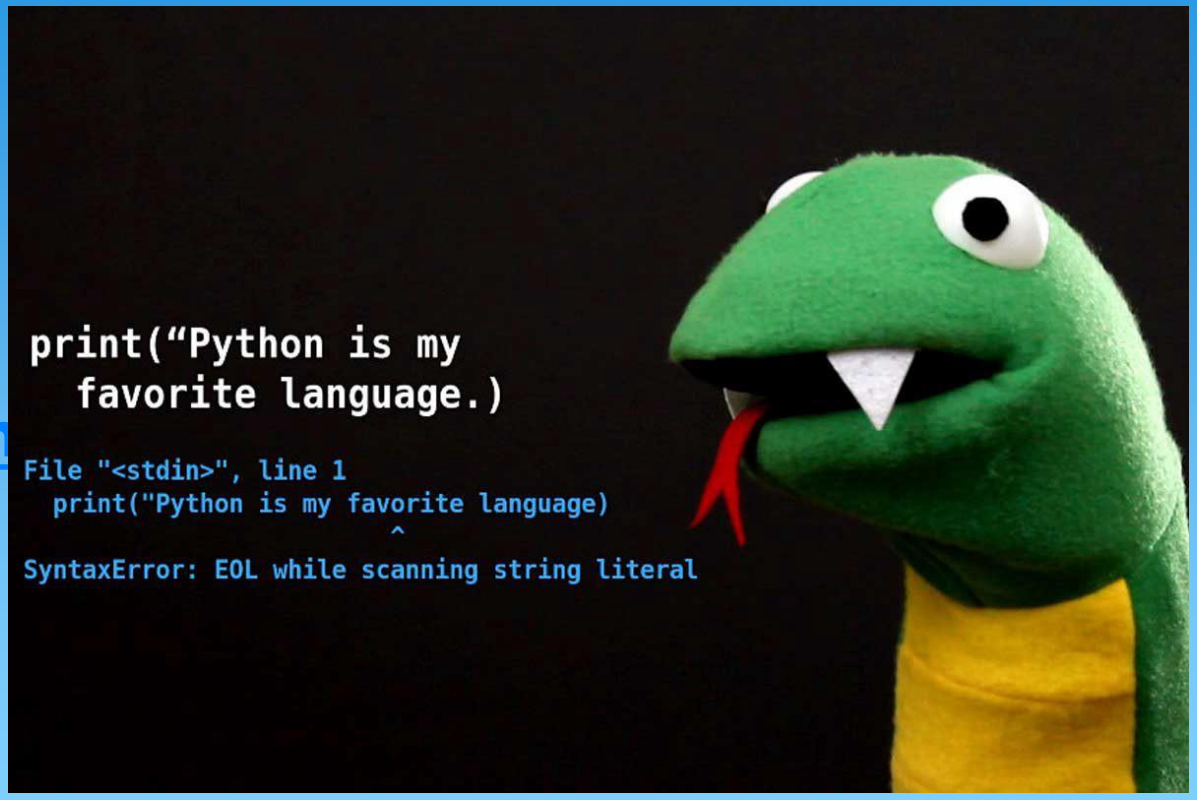

print("Python is my
favorite language.)
File "<stdin>", line 1
print("Python is my favorite language)
^
SyntaxError: EOL while scanning string literal

Características

Simple

Propósito General

Open Source

Lenguaje Orientado a Objetos

Lenguaje de Alto Nivel

Incrustable

Extensas Librerías

Sintaxis clara

C:\Python\Python36\python.exe –i "$(FULL_CURRENT_PATH)"

1er programa

Holamundo**.py**

```
print ("Hello World");
```

Variables

Un **valor** es una de las cosas fundamentales que un programa manipula

Estos valores pertenecen a diferente **clases** (classes) o **tipos**

Python **no define** para cada variable su tipo de valores

```
mientero = 2;
micadena=“mi cadena”;
mireal = 7.0;
mireal2= float(10);
```

```
type(mientero);
type(micadena);
type(mireal);
type(mireal2);
```

Variables

Restricciones a los nombres de las variables:

- Han de comenzar con una **letra** o con el carácter _
- No ha de ser **ninguna** palabra reservada

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

https://docs.python.org/3.6/reference/lexical_analysis.html?highlight=keywords

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Tipos de datos

- Entero

En plataformas de 32 bits → **-2.147.483.648** a **2.147.483.647**

En plataformas de 64 bits → **-9.223.372.036.854.775.808** a **9.223.372.036.854.775.807**

Entero largo (**Long** en C) → Cualquier número, limitado sólo por el hardware

```
# type(entero) daría int
entero = 23
```

```
# type(entero) daría long
entero = 23L
```

- Reales

Doble precisión (**double** en C) → 1 bit el signo, 11 el exponente, y 52 la mantisa

±2,2250738585072020 x 10^{-308} hasta **±1,7976931348623157×10^{308}**

```
# type(real) daría float
real = 23.5
```

```
# type(real) daría float
real = 0.1e-3
```

- Complejos

Tipo de datos *Complex*.

```
# type(complejo) daría float
complejo = 2+4j
```

Variables

- Booleanos

Formado por dos valores: **True** (cierto) y **False** (falso)

En realidad son un subconjunto de los enteros

Son importantes para las condiciones de los bucles y las expresiones condicionales

and	Y → Verdadero si ambos son verdadero true and true → true true and false → false
or	O → Verdadero si alguno es verdadero true or true → true true or false → false
not	No → Negación not true → false not false → true

Operaciones básicas

- Números

+	Suma	3+2 → 5
-	Resta	3-2 → 1
*	Multiplicación	3*2 → 6
/	División	3/2 → 1.5
//	División entera	3//2 → 1
%	Resto	5%3 → 2
**	Exponente	3**2 → 9

- Cadenas

+	Concatenación “ab” + “bc” → “abbc”
*	Multiplicación “abc”*2 → “abcabcabc”

Un programa básico

```
#Primer número
primerN=4;
#Segundo número
segundoN=5;
#Sumo los dos números
resultadoN=primerN+segundoN;
#Obtengo el resultado por pantalla
print("El resultado es:");
print(resultadoN);
```

¿Cómo sería el programa si también quisiéramos que restara, multiplicara, dividiera y nos diera el resto de estos números?

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Otro programa básico

```
#Leo un nombre
nombre=input("¿Cual es la nombre? ");
#Leo un entero
edad=int(input("¿Cual es tu edad? "));
#Leo un real
precio=float(input("¿Cuanto vale un cafe? "));
#Imprimo por pantalla el resultado
print(nombre+" ¿Pagas "+str(precio)+" euros a tus "+str(edad)+" por un café?");
```

Con el programa de la diapositiva anterior y este ¿Cómo sería el ejercicio de los números para que te pida los números por teclado ?

Ejercicios

Una empresa que trabaja con vehículos desea calcular las necesidades de combustible (cantidad de combustible necesario para llenar los depósitos de todos sus vehículos) para lo cual nos han facilitado este esquema de cálculo. Se desea crear un programa para que puedan realizar el cálculo de forma automatizada.

La empresa tiene:

Turismos = 32 con una capacidad de 40 litros cada uno

Todoterrenos = 11 con una capacidad de 65 litros cada uno

El combustible está a 1,005€ el litro

Modifica el programa anterior para que nos pida:

- Cuantos vehículos hay de cada tipo
- Cuantos litros tiene el depósito de cada tipo de vehículo
- Cuál es el precio del combustible

Ejercicios

Crea un programa que nos pida la edad y nos diga cuantos meses, semanas, días, horas, minutos y segundos tenemos de vida.

