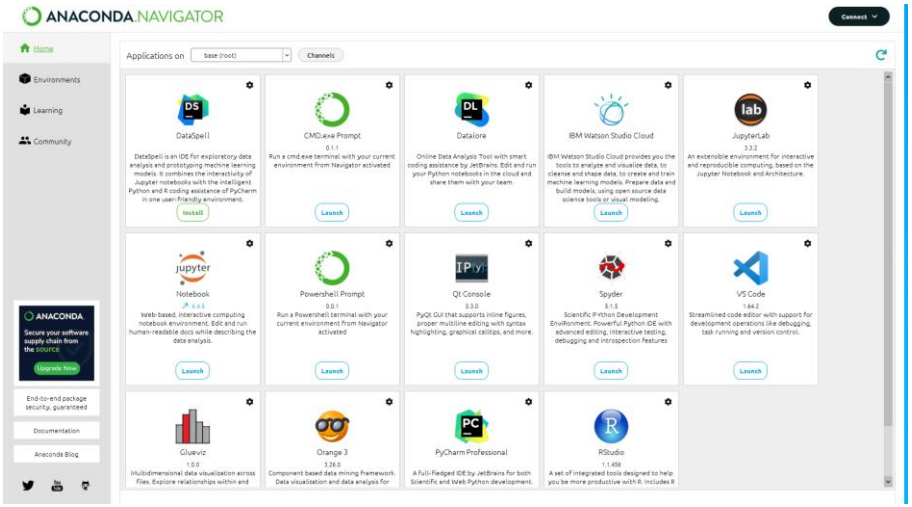
Entorno de desarrollo / Editor

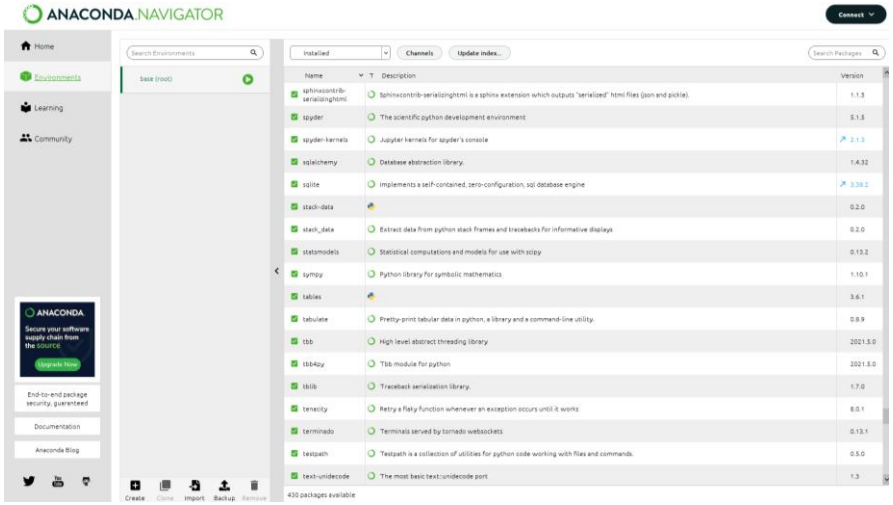
- Existen multitud de entornos de desarrollo para Python, pero os recomandos dos de los más populares:
  - **PyCharm**.
    - Editor con licencia comercial/Freemium
    - Editor específico para python
    - Descarga disponible en https://www.jetbrains.com/es-es/pycharm/

  - **VS Code**
    - Editor gratuito de código abierto
    - Soporta múltiples lenguajes mediante plugins
    - Descarga disponible en https://code.visualstudio.com/download

ANACONDA: Distribución de Python + Gestor de paquetes + Entorno de edición de notebooks

https://www.anaconda.com/products/distribution





Configuración del entorno de ejecución
Nuestra recomendación (para la docencia)

https://www.anaconda.com/products/distribution

https://jupyter.org/try-jupyter/lab/

Configuración del entorno de ejecución
Nuestra recomendación (para programar/investigación)

https://www.anaconda.com/products/distribution

Visual Studio Code

https://code.visualstudio.com/download

+ Python plugins:

- Extensión oficial de MS para Python

  Incluye soporte para el coloreado de sintaxis, la ejecución y depuración.
- Extensión para notebooks Jupyter (.ipynb)

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Descripción del lenguaje

Lenguajes compilados

Lenguajes Interpretados

Python es un lenguaje interpretado:

+ Flexible, no necesita compilación y el código vale para cualquier entorno de ejecución

+ Las modificaciones pueden ejecutarse directamente, no hace falta volver al compilar y los programas pesan menos

- Eficiente, los optimizan el código generado para el hardware

Descripción del lenguaje
El zen de python

Beautiful is better than ugly
Explicit is better than implicit
Simple is better than complex
Complex is better than complicated
Flat is better than nested
Sparse is better than dense
Readability counts
Special cases aren't special enough to break the rules
Although practicality beats purity
Errors should never pass silently
Unless explicitly silenced
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.

