**Introducción Python**

Se trata de un lenguaje interpretado, relativamente nuevo, y pese a su simpleza es muy potente.

Una característica importante de Python es que los Strings son indexables (String = String[]).

Dichas Strings son INMUTABLES, es decir, no son reasignables, sólo es posible reasignar su variable.

Todas las secuencias se numeran desde el CERO. Admite índices negativos, en cuyo caso -1 es el último carácter de la cadena.

Para sacar un segmento o substring se usa el slicing, que sigue el principio de Inclusión-Exclusión

*El número más pequeño ha de ir siempre a la izquierda, en caso contrario se devuelve una cadena vacía.*

*Si no se especifica un límite inferior o superior se considera que se toma desde el principio o*

Una lista representa una variable estructurada indexable de N elementos, sin tamaño fijo. Lo que viene a funcionar como un Array+List de Java pero más chulo. Es mutable y multitipo.

Se puede usar el mismo algoritmo de los contadores y acumuladores a las cadenas para ir acumulando caracteres:

* c = c + 1 (c +=1 )
* a = a + t (a += t)
* str = str + con (str += con)

Dentro de las funciones podemos incluir un docstring, que es un comentario que sirve para explicar qué es lo que hace a función (tipo Javadoc pero dentro de la función, no fuera). Además Python cuenta con un sistema de autodocumentación que estudia los docstrings de las funciones.

**Normas de Estilo**

* Los nombres de variables, funciones, clases, etcétera han de ser significativos (las ofuscaciones que las haga algún programita automáticamente después).
* Los nombres de variables empiezan siempre con minúsculas y se pueden separar usando CamelCase o guión bajo.
* Como las variables privadas no existen, su nombre se inicia con un guión bajo para no usarlas cuando no toque.
* Los nombres de las clases empezarán con mayúsculas y se separará mediante CamelCase.
* Los nombres de constantes estarán escritos en mayúsculas y se separarán con guión bajo.
* Para el sistema de indexación vamos a (muy a mi pesar) mantener el uso de 4 espacios, habría que asegurarse de que el editor de texto tiene dicha configuración.
* Cuando escribamos una función que devuelva un valor, el código de la función debe ser estrictamente el necesario para calcular el valor a devolver. Nada más. NO deben realizar ninguna petición de entrada ni de salida.
* Los nombres de los identificadores de funciones se escribirán en minúsculas, separando las palabras con guiones bajos. Al igual que los nombres de los parámetros. Han de ser significativos.
* La definición de las funciones se hará después de la inicialización de variables.
* La longitud de las líneas de los comentarios ha de ser de unos 80 caracteres.
* Hay que separar las declaraciones de funciones y clases con saltos de línea.
* Los comentarios de final de línea son convenientes, pero sin abusar.
* Usar un blanco delante y detrás de los operadores.
* La codificación de caracteres en Python ha de ser en UTF-8.
* [PEPs - Python Enhancement Proposals](https://www.python.org/dev/peps/)
* No poner espacios alrededor de los índices ni los parámetros.
* Poner un blanco después de una coma en las listas.
* Siempre que vayamos a recorrer una lista con un for, usaremos una copia de dicha lista usando bien el método *copy()* on*[:]*.
* Los programas han de estar Factorizados. Se llama factorizar a definir diversas funciones que realicen partes de trabajo de un código sustituyéndolo luego en éste. Mejor tener muchas funciones que un código espagueti. Gracias a esto no sólo podemos leer y depurar el código más fácilmente, sino que nuestro código será reutilizable.
* Evitar *prints* de variables muy largas.
* Los *import* irán debajo de la documentación inicial, al principio del archivo.

**Sentencias de Control de Flujo**

Son sentencias que cambian la ejecución secuencial del programa, permitiendo seguir otros "caminos" a la hora de ejecutar.

No tienen delimitador sintáctico (no hay {} como en otros lenguajes), funciona por indexaciones (como los .yml).

if <condition>:

code  
 code

elif <condition>:

code  
 code  
else:   
 code  
 code

while <condition>:

code   
 code

for <elem> in <list>:

code  
 code

pass

**Listas**

* lista.append(Object) Agregar un objeto al final de la lista
* lista.extend(List) Concatena una lista a la primera
* lista.insert(i, x) Insertar el objeto *x* en la posición *i*
* lista.remove(x) Elimina la primera ocurrencia de *x* en la lista
* lista.pop() Elimina y devuelve el último elemento de la lista
* lista.pop(i) Elimina y devuelve el elemento de la posición *i*
* lista.clear() Limpia la lista, dejándola vacía.
* lista.index(x) Devuelve el índice de la primera ocurrencia de *x*
* lista.count(x) Devuelve el número de veces que aparece *x*
* lista.sort() Ordena una lista siguiendo su orden natural
* lista.reverse() Invierte el orden de los elementos de la lista
* lista.copy() Devuelve una nueva lista copia de la primera

Glosario de Funciones

* print(String) Imprimir la String en pantalla
* input('prompt') Capturar el String del teclado, estaría bien asignarlo
* type(Object) Devuelve el tipo del Objeto (literal, variable o expresión)
* len(Object[]) Devuelve el tamaño de un Array (String o List)
* literal(Object) Convertir a tipo de literal
  + int(Object)
  + float(Object)
  + str(Object)
  + bool(Object)
  + list(Object)
* abs(Float) Toma el valor absoluto del float o integer introducido
* range(Integer) Crea una lista de N elementos comprendidos entre 0 y N-1
  + range(n, i) Crea una lista de I-N elementos comprendidos entre N e I-1
  + range(n, i, t) Lista entre N e I-1 saltando elementos de T en T
* *ord(Char) Devuelve el código ASCII de un carácter.*
* *chr(Integer) Devuelve el carácter según un código ASCII.*
* *reversed(List) Devuelve un iterableinvertido de la lista introducida.*
* *help(Function) Nos da una descripción de la documentación interna*
* *del(Object) Destruye cualquier variable en cualquier contexto*

**Listas por comprensión**

Listas por comprensión:

lista = [(expresión) for…. if...if]

con dos for.