Restricciones

- Un año tiene 360 días
- Tomaremos los años completos (imaginad que esto se hace el día de vuestro cumpleaños)
- Todos los meses tienen 30 días
- Habéis nacido a las 0 horas

Crea un programa que dado el diámetro de la base de un cilindro y la altura nos diga el perímetro y el área de la base, y el volumen del cilindro.
Nota:

- Perímetro de una circunferencia, dos por pi por radio
- Área de una circunferencia, pi por radio al cuadrado
- Volumen de un cilindro, pi por radio al cuadrado por altura

if ... elif ... else ...

La estructura de control **if** permite que un programa ejecute unas instrucciones cuando se cumplan una condición.

La estructura de control **if.. else** permite que un programa ejecute unas instrucciones cuando se cumple una condición y otras instrucciones cuando no se cumple esa condición

La estructura de control **if ... elif ... else** permite encadenar varias condiciones

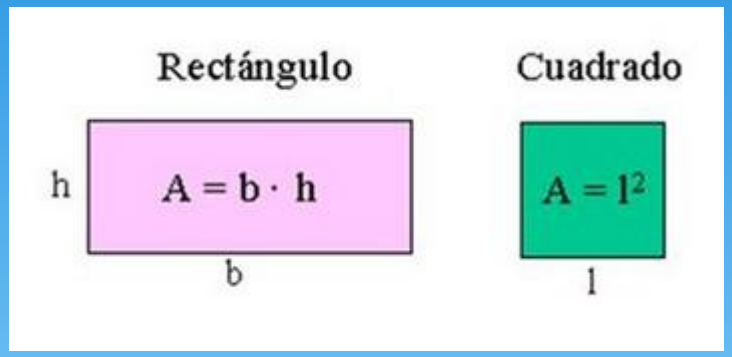
Un **bloque** de instrucciones puede contener **varias** instrucciones. Todas las instrucciones del bloque tener **el mismo sangrado**

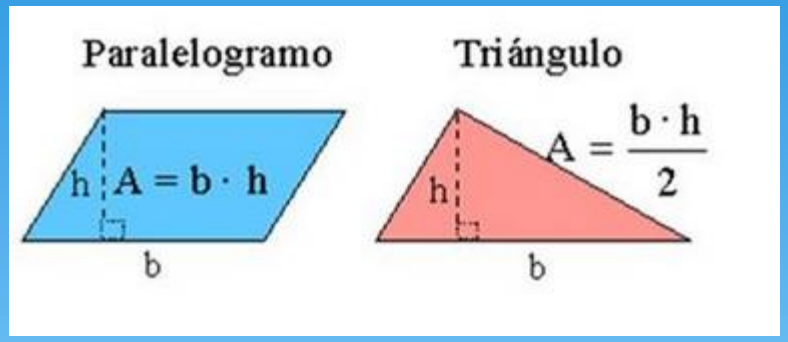
if condición_1: ** bloque 1** **elif** **condición_2:** ** bloque 2** **else:** ** bloque 3**	**if edad>=18:** ** print(“Eres mayor de edad”);** **print(“Ahora ya lo sabes, o no”);**	**if edad>=18:** ** print(“Eres mayor de edad”);** **else:** ** print(“Eres menor de edad”);** **print(“Ahora ya lo sabes, o no”);**	**if edad<12:** ** print(“Eres un crio”);** **elif edad <18:** ** print(“Eres menor de edad”);** **else:** ** print(“Eres mayor de edad”);** **Print(“Ahora ya lo sabes, o no”);**

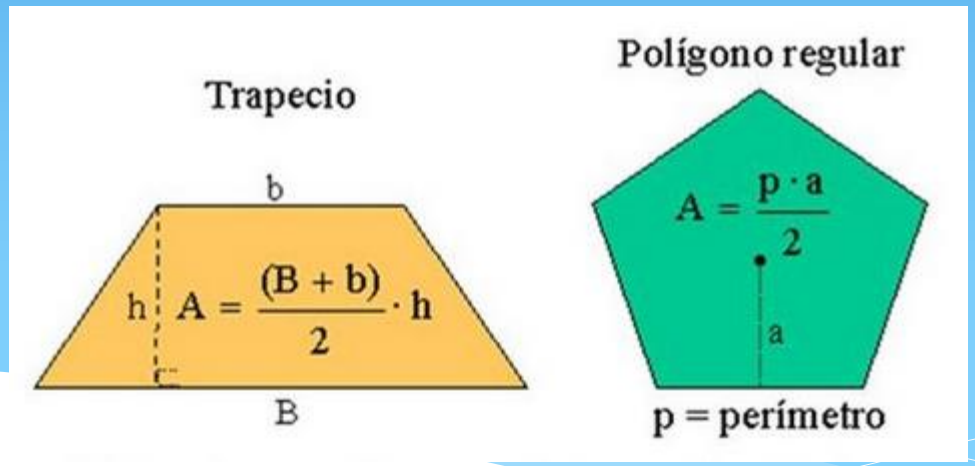
No existe **CASE** o **SWITCH**

Ejercicios

Crea un programa que calcule áreas de **una** de las siguientes figuras geométricas:







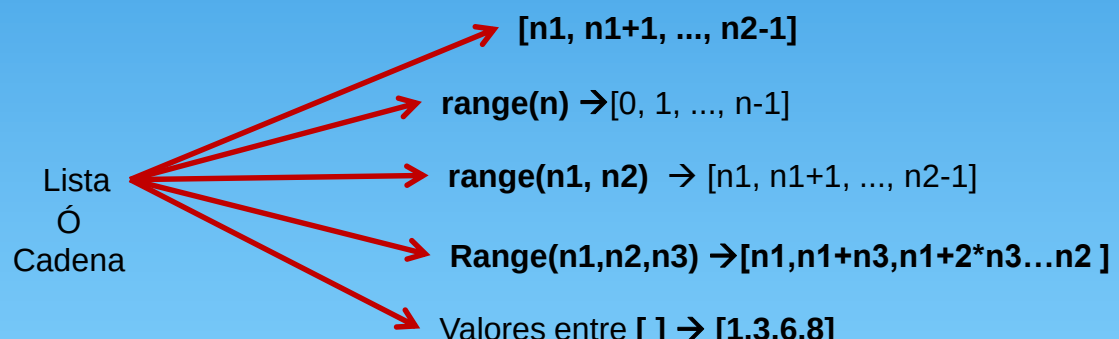
Ciclos (Bucles)

Es una sentencia que se realiza repetidas veces a un trozo aislado de código.

Se ejecuta hasta que la condición asignada a dicho bucle deje de cumplirse.

Ciclos definidos. Se sabe exactamente cuántas veces se ejecutará el código

for <variable> **in** <Lista>**:**
<cuerpo>




```
for x in [1, 2, 3, 4]:
    print(x*x)
```

```
for x in range(5):
    print(x*x)
```

```
for x in range(2,7):
    print(x*x)
```

```
for x in range(1,8,2):
    print(x*x)
```

```
for x in [5, 2, -3, 4]:
    print(x*x)
```

```
for x in “Amigo”:
    print(x)
```

Ciclos (Bucles)

Ciclos indefinidos. Se repite el cuerpo mientras una cierta condición sea verdadera

PELIGRO!!!! → Puede ocurrir un bucle INFINITO!!!!!!!

```
while True:
   x = input("Ingrese un numero ('*' para terminar): ");
   if x == "*":
      break;
   else:
      nx = int(x);
      if nx > 0:
         print("Número positivo");
      elif nx < 0:
         print("Número negativo");
      else:
         print("Es cero");
```

Ciclos (Bucles). Ejercicios

1.- Crea un programa que pida dos números y cuente desde el primero al segundo

2.- Crea un programa que pida las 8 notas de un alumno, calcule la nota media y saque por pantalla la nota máxima, la mínima y la nota media de dicho alumno.