Descripción del lenguaje
Nuestro primero programa en Python (en la consola)

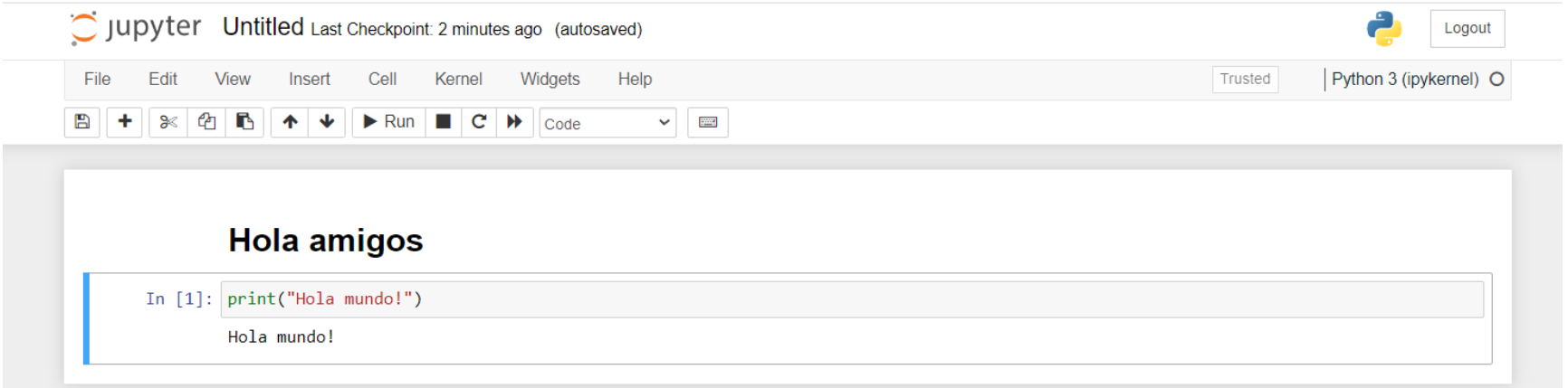
1. Abrir la consola de anaconda (buscarla en el menú de inicio “Anaconda power shell”).
2. Escribir “python” en la línea de comandos
3. Escribir en la consola de python: “print(‘Hola mundo’)”
4. Observar el resultado.

Descripción del lenguaje
Nuestro primero programa en Python (como script)

1. Abrir una consola de Anaconda
2. Abrir cualquier editor de texto (en nuestro caso tenemos instalado VS Code).
3. Crear un nuevo fichero Python llamado HolaMundo.py (File new --> Save --> HolaMundo.py)
4. Escribir en el fichero print(“Hola Mundo”)
5. Ejecutar el fichero con el comando python HolaMundo.py

Descripción del lenguajes
Nuestro primer programa en Python (como notebook)

- Arrancar Jupyter Notebook (desde la consola o desde anaconda navigator)
- Crear un nuevo notebook
- Escribir en la celda que aparece en pantalla print(“Hola mundo”)
- Ejecutar la celda actual con el comando Ctrl+Enter



Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos (de datos)
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Variables y tipos de datos

- Cualquier dato/expresión que se usa en Python tiene asociado un tipo
  - Python es en realidad un lenguaje de tipado dinámico, por lo que el intérprete de Python solo comprueba los tipos al ejecutar el código

- Una variable es una “etiqueta” que asignamos a un dato (más bien a un trozo de memoria capaz de almacenar valores)

- Podemos transformar valores y variables de un tipo a otro (esto recibe el nombre de “casting” en programación).

Funciones por defecto de los tipos

- En Python los tipos por defecto son objetos, y tienen funciones por defecto asociadas para manejarlos.

- Ej: Algunas de las funciones por defecto para str son:

  - upper(): devuelve una versión de la cadena en mayúsculas.
  - lower(): devuelve una versión de la cadena en minúsculas.
  - isnumeric(): devuelve cierto si los caracteres de la cadena son numéricos
  - split(): devuelve una lista de cadenas en base a un separador.