3.- Escribir un programa que contenga una contraseña inventada, que le pregunte al usuario la contraseña, y no le permita continuar hasta que la haya ingresado correctamente.

4.- Modificar el programa anterior para que solamente permita una cantidad fija de intentos.

5.- Crear un programa que calcule el factorial de un número dado.

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Cadenas de Texto

En Python las cadenas de texto se delimitan con comillas simples (') o dobles (")

En Python las comillas dobles y las comillas simples son completamente equivalentes,

Las cadenas se deben cerrar con las mismas comillas con las que se abrieron

```
>>> "Esto es una cadena"
'Esto es una cadena'
```

Se pueden escribir comillas simples en cadenas delimitadas con comillas dobles y viceversa

```
>>> "Las comillas simples ' delimitan cadenas."
"Las comillas simples ' delimitan cadenas."
>>> 'Las comillas dobles " delimitan cadenas.'
'Las comillas dobles " delimitan cadenas.'
```

No se pueden escribir en el interior de una cadena comillas del mismo tipo que las comillas delimitadoras

```
>>> "Las comillas dobles " delimitan cadenas"
SyntaxError: invalid syntax
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Cadenas de Texto

Dentro de una cadena, escribir **\"** y **\'** representa los caracteres comillas dobles y simples respectivamente y Python los no interpreta en ningún caso como delimitadores de cadena

```
>>> 'Las comillas simples \' delimitan cadenas.'
"Las comillas simples ' delimitan cadenas.“
 >>> "Las comillas dobles \" delimitan cadenas."
'Las comillas dobles " delimitan cadenas.'
```

Caracteres especiales

Los caracteres especiales empiezan por una **contrabarra** (\).

- Comilla doble: \"

```
>>> print("Las comillas  \" delimitan cadenas.“)
 'Las comillas " delimitan cadenas.'
```

- Comilla simple: \'

```
>>> print("La comilla  \’delimita cadenas.“)
 'Las comilla ‘ delimitan cadenas.'
```

- Salto de línea: \n

```
>>> print(“Una línea\nOtra línea.“)
Una línea
Otra línea
```

- Tabulador: \t

```
>>> print("1\t2\t3")
1               2               3
```


```
nombre="Veronica";
print(len(nombre));
print(nombre[2:4]);
print(nombre[-4:-2]);
print(nombre[:3]);
print(nombre[3:]);
print(nombre[:-3]);
print(nombre[-3:]);
```

cadena = "Hola amiguetes"

```
print(cadena[1:2]) # Esto dará como resultado 'o'
print(cadena[0:6]) # 'Hola a‘
print(cadena[0:]) # Si no indicamos fin recogemos la cadena hasta el final 'Hola amiguetes'
print(cadena[5:]) # 'amiguetes'
print(cadena[:5]) # 'Hola‘
print(cadena[:]) # 'Hola amiguetes'
print(cadena[-8]) # 'm' recoge el octavo elemento empezando por el final
print(cadena[-8:]) # 'miguetes' desde el octavo empezando por el final hasta el final
print(cadena[:-4]) # 'Hola amigu' Desde el principio hasta el cuarto elemento empezando por el final
```

Operaciones con cadenas

Acceder al elemento i de una cadena→ **cadena[i]**

El primer carácter es el 0 No podemos cambiar el valor de una posición de una cadena

Encontrar un carácter→ **cadena.find(carácter)** → Devuelve la posición del carácter en la cadena

Reemplazar unos caracteres por otros→ **cadena.replace(cadena, cadena)**

Elimina todos los espacios en blanco por los extremos, solo los que aparecen a la izquierda y solo los que se encuentran a la derecha → **cadena.strip()** **cadena.lstrip()** **cadena.rstrip()**

Divide una cadena por un carácter→ **cadena.split(caracter)** → Devuelve un array

Inversa de split→ **caracter.join(array de cadenas)** → Devuelve una cadena

```
cadena = "      esta cadena tiene espacios a los lados      ";
print (cadena.find("t"));
print (caena.replace("cadena","línea"));
print(cadena.strip());
print(cadena.lstrip());
print(cadena.rstrip());
print(cadena.split("a"));
print(cadena[15]);
```

¿Qué hace?

```
cadena = "      esta cadena tiene espacios a los lados      “;
print (cadena.find(“t”));
print (caena.replace(“cadena”,”línea”));
print(cadena.strip());
print(cadena.lstrip());
print(cadena.rstrip());
print(cadena.split(“a“));
print(cadena[15]);
```

Ejercicios con cadenas

1. Escribir una función que reciba una cadena que contiene un largo número entero y devuelva una cadena con el número y las separaciones de miles. Por ejemplo, si recibe 1234567890, debe devolver 1.234.567.890.

2. Hacer un programa que dada una palabra, y un número (menor que la longitud de la palabra) te devuelva la palabra partida en tantos trozos como indique el numero, del mismo tamaño.
   Ejemplo: **"Otorrinolarigologo" y 3 → Otorri nolari gologo**

3. Hacer un programa que dado un texto y una vocal, te calcule la longitud del texto y cambie todas las vocales por la vocal introducida

4. Hacer un programa que simule el juego del ahorcado (para dos jugadores). Ha de contar el número de fallos y como mucho son 11

Colecciones

Listas

Es un tipo de colección **Ordenada** → Equivale a un array

Admite **cualquier** tipo de datos. Incluso listas.

No tienen porque ser todos los elementos del mismo tipo.

```
>>> lista=[22, True, “una lista”, [1,2]]
```

Se indican los valores de la lista entre **[** y **]** y separados por **coma** cada valor.

El **primer** elemento siempre es el **0**

Acceso a cada elemento →

```
>>> lista[2]
‘una lista’
```

```
>>> lista[2][1]
n
```

Se permite modificar el valor de cada elemento

```
>>> lista[2]
‘una lista’
>>> lista[2]=“Vaya lista”
>>> lista[2]
‘Vayalista’
```

Colecciones

Listas

Son **OBJETOS**. Tienen sus propios **métodos**

Python permite índices **negativos** → Se comienza a contar por el final
Siento el **-1** el **último elemento**

Python hace **Slicing**

Lista [inicio : fin]
Lista [inicio : fin : salto] → Coge desde el elemento *inicio* hasta *fin-1*
Coge desde el elemento *inicio* hasta *fin-1* de *salto en salto*

```
>>> lista=[22,True,"una lista",34]
>>> lista[1:3]
[True, 'una lista']
>>> lista[1:5:2]
[True, 34]
```

Los **negativos** funcionan igual

Si no se indica **principio** o **fin**, se tomará el **primero** o el **último** respectivamente

```
>>> lista=[22,True,"una lista",34]
>>> lista[2:]
["una lista",34]
>>> lista[:3]
[22,True,”una lista”]
```

En una asignación, si los valores de la **derecha** son **más** o **menos** que los existentes se modifica la longitud de la lista

```
>>> lista=[1,2,3,4,5,6,7,8,9,0]
>>> lista
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
>>> lista[-1:]=[10,11,12,13,14]
>>> lista
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Colecciones

Tuplas

```
>>> lista=(22, True, “una lista”)
```

Es lo mismo que las **Listas** → Cambia la forma de definirlo

En vez de usar **[** y **]** utilizamos **(** y **)**

El constructor es la **,** no los **()**

```
>>> lista=(22, True, “una lista”)
```
=
```
>>> lista=22, True, “una lista”
```

Una tupla de un **único elemento** se construye con el valor y una **,**

Acceso como las **Listas**

Salvo

Son **inmutables**

No son **OBJETOS** → No tienen métodos

```
>>> tupla=(1)
>>> type(tupla)
<class 'int'>
>>> tupla=(1,)
>>> type(tupla)
<class 'tuple'>
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Colecciones

Diccionarios

Matrices asociativas → Relacionan **clave** y **valor**

Se definen como las listas

Cada elemento formado por dos valores separados por **:**

A la izquierda de **:** está la **clave** y a la derecha está el **valor**

La clave es un **valor inmutable** (números, cadenas, tuplas… pero **no** listas o diccionarios)

Se accede al dato por la **clave** no por el índice usando **[** y **]**

```
>>> cine={"Los pajaros":"Alfred hichcock", "Kill bill":"Quentin Tarantino"
... , "Banderas de nuestros padres":"Clint Eastwood"}
>>> cine["Kill bill"]
'Quentin Tarantino'
```

Para añadir o modificar un elemento

```
>>> cine={"Los pajaros":"Alfred hichcock", "Kill bill":"Quentin Tarantino"
... , "Banderas de nuestros padres":"Clint Eastwood"}
>>> cine["Casablanca"]="Michael Curtiz"
>>> cine["Casablanca"]
'Michael Curtiz'
```

NO admite slicing

Para saber más de colecciones

https://docs.python.org/3.6/tutorial/datastructures.html

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Ejercicios

1. Escribe un programa que dada una lista de números, nos muestre el mínimo, el máximo y la media.

2. Escribe un programa que pida números (acabada con un <intro>). Luego calcula la media aritmética, da el valor máximo y el mínimo.

3. Escribe un programa que pida una lista de nombres (acabada con un <intro>). El programa mostrará la lista de nombres en orden alfabético sin repetir ninguno

4. Escribir un programa que pida un número superior a 300, y haga la Criba de Eratóstenes.

5. Haz el juego del ahorcado, de tal manera que a cada intento se muestre la solución encontrada a cada momento (A entregar)

Funciones

Python es un lenguaje funcional, se basa en funciones

Python al igual que otros lenguajes funcionales, permite al usuario definir sus propias funciones

Entre la última línea de código y la definición de la función **ha de haber 2 (dos)** líneas en blanco

Formato:

```
def mi_funcion():
        # aquí el algoritmo
```

```
def mi_funcion( par1, par2, … ):
        # aquí el algoritmo
```

```
>>> def mi_funcion():
...     print("Hola");
...
>>> mi_funcion()
Hola
```

La función no se ejecuta hasta que no se invoca

```
mi_funcion()
```

Parámetros

- valor que la función espera recibir cuando sea llamada (invocada)
- variables de **ámbito local**
- Respetar el **Orden de los parámetros al pasar los valores**

Toda función puede, o **no**, contener la instrucción **return** para devolver **UN UNICO** resultado

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Funciones

Se le puede dar un **valor por defecto** a los parámetros

```
def saludar(nombre, mensaje='Hola'):
        print (mensaje+" "+nombre)

if __name__ == '__main__':
        saludar('Pepe Grillo') # Imprime: Hola Pepe Grillo
```

NO SE ADMITEN espacios en blanco alrededor del **=**

Podemos pasar los argumentos como pares **clave=valor**. El **orden** no importa

```
def saludar(nombre, mensaje='Hola'):
        print (mensaje+" "+nombre)

saludar(mensaje="Buen día", nombre="Juancho")
```

Podemos no saber cuantos parámetros vamos a pasarle a la función. Entonces se le pasan mediante una **TUPLA**

```
def recorrer_parametros_arbitrarios(parametro_fijo, *arbitrarios):
        print (parametro_fijo)   # Los parámetros arbitrarios se corren como tuplas
        for argumento in arbitrarios:
                print (argumento)

recorrer_parametros_arbitrarios('Fixed', 'arbitrario 1', 'arbitrario 2', 'arbitrario 3')
```

Se colocan **TRAS** los fijos

Funciones

Se pueden pasar parámetros arbitrarios como pares de **clave=valor**.

El nombre del parámetro debe ser precedido de **dos astericos (**)**

```
def recorrer_parametros_arbitrarios(parametro_fijo, *arbitrarios, **kwords):
        print (parametro_fijo);
        for argumento in arbitrarios:
                print (argumento)
                # Los argumentos arbitrarios tipo clave, se recorren como los diccionarios
        for clave in kwords:
                print ("El valor de“+ clave + "es“ + kwords[clave])

recorrer_parametros_arbitrarios("Fixed", "arbitrario 1", "arbitrario 2", "arbitrario 3",
        clave1="valor uno", clave2="valor dos")
```

Cuando pasamos valores mediante una **Lista** ó **tupla** se pasan los valores con *****, si es un **diccionario** se pasa mediante ******

```
def calcular(importe, descuento):
    return importe - (importe * descuento / 100) ;

if __name__ == '__main__':
    datos = [1500, 10]
    print (calcular(*datos) );
```

```
def calcular(importe, descuento):
    return importe - (importe * descuento / 100) ;

if __name__ == '__main__':
    datos = {"descuento": 10, "importe": 1500};
    print (calcular(**datos)) ;
```

Ejercicios

1. Escribir un programa que utilice una función que nos diga si un número es par o impar

2. Utilizar esa función en la criba de Eratóstenes

3. Crear un programa que mediante funciones haga (cada apartado una función):
   a) Pedir un numero
   b) Pedir tantas palabras como indiquen el numero introducido en el paso anterior y meterlo en una lista
   c) Recorrer a lista
   d) Para cada palabra obtener el numero de letras que tiene
   e) Imprimir la palabra y la longitud de la palabra

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Excepciones

Cuando se ejecuta un programa y se produce un error, python lanza una excepción

Si la excepción no es capturada, el programa se detiene

Al producirse el error, se produce un volcado de las llamadas y muestra la excepción generada

```
def division(a,b):
   return a/b;

def divide(uno,dos):
   return division(uno,dos);

print(divide(1,0));

Traceback (most recent call last):
  File "<module1>", line 7, in <module>
  File "<module1>", line 5, in divide
  File "<module1>", line 2, in division
ZeroDivisionError: division by zero
>>>
```

Excepciones

Se usa una estructura try-exception

El bloque *try* contiene el código que puede generar la excepción

El bloque *exception* trata la excepción capturada

Se puede capturar una excepción general con ***except:***

O una en particular ***except ZeroDivisionError:***

O varias en particular ***except (ZeroDivisionError, NameError)***

```
def division(a,b):
  try:
      return a/b;
   except ZeroDivisionError:
       print('El divisor no puede ser 0');

def divide(uno,dos):
   return division(uno,dos);

print(divide(1,0));
```

Excepciones

Estructura

try:
    <<Código a ejecutar>>
except: except <<nombreExcepción>>: except (<<excepciones>>):
    <<tratamiento de una excepción general>>
finally:
    <<Código a ejecutar tras la zona conflictica, tanto si hay como si no hay excepcion>>

```
try:
  z=x/y;
except ZeroDivisionError
  print('División por cero');
finally:
  print('Limpiando');
```

Más sobre excepciones

http://docs.python.org.ar/tutorial/3/errors.html

Built-in Exceptions

https://docs.python.org/3/library/exceptions.html

Ejercicios

1. Crea una calculadora que solo permita números y operaciones. Ha de pedir primero la operación (si no es una operación válida a de volver a pedirla). Después ha de pedir los operandos y si no es un número ha de preguntarnos si queremos cambiarlo. En caso de una operación de división por cero, debe preguntarnos si queremos cambiar el divisor.