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Operadores y Expresiones

Operadores aritméticos:

Operación	Símbolo	Ejemplo
Adición / Suma	+	1 + 2 = 3
Subtracción / Resta	-	5 - 4 = 1
Multiplicación	*	2 * 4 = 8
División	/	6 / 3 = 2
Exponente	**	3 ** 2 = 9

Operadores lógicos:

Descripción	Símbolo	Ejemplos
Comprobar si los operandos son iguales	==	1 == 1 = True, 1 == 2 = False
Comprobar si los operandos son distintos	!=	1 != 1 = False, 1 != 2 = True
Comprobar si el operando de la izquierda es mayor	>	1>1=False 1>2=False 2>1=True
Comprobar si el operando de la izquierda es menor	<	1<1=False 1<2=True 2<1=False
Comprobar si el operando de izda. es mayor o igual	>=	1>=1=True 1>=2=False 2>=1=True
Comprobar si el operando de izda. es mayor o igual	<=	1<=1=True 1<2=True 2<1=False
Comprobar si ambos operandos son ciertos	and	T and T = T, F and T =F, F and F =F
Comprobar si algún operando es cierto	or	T or T = T, F or T =T, F or F =F
Invertir el valor lógico del operando	not	not T =F, not F=F, not not F = F

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Tipos de datos en python

Tipo de texto:	str
Tipos numéricos:	int, float, complex
Tipos de secuencia:	list, tuple, range
Tipo de mapeo:	dict
Tipos de conjunto:	set, frozenset
Tipo booleano:	bool
Tipos binarios:	bytes, bytearray, memoryview
Tipo genérico (sin tipo):	NoneType

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Colecciones de datos

- En Python existen 4 tipos de colecciones de datos predefinidas por el lenguaje:
  - Listas: ordenada, admite valores repetidos, es mutable e indexable. Ej: [1,8,32,8,198]
  - Conjuntos: no ordenada, no admite valores repetidos, es mutable y no es indexable. Ej: {1,8,32}
  - Tuplas: ordenada, indexable, admite valores repetidos y es inmutable (no se puede modificar una vez declarada). Ej: (2,5)
  - Diccionarios: Indexable mediante claves personalizadas, admite valores repetidos y es mutable. Ej:{“nombre”:”JAParejo”,”edad”:42}

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Funciones

- Una función es un bloque de código solo se ejecuta cuando es invocada.
- Se pueden proporcionar datos como parámetros a una función.
- Una función puede devolver un resultado.
- Ej:

```
def my_function():
  print("Hello from a function")

my_function()
```

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Excepciones

- Una excepción es un objeto que representa una situación inesperada en la ejecución del programa
- El desarrollador puede proporcionarle al intérprete la definición de cómo actuar o reaccionar a la situación.
- Las excepciones en Python se capturan en el contexto del bloques try – except (que pueden anidarse):

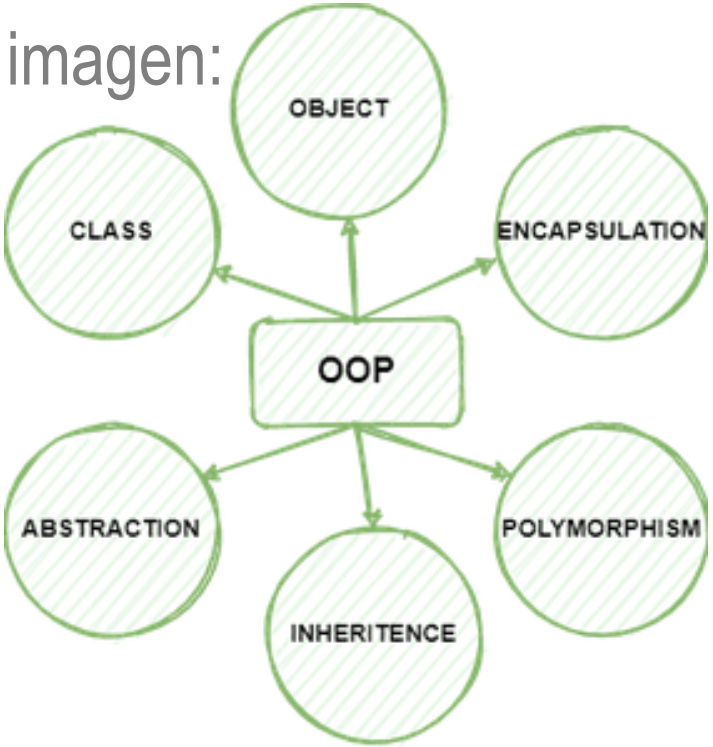
```
try:
    n=int(input("Indique un número entero"))
    try:
        result=2/n
        print("Hemos podido realizar la operación:",result)
    except ZeroDivisionError as error:
        print("Se produjo un error al realizar la operación, división por cero",error)
except Exception:
    print("Se produjo un error al intentar leer el dato, proporcione un valor entero")
```

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Clases y objetos

- La programación orientada a objetos (POO) es un método para estructurar un programa agrupando propiedades y comportamientos relacionados.
- Una clase es un esquema del objeto.
- Los principios clave de la orientación a objetos se resumen en la siguiente imagen:



Clases y Objetos

- Mientras que la clase es el esquema, una instancia es un objeto que se construye a partir de una clase y contiene datos reales.
- Una instancia de la clase Perro ya no es un esquema. Es un perro real con un nombre, como Miles, que tiene cuatro años.