```
Operación: +
>4
>5
>=
9
```

```
Operación: /
>10
>0
>=
División por 0. ¿Desea cambiar el divisor? Y
>2
5
```

Excepciones - Levantar

El programador puede levantar su propia excepcion

raise NameError(“*texto*”)

Raise indica la excepción a levantarse.

- Instancia de excepción
- Clase de excepción → se instancia en el levantamiento

```
>>> raise NameError('Hola')
 Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
NameError: Hola
```

```
raise ValueError # atajo para 'raise ValueError()'
```

```
>>> try:
 ... raise NameError('Hola')
 ... except NameError:
 ... print('Voló una excepción!')
 ... Raise
 ...
Voló una excepción!
 Traceback (most recent call last):
 File "<stdin>", line 2, in <module>
NameError: Hola
```

Ejercicios

2. Crea un programa que pida los nombres de personas hasta que pulsemos solo el enter. Después irá pidiendo un número y mostrará el nombre de la persona que hayamos metido en esa posición.

3. Realiza una función llamada **agregar_una_vez(lista, elemento)** que reciba una lista y un elemento. La función debe añadir el elemento al final de la lista con la condición de no repetir ningún elemento. Además si este elemento ya se encuentra en la lista se debe invocar un error de tipo *ValueError* que debes capturar y mostrar este mensaje en su lugar:

   Error: Imposible añadir elementos duplicados => [elemento].

Nota: "2" ***in*** *["1","2","3","4"] devuelve true*

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Módulos y paquetes

Python permite la división en módulos

- Simplifica el diseño.
- Disminuye la complejidad de los algoritmos.
- Disminuye el tamaño total del programa.
- Ahorra en tiempo de programación al promover la reusabilidad del código.
- Favorece el trabajo en equipo.
- Facilita la depuración y prueba.
- Facilita el mantenimiento.
- Permite la estructuración de librerías específicas.

Cada módulo recibe el nombre *nombre**.py***
Para usar el módulo *nombre**.py*** en el programa se ha de usar la clausula *import*

import nombre ← Estará en el mismo directorio que lo importa

Import permite varios módulos a la vez *import nom1, nom2, nom3*

Ejercicio

1. Partiendo del ejercicio de la calculadora modifícalo para que cumpla:
   A. Un módulo para cada operación (suma, resta, división, multiplicación)
   B. Cada módulo ha de tener operaciones para operar con 2, 3, 4 o 5 números
   C. Dependiendo del número de elementos que se introduzcan por teclado se llamará a la operación correspondiente.
   D. Sólo se puede realizar un tipo de operación, no mezclar operaciones
   E. Se han de tener en cuenta las excepciones

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Clases

Modelos de nuestros objetos

El nombre de la clase es en singular y en CamelCase

Cada clase esta compuesta por:

- Atributos
- Mátodos

Formato:

class *nombre*

atributos → Variables

métodos → Funciones

Se instancia llamando solo al objeto

El primer parámetro de un método es siempre *self*

Cada objeto puede o no tener un método constructor que se llama `__init __`

```
class objeto:
        dia=0;
        def __init__(self, numero):
                self.dia=numero;
```

```
---Programa principal--
        dia_mes=objeto(7);
        print(dia_mes.dia);
```

Ejemplos

```
class Objeto:
    color = "verde"
    tamanio = "grande"
    aspecto = "feo"

    def flotar(self):
        pass
```

Instrucción de rellenp →

```
et = Objeto()
print (et.color)
print (et.tamanio)
print (et.aspecto )
et.color = "rosa"
print (et.color)
```

Animal
• Patas • Pelo • Nocturno Sonido() Comer()

```
class Animal:
    patas=-1
    pelo=False
    nocturno=False

    def __init__ (self, pat, pel, noc):
            self.patas=pat;
            self.pelo=pel;
            self.nocturno=noc
```

```
lechuza = Animal(2,False, True)
perro = Animal(4,True, False)
print (lechuza.patas)
print (perro.patas)
```

Ejercicios

- Crear la clase números complejos que permita sumar y restar números complejos. Ej: 3+1i + 5-2i = 8 -1i

Es decir, crea una clase que tiene dos atributos llamados real e imaginario, de tal forma que al sumar dos objetos de la misma clase, sólo podemos sumar atributos iguales.

- Crear una clase llamada persona que identifica a una persona, va a tener como atributos "nombre“, "edad“, “dni”, “sexo”, “peso” y “altura” y los métodos para utilizarlos

Los métodos del ejercicio anterior que se implementaran son:
calcularIMC(): devolverá el imc de la persona (peso en kg/(altura^2 en m)),

esMayorDeEdad(): indica si es mayor de edad, devuelve un booleano.

Ejercicios

- Crear la clase vehículos que modele los diferentes vehículos (tipo, ruedas, color, puertas, combustible, abs) y sus métodos.

- Crear un programa principal que nos permita a través de un menú, crear una serie de vehículos (por cada vehículo nos pida sus características). Al dar a la opción de salir, ha de hacer un listado de los vehículos que tenemos creados.

- Añadir el método que permita eliminar un vehículo de la lista

Clases

Herencia

```
class NombreSubClase (NombreClaseSuperior):
  'Cadena de documentación opcional'
  Declaración de atributos y métodos...
```

- Se crea una clase a partir de otra ya existente
- Se hereda atributos y métodos
- Se declara como se declara una clase pero se pone entre paréntesis la clase padre.
- Método super()

```
class instrumento:
    precio=0;
    def __init__ (self, pre):
        self.precio=pre;
```

- La herencia múltiple utiliza varias clases padre en la definición.

```
class NombreSubClase (Clase1, Clase2, Clase3):
  'Cadena de documentación opcional'
  Declaración de atributos y métodos...
```

```
Class Guitarra(instrumento):
    n_cuerdas=0;
    def __init__ (self, n, prec):
      super().__init__(prec)
      n_cuerdas=n;
```

Clases

Funciones para atributos: getattr(), hasattr(), setattr() y delattr()

getattr() → accede al valor del atributo de un objeto. Si un atributo no existe retorna el valor del tercer argumento (es opcional)

hasattr() →devuelve ***True*** o ***False*** dependiendo si existe o no el atributo indicado.

setattr() → Se utiliza para asignar un valor a un atributo. Si el atributo no existe entonces será creado.

delattr() →es para borrar el atributo de un objeto. Si el atributo no existe se producirá una excepción del tipo ***AttributeError***.

```
nota_alumno2 = getattr(alumno2, 'nota', 0)
edad_alumno2 = getattr(alumno2, 'edad', 0)
print(nota_alumno2)  # 6
print(edad_alumno2) #18
if not hasattr(alumno2, 'edad'):
   print("El atributo 'edad' no existe")
setattr(alumno2, 'edad', 18)
print(alumno2)  # 18
```

```
delattr(alumno2, 'edad')
delattr(alumno2, 'curso')
```

Clases

Encapsulación

- Marca el acceso a los métodos y atributos
- Si comienza con dos guiones bajos, es privado, sino, es público
- No hay que equivocarlo con los métodos que comienza y terminan con dos guiones bajos, que son métodos especiales de python

```
class Ejemplo:
    def publico(self):
        print("Publico");

    def __privado(self):
        print("Privado");

Ej = Ejemplo();
Ej.publico();
Ej.__privado();
```

Clases

Encapsulación - Ejercicio

Tenemos la clase punto. Encapsúlala de tal forma que sólo sean accesibles desde fuera sus **getters y setters**

```
class punto:

    _ _x=0;
    _ _y=0;
    def _ _init_ _(self,dx,dy):
        self.x=dx;
        self.y=dy;

    def getX(self):
        return self.x;

    def getY(self):
        return self.y;

    def setX(self,dx):
        self.x=dx;

    def setY(self,dy):
        self.y=dy;
```

Clases

Métodos especiales

__init__(self, args) → Método llamado después de crear el objeto para realizar tareas de inicialización.

__new__(cls, args) → Método exclusivo de las clases de nuevo estilo que se ejecuta antes que __init__ y que se encarga de construir y devolver el objeto en sí.

__del__(self) → Método llamado cuando el objeto va a ser borrado. También llamado destructor, se utiliza para realizar tareas de limpieza.

__str__(self) → Método llamado para crear una cadena de texto que represente a nuestro objeto.

__cmp__(self, otro) → Método para comparar objetos. Debe devolver un número negativo si nuestro objeto es menor, cero si son iguales, y un número positivo si nuestro objeto es mayor.

__len__(self) → Método llamado para comprobar la longitud del objeto.

Empaquetado - Módulos

Un módulo es una colección de métodos en un fichero que acaba en **.py**

El nombre del fichero determina el nombre del módulo en la mayoría de los casos

E.j. **modulo.py**

```
def one(a):
    print "in one"
def two (c):
    print "in two"
```

Uso de un módulo:

```
>>> import modulo
>>> dir(modulo) # lista contenidos módulo
['__builtins__', '__doc__', '__file__',
'__name__', 'one', 'two']
>>> modulo.one(2)
in one
```

Empaquetado - Módulos

`import` hace que un módulo y su contenido estén disponibles para su uso

Algunas formas de uso son:

`import test` → Importa modulo `test`. Referir a `x` en `test` con `"test.x"`

`from test import x` →Importa `x` de test.
Nos referiremos a `x` en `test` con "x".

`from test import *` →Importa todos los objetos de `test`.
Nos referiremos a x en `test` con "x".

`import test as theTest` → Importa `test`; lo hace disponible como `theTest`.
Referir a objecto `x` como `"theTest.x"`

$$m = \frac{y_2 - y_1}{x_2 - x_1} \qquad d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad PM = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$$

Ejemplo: Clase que modela un punto en un plano

```
class punto:
    x=0;
    y=0;
    def __init__(self,dx,dy):
        self.x=dx;
        self.y=dy;

    def getX(self):
        return self.x;

    def getY(self):
        return self.y;

    def setX(self,dx):
        self.x=dx;

    def setY(self,dy):
        self.y=dy;
```

```
class recta:
    origen="";
    destino="";

    def __init__(self, ori, dest):
        self.origen=ori;
        self.destino=dest;

    def longitud(self):
        restax=self.destino.getX()-self.origen.getX();
        restay=self.destino.getY()-self.origen.getY();
        return math.sqrt((restax)**2 + (restay)**2);
```

Clases – super()

Es una función *build-in* que sirve para acceder a atributos que pertenecen a una clase superior

Muy sencillo y evidente

- ~~Tiene como parámetros la clase hijo~~
- ~~Y si mismo~~
- Y se llama al método del padre

super().<método>()

```
01 class Madre(object):
02     def apellido(self):
03         print " Gonzalez "
04
05 class Hijo(Madre):
06     def nombre(self):
07         print " Manuel "
08         super().apellido()
09
10 hijoA = Hijo()
11 hijoA.nombre()
```

En caso de herencia múltiple, la búsqueda se hace de derecha a izquierda

```
01 class Madre(object):
02     def apellido(self):
03         print " Gonzalez "
04 class Padre(object):
05     def apellido(self):
06         print " Sanz "
07 class Hijo(Madre, Padre):
08     def nombre(self):
09         print " Manuel "
10         super().apellido()
```

```
15 hijoA = Hijo()
16 hijoA.nombre()
17 hijoB = Hija()
18 hijoB.nombre()
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Datos – Leer fichero de texto

Se lee en 3 pasos

1. Abrir (open)
2. Leer (read, readlines, readline)
3. Cerrar (close)

Abrir

handler = open(filename[, mode[, bufsize]])

Abre un archivo y nos proporciona el manejador para poder utilizarlo

Parámetros:

- Nombre del archivo → Nombre del archivo a abrir
- Tipo de acceso → r (lectura), w (escritura), a (añadir). Si se les añade + permite leer y escribir
  b (binario)
- Buffer → 0 (unbuffered), 1 (buffered), n (tamaño del buffer en bytes)

Cerrar

handler.close();

Cierra un archivo abierto

Datos – Leer fichero de texto

Leer

- **read(n):** Lee n bytes, por defecto lee el archivo entero.
- **readline():** Devuelve una cadena con una sola línea.
- **readlines():** Devuelve una lista con una cadena como elemento por cada línea del archivo.

Se lee el archivo secuencialmente hacia delante, línea a línea hasta que finaliza

Pedro López, 32
Ana Caña, 27
Luis Árias, 58
Sandra Pérez, 25

C:\archivo.txt

```
fh = open('c:\\archivo.txt','r');
contenido = fh.read();
print(contenido);
fh.close();
```

```
fh = open('c:\\archivo.txt','r');
contenido = fh.readlines()
print(contenido);
fh.close();
```

```
fh = open('archivo.txt') ;
while True:
    line = fh.readline();
    print(line);
    if line==":
        break;
fh.close();
```

```
fh = open('archivo.txt') ;
for contenido in fh:
    print(contenido);
fh.close();
```

Datos – Leer fichero de texto

Leer sin cerrar el fichero

Existe una solución para leer un fichero de texto sin preocuparnos de cerrarlo

With <expresión> as <variable>
<Código>

→ Cierra el archivo automáticamente al salir del bloque

Pedro López, 32
Ana Caña, 27
Luis Árias, 58
Sandra Pérez, 25

C:\archivo.txt

```
with open('c:\\archivo.txt') as fh:
    for contenido in fh:
        print(contenido);
```

Problema:
Hasta ahora no se han respetado los caractéres del idioma

Solución: Añadir lo necesario para decirle al programa la codificación del archivo

```
import codecs;
fh = open('archivo.txt', encoding='utf-8') ;
for contenido in fh:
    print(contenido);
fh.close();
```

```
import codecs;
with open('c:\\archivo.txt',encoding='utf-8') as fh:
    for contenido in fh:
        print(contenido);
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Datos – Leer fichero de texto

Crea una agenda.
Los datos dentro del CSV serán del tipo:

Alejandro Santos Martinez; Av Ordoño Segundo 3;987806054
Maria Lopez Santos; Av Padre Isla 127;987407754
Luas gurutxaga Perez; Calle del canto 9;912866898
…

Crea un programa q lo lea, lo muestre y nos permita gestionarlo. Puedes usar objetos, tuplas, lo que se te ocurra…

Datos – Escribir fichero de texto

Exactamente igual que para leer

1. Abrir (open)
2. Escribir Modo **x**, **r+**, **a**
3. Cerrar (close)

Escribir

- **write(variable):** Escribe el contenido de la variable
- **writelines(lista):** Escribe TODOS los elementos de una lista en el fichero

Pedro López, 32
Ana Caña, 27
Luis Árias, 58
Sandra Pérez, 25

```
fh = open('c:\\archivo.txt','a');
contenido = fh.write("Pedro López, 32\n");
contenido = fh.write("Ana Caña, 27\n");
contenido = fh.write("Luis Árias, 58\n");
contenido = fh.write("Sandra Pérez, 25\n");
fh.close();
```

```
Lista=[]
Lista.append("Pedro López, 32\n");
Lista.append("Ana Caña, 27\n");
Lista.append("Luis Árias, 58\n");
Lista.append("Sandra Pérez, 25\n");
fh = open('c:\\archivo.txt','a');
fh.writelines(Lista);
fh.close();
```

Datos – Leer CSV

Los archivos **CSV** son documentos en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma) y las filas por saltos de línea.

-- Wikipedia--

Es un estándar de facto para muchas aplicaciones a la hora de utilizar ficheros de datos

Tratamiento:

- Como fichero de texto normal
- Como fichero CSV (módulo específico)