- Los atributos pueden ser atributos de clase (pertenecen a la clase y por tanto son compartidos por todas las instancias), o atributos de instancia, donde cada instancia concreta tiene su propia copia independiente del atributo.

- La creación de un nuevo objeto a partir de una clase se denomina instanciar un objeto.

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Herencia

- La herencia es el proceso por el cual una clase adquiere los atributos y métodos de otra.
- Las clases recién formadas se llaman clases hijas o subclases, y las clases de las que derivan las clases hijas se llaman clases padre o superclases.
- Las clases hijas pueden anular o ampliar los atributos y métodos de las clases padre.
- Pero también pueden especificar atributos y métodos exclusivos para ellas.

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

Módulos y paquetes

- Como en Python se usan varias palabras distintas para denotar al código que reutilizamos, comencemos con unas definiciones básicas:
  - **Script**: es un *archivo* de Python para a ser ejecutado directamente. Suele estar orientado a una tarea, no es reutilizable.
  - **Módulo**: es un *archivo* de Python destinado a ser importado en scripts u otros módulos. Define clases, funciones y variables destinadas a ser utilizadas en otros archivos que lo importan.
  - **Paquete**: es una colección de módulos relacionados que trabajan juntos, y están almacenados en una misma *carpeta*. Esta carpeta suele contener un archivo especial __init__ que indica que se trata de un paquete, que puede contener más módulos anidados en subcarpetas

Nuestro camino

Bienvenida e Introducción
Configuración del entorno de ejecución
Descripción del lenguaje
Variables y tipos
Operadores y expresiones
Control del flujo de ejecución del programa
Colecciones de datos
Funciones
Excepciones
Clases y objetos
Herencia
Módulos y paquetes
Módulos básicos de interés (NumPy, Matplotlib,etc.)

NumPy

- NumPy es una biblioteca de Python utilizada para trabajar con matrices. Fue creada en 2005 por Travis Oliphant. Es un proyecto de código abierto y se puede utilizar libremente.

- También tiene funciones para trabajar en el dominio del álgebra lineal, la transformada de Fourier y las matrices.

- En Python tenemos las listas que cumplen la función de los arrays, pero son lentas. NumPy pretende proporcionar **un objeto array que sea hasta 50 veces más rápido que las listas** tradicionales de Python.

- El objeto array en NumPy se llama ndarray, y proporciona un montón de funciones de apoyo que hacen que trabajar con ndarray sea muy fácil.

- Numpy incorporta métodos para generar un arrays rellenos de ceros o unos, matrices identidad, y datos conforme a distribuciones de probabilidad comunes.

Matplotlib

- Matplotlib es una biblioteca de gráficos de bajo nivel en Python creada por John D. Hunter que sirve para crear diagramas y visualizaciones

- Matplotlib es de código abierto y podemos utilizarla libremente.

- Matplotlib está escrito principalmente en python, pero algunos segmentos están escritos en C, Objective-C y Javascript para alcanzar la máxima eficiencia y compatibilidad con otras plataformas.

- La mayoria de las funciones de Matplotlib se basan en el módulo pyplot, que normalmente se importa con el alias plt.

Pandas

- Pandas es una biblioteca de Python que se utiliza para trabajar con conjuntos de datos.
- Tiene funciones para analizar, limpiar, explorar y manipular datos.
- El nombre "Pandas" tiene una referencia tanto a "Panel Data", como a "Python Data Analysis" y fue creado por Wes McKinney en 2008.

Disclaimer and Terms of Use

All material displayed on this presentation is for teaching and personal use only.

Many of the images that have been used in the presentation are Royalty Free images taken from http://www.everystockphoto.com/. Other images have been sourced directly from the Public domain, from where in most cases it is unclear whether copyright has been explicitly claimed. Our intention is not to infringe any artist's copyright, whether written or visual. We do not claim ownership of any image that has been freely obtained from the public domain. In the event that we have freely obtained an image or quotation that has been placed in the public domain and in doing so have inadvertently used a copyrighted image without the copyright holder's express permission we ask that the copyright holder writes to us directly, upon which we will contact the copyright holder to request full written permission to use the quote or images.