```
Juan,Gómez Rofriguez,23
María,Lopez Ayala,19
Roberto,Catillo Díaz,40
```

C:\archivo.txt

```
fh = open('c:\\archivo.txt');
 for line in fh:
   linea = line.split(',') ;
   # procesar elementos:
   nombre = linea[0];
   apellido = linea[1] ;
    # etc, etc
   fh.close();
```

```
import csv;
f = open('d:\\python\\Nueva carpeta\\prueba.xml')
lns = csv.reader(f)
for line in lns:
   nom = line[0];
   ape = line[1];
   ed=line[2];
   print("Nom: "+nom+"\t Apel: "+ape+"\t Edad: "+ed);
f.close();
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Datos – Leer CSV

Crea una agenda.
Los datos dentro del CSV serán del tipo:

Alejandro Santos Martinez; Av Ordoño Segundo 3;987806054
Maria Lopez Santos; Av Padre Isla 127;987407754
Luas gurutxaga Perez; Calle del canto 9;912866898
…

Crea un programa q lo lea, lo muestre y nos permita gestionarlo. Puedes usar objetos, tuplas, lo que se te ocurra…

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Datos – Leer CSV

Crea un archivo de texto que contenga nombres de archivos (con ruta completa)

Has de leer ese archivo línea a línea y crear las estructuras necesarias (Objetos) para almacenar en memoria los datos y después mostrarlo por pantalla.

Los archivos seran:

agenda.csv → Contendra los datos de una agenda

red.csv → Cada linea tendra nombre de ordenador, ip, mascara de red, puerta de enlace.

Lista_compra.csv → Contendra la lista de la compra (cod. Producto, descripcion, unidades, precio sin iva, iva aplicable, precio con iva).

Datos – Leer CSV

Descargad el siguiente archivo

https://bit.ly/2WGmlFg

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Datos – Leer CSV

CSV con un delimitador distinto

Los ficheros CSV pueden estar delimitados por otro carácter distinto a la **,**

; | -

Se determina el carácter mediante la parámetro **delimiter**

items.csv

```
pen,cup,bottle
chair,book,tablet
```

```
Pen
Cup
Bottle
Chair
Book
tablet
```

```
import csv;
f = open('items.csv', 'r');
with f:
    reader = csv.reader(f);
    for row in reader:
        for e in row:
            print(e)
```

```
import csv;
f = open('items.csv', 'r');
with f:
    reader = csv.reader(f, delimiter="|");
    for row in reader:
        for e in row:
            print(e)
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Datos – Escribir CSV

Además de abrir el fichero en modo escritura, hay que decirle **cómo** ha de escribir.

```
import csv;
csvsalida = open('<ruta>\\salida.csv', 'w',newline='')
salida = csv.writer(csvsalida,delimiter='|', quotechar='"')
salida.writerows(["hola","que","tal"])
salida.writerow(["Yo","bien","y tu?"])
csvsalida.close()
```

csv.QUOTE_ALL

csv.QUOTE_MINIMAL

csv.QUOTE_NONNUMERIC

csv.QUOTE_NONE

```
csv.writer(csvfile, delimiter=' ', quotechar='|', quoting=csv.QUOTE_MINIMAL)
```

Datos – Leer CSV

Ejercicios

Leer el contenido del archivo “estudiantes.csv" y a partir de sus datos realizar las siguientes actividades:

- listar los nombres completos de todas las personas de las cuales también se posee el email
- listar los nombres (no los apellidos). No se deben imprimir cada nombre más de una sola vez.

<lista>.sort() ó <lista>.sort(reverse=True)

- listar los nombres completos de las personas que se graduaron antes del año 2005
- listar el nombre de la escuela de todas las personas que se graduaron entre el año 2000 y el 2010. Cada escuela debe aparecer una sola vez. Indicar si alguna escuela no quedó incluida en el listado.

Completa el contenido del archivo “estudiantes.csv“ con los datos de tus compañeros. Se hará de la siguiente manera:

1. Pedirás los datos por teclado
2. Lo añadirás al listado en memoria
3. Escribirás todo en un fichero nuevo.

Datos – Leer XML

XML (**L**enguaje de **M**arcas **E**xtensible) es un meta-lenguaje que permite definir lenguajes de marcas utilizado para almacenar datos en forma legible. Permite definir la gramática de lenguajes específicos para estructurar documentos grandes mediante SGML

-- Wikipedia--

Supondremos siempre que un fichero XML está siempre bien formado

```
<?xml version="1.0" encoding="iso-8859-1"?>
<agenda_personal>
<codigo_referencia fecha="20100201">
<texto>Texto en el día 01/02/2010.</texto>
</codigo_referencia>
<codigo_referencia fecha="20100215">
<texto>Texto en el día 15/02/2010.</texto>
</codigo_referencia>
</agenda_personal>
```

C:\agenda.xml

Leer y mostrar el documento XML

```
from xml.dom import minidom

# Cargamos en objeto arbol_dom el documento xml
arbol_dom = minidom.parse('c:\\agenda.xml');

# Mostramos por pantalla el fichero XML, tal cual.
print (arbol_dom.toxml());
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Datos – Leer XML

Un fichero XML puede verse como un árbol de nodos por el cual hemos de movernos para tratar su información.

Los ficheros que se han de usar son ficheros bien formados.
Se leen utlizando varios paquetes de Python, pero nosotros usaremos **minidom**:

from xml.dom import minidom;

Los métodos que vamos a usar son:

- **parse** → Lee un fichero XML de una sola vez
- **childNodes** →Lista de nodos contenidos en el nodo tratado
- **firstChild** → Primer hijo del nodo tratado (**firstChild.data** → Devuelve el dato que tiene el nodo)
- **getElementsByTagName** → Devuelve una **lista** con todos los elementos que tienen esa etiqueta
- **getElementsByName** → Devuelve una **lista** con todos los elementos que tienen ese nombre
- **getAttribute** → Devuelve el valor de un atributo de un objeto

Datos – Leer XML

```
<?xml version="1.0" encoding="utf-8"?>
<objeto nombre="Mi objeto XML">
        Información del Objeto XML
</objeto>
```

tagName	objeto
Atributos	Nombre: Mi objetoXML
data	Información del Objeto XML

```
from xml.dom import minidom
arbol_dom = minidom.parse('C:\\Python\\e1.xml');

lista = arbol_dom.getElementsByTagName("objeto");
nodo=lista[0]
print(nodo.tagName);
print(nodo.getAttribute("nombre"));
print(nodo.firstChild.data);
```

Datos – Leer XML

```
<? xml version="1.0" encoding="utf-8"?>
<objeto nombre="Mi objeto XML">
  Información del Objeto XML
  <lista nombre="Elemento Lista 1">
    dato
  </lista>
  <lista nombre="Elemento Lista 2">
    dato 2
  </lista>
</objeto>
```

tagName	objeto
Atributos	Nombre: Mi objetoXML
data	Información del Objeto XML

tagName	lista
Atributos	Nombre: Elemento Lista 1
data	dato

tagName	lista
Atributos	Nombre: Elemento Lista 2
data	dato 2

```
from xml.dom import minidom
arbol_dom = minidom.parse('C:\\Python\\e2.xml');

lista = arbol_dom.getElementsByTagName("objeto");

for obj in lista:
  print(obj.tagName);
  print(obj.getAttribute("nombre"));
  print("Tiene hijos:"+str(obj.hasChildNodes()));
  objetos=obj.getElementsByTagName("lista");
  for l in objetos:
     print("Tagname: "+l.tagName);
     print("Atributo: "+l.getAttribute("nombre"));
     print("Infor: "+l.firstChild.data);
```

Datos – Leer XML
Ejercicio
Leer y mostrar el siguiente fichero

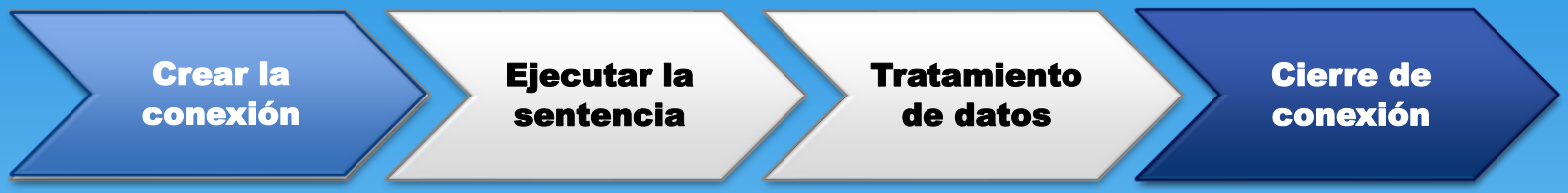
```
<?xml version="1.0" encoding="utf-8"?>
<bookstore name="Libreria Pastor">
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>
     <writer>Giada De Laurentiis</writer>
     <resumer>Pepe Lopez</resumer>
  </author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>
     <writer>J K. Rowling</writer>
     <resumer>Ana Martinez</resumer>
  </author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="PROGRAMMING">
  <title lang="en">Python for All</title>
  <author>
     <writer>M.L. Jobs</writer>
     <resumer>Delton Jones</resumer>
  </author>
  <year>2015</year>
  <price>39.99</price>
</book>
</bookstore>
```

Datos – Leer XML
Ejercicio
Leer y mostrar el siguiente fichero

```
<factura n_fac=f999>
  <datos_empresa>
    <nombre>Equipos Digitales S.L.</nombre>
    <dir>Av. Valladolid</dir>
    <poblacion cod_postal="28043" Provincia="Madrid">Madrid<población/>
    <cif>Q-9876543<cif/>
    <telefono/>
  </datos_empresa>
  <datos_cliente n_cli="c879">
    <nombre>Darío, Bueno Gutiérrez</nombre>
    <dir_env>Av. Oporto nº7 4ºd</dir_env>
    <poblacion cod_postal=28043>Madrid</poblacion>
    <provincia>Madrid</provincia>
  </datos_cliente>
  <datos_factura n_ped="p731" iva="16" f_pago= "efectivo" moneda="euro">
    <fecha>12-01-2005</fecha>
    <linea>
      <ref>MII93000F/8</ref>
      <desc/>MICRO PENTIUM IV 3000MHZ FB800</desc>
      <cant>1</cant>
      <precio>230</precio>
      <importe>266,80</importe>
    </linea>
    <base>970,00</base>
    <cuota iva>155,20</cuota_iva>
    <total>1125,20</total>
  </datos_factura>
</factura>
```

Acceso a MySQL

Todo se utiliza importando el paquete MySQLdb



Crear la conexión → **connect()** crea la conexión con el servidor MySQL y devuelve un objeto de la clase MySQLConnection

Cerrar la conexión → **close()** cierra la conexión con el servidor MySQL

```
import MySQLdb

cnx = MySQLdb.connect ('IP Servidor','usuario', 'contraseña', 'BBDD')
cnx = MySQLdb.connect (host='IP Servidor',user='usuario',
                       passwd='contraseña', db='BBDD')

cnx.close()
```

IES SAN ANDRÉS
Junta de Castilla y León
GOBIERNO DE ESPAÑA
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
sge

Acceso a BBDD

Para leer información se usan los cursores

- Crear el cursor → cursor()
- Ejecutar la sentencia SQL → execute(SQL)
- Tratar la información → fetchone() , fetchmany() or fetchall()
- Cerrar el cursor → close()

fetchone() → Devuelve la siguiente fila del cursor en fora de tupla o None
fetchmany(n) → Devuelve el número n de filas del cursor en una lista de tuplas.
También puede devolver una lsita vacía si no hay nada a devolver.
Se puede consultar el valor del argumento size para saber cuantas filas ha devuelto.
fetchall() → Devuelve todas las tuplas que faltan por traer como una lista de tuplas

IMPORTANTE: Commit y Rollback

Acceso a MySQL

Ejemplo

```
import MySQLdb

try:
    conn = MySQLdb.connect(user='root', password='',
                           host='127.0.0.1',database='BBDD')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM books")
    row = cursor.fetchone()
    while row is not None:
        print(row)
        row = cursor.fetchone()
except MySQLdb.Error as e:
    print(e)
finally:
    cursor.close()
    conn.close()
```

Acceso a MySQL

Ejemplo

```
import MySQLdb

try:
        conn = MySQLdb.connect('127.0.0.1','root','','test')
        cursor = conn.cursor()
        sql="CREATE TABLE persona (
                dni varchar(9) NOT NULL PRiMARY KEY,
                Nombre varchar(25) NOT NULL,
                Apellidos varchar(50) NOT NULL,
                Edad int(11) NOT NULL);"

        cursor.execute(sql)

except MySQLdb.Error as e:
        print(e)
finally:
        cursor.close()
        conn.close()
```

Acceso a MySQL

```
import MySQLdb

try:
        conn = MySQLdb.connect(user='root', password='',
                                host='127.0.0.1',database='test')
        cursor = conn.cursor()

        sql="INSERT INTO persona (dni, Nombre, Apellidos, Edad) VALUE
        ('11q', '1', '1', 1), ('12q', '12', '2', 2), ('123q', '12', '2', 2)"

        cursor.execute(sql)
except MySQLdb.Error as e:
        print(e)
 finally:
        cursor.close()
        conn.close()
```

Acceso a MySQL

Commit

y

rollback

Acceso a MySQL

```
import MySQLdb

try:
        conn = MySQLdb.connect("127.0.0.1","root","","BBDD")
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM books")
        row = cursor.fetchone()
        while row is not None:
            print("Titulo: "+row[2]+" || "+"Autor: "+row[1]+" || "+"Editorial: "+row[3])
            row = cursor.fetchone()
    except MySQLdb.Error as e:
        print(e)
    finally:
        cursor.close()
        conn.close()
```

ID	Autor	Título	Editorial
…	…	…	…
765	Stephen King	It	Planeta
…	…	…	…

Acceso a MySQL

Trabajar con los nombres de los campos de las tablas

Definir **cursor** como diccionario

```
import MySQLdb

try:
        conn = MySQLdb.connect("127.0.0.1","root","","BBDD")
        cursor = conn.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute("SELECT * FROM books")
        row = cursor.fetchone()
        while row is not None:
            print("Titulo: "+row['Titulo']+" || "+"Autor: "+row['Autor']+" || "+"Editorial: "+row['Editorial'])
            row = cursor.fetchone()    except MySQLdb.Error as e:
        print(e)
    finally:
        cursor.close()
        conn.close()
```

SOLO EN **MYSQL**

ID	Autor	Título	Editorial
…	…	…	…
765	Stephen King	It	Planeta
…	…	…	…

Acceso a Postgress

El acceso a datos en Postgres sigue el modelo estándar:

```
import psycopg2
try:
    conn = psycopg2.connect("dbname='template1' user='dbuser' host='localhost' password='dbpass'")

except:
   print ("No se puede conectar a la base de datos")
```

Acceso a Postgres

```
import psycopg2

try:
        conn = psycopg2.connect("dbname='BBDD' user='postgres'
host='127.0.0.1' password='123'")
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM books")
        row = cursor.fetchone()
        while row is not None:
            print(row)
            row = cursor.fetchone()
except:
        print "I am unable to connect to the database"
finally:
        cursor.close()
        conn.close()
```