El efecto que tiene es un bucle for dentro de un bucle for. El derecho dentro del izquerdo.

For a

for b

podemos importar de un módulo a otro, con una función import.

podemos import <nombre del modulo>

o puedo importar una función de un modulo.

from (nombremodulo) import (nombrefunción).

Hay un módulo llamado Random que tiene todas las funciones aleatorias.

* randint genera números aleatorios entre un número inicial y otro final.
* El módulo random contiene todas las funcionalidades para generar números al azar.
* Cuando se hace una importación, bien de un módulo o bien de una función se genera los nombres reconocibles.

* Función help(): nos da una descripción de la documentación interna de cualquier módulo o función
* del(): Borra una variable. Cualquier variable en cualquier contexto. Cuando hacemos un del se destruye to

25-10-2016

Tuplas (tuple)

Son un nuevo tipo de secuencias es una sucesión de elementos separados por comas, usualmente van entre paréntesis pero pueden usarse sin paréntesis depende del contexto

Las tuplas son ***INMUTABLES***  ocupan menos espacio que las listas y son más rápidas en la ejecución, son multitipo.

Permiten indexación y segmentación.

Una tupla permite tener elementos mutables.

con una lista en la pos 6

t[6][1]=3

Cambia el elemento 1 de la lista que está dentro.

para crear una tupla de un elemento hay que poner una coma detrás para que no se confunda con un número o una variable.

t=(1,)

Una lista vacía es l =[]

una tupla vacía es t=()

el nombre del tipo me crea una tupla vacía.

t= tuple()

No emplear nunca el nombre de la función como variable.

Las tuplas se pueden desempaquetar.

python permite asignación múltiple de variables en una sentecia mientras el desempaquetamiento de tuplas

t= 1,2,3

a,b,c=t

Permite hacer un intercambio en una sola sentencia.

Permite el paso de parámetros en números variables

También se utiliza una tupla como si fuera un registro (será una combinación de varios tipos)

Importar la función sqrt del módulo math

Def nombre (a,b=0)

No se tiene en cuenta el parámetro que tenga 0

si tu pones nombre(10), el parámetro b toma valor 0

Si el parámetro por valor es un objeto mutable, pues sida.

En general es recomendable usar parámetros por defecto solo con constantes inmutables.

Otra posibilidad con parámetros es utilizar palabras clave como parámetros, estos parámetros tienen la forma de la llamada de función

f(parametro=valor)

Si utilizo parámetro clave el orden del parámetro no es significativo. Python me permite cambiarlo.

No está permitido asignar 2 valores a un mismo parámetro.

Permite crear funciones con un número variable de parámetros

def f(a,b,\*args):

Esto indica que tiene 2 parámetros obligatorios pero se pueden añadir más

f(1,2,3,4) con el resto construye una tupla y se añade al \*args y desempaqueta la tupla

solo se utiliza para desempaquetar una tupla, es un operador unario

El rango se puede cambiar en tiempo de ejecución

SET

tipo set representa el concepto de conjunto en matemáticas. La característica de los conjuntos es que no son ordenados y no pueden estar repetidos.

se pone set ().

El set permite operaciones clásicas entre conjuntos

| unión

& interseccion hace los elementos de A y de b

{}

EL len funciona con conjuntos también

**Diccionarios.**

Es una estructura similar a una lista pero en vez de accederse por posición se accede por clave.

Un diccionario es una serie de claves asociados a un valor equivalente a índice - valor que son las listas.

Una clave puede ser cualquier valor inmutable.

Un valor puede ser de cualquier tipo

dic=dict() Diccionario vacío.

dic={}

dic[clave]

dic[‘b’]=25

Las claves no están ordenadas.

Si se reasigna con la misma clave lo machaca

Métodos string:

join(...)

| S.join(iterable) -> str

|

| Return a string which is the concatenation of the strings in the

| iterable. The separator between elements is S.

split(...)

| S.split(sep=None, maxsplit=-1) -> list of strings

|

| Return a list of the words in S, using sep as the

| delimiter string. If maxsplit is given, at most maxsplit

| splits are done. If sep is not specified or is None, any

| whitespace string is a separator and empty strings are

| removed from the result.

Los diccionarios permiten definiciones por comprensión, al igual que las listas.

Tienen la misma sintaxis solo que puestos entre llaves.

dc={exp.k:exp.v for for if if }

**SORTED**

D.keys()

D.Values()

D.item() Por orden alfabético.

los items devuelve los pares del diccionario

**ZIP**

devuelve una tupla formada por elementos correlativos de 2 o más secuencias.

Si no fueran secuencias de la misma longitud

el zip sirve para iterar con un bucle for.

**TORTUGA**

register\_shape añade la forma a la tortuga

Eventos: Son mierdas que vienen del sistema operativo y lo detecta el programa

La programación orientada a eventos, el flujo no es lineal

El evento tiene que tener definido coordenada x y coordenada y

Diferenciar eventos sobre la tortuga de eventos sobre la pantalla

El onclick es para pantalla y para tortuga, pero para la pantalla se puede llamar onscreenclick

si yo pongo import modulo se reconocen las funciones como modulo.f1()

ESTO ES LO QUE EXPLICÓ HACE 2 DÍAS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Si pones un import el import se ejecuta la primera vez.

si no queremos que se ejecute ese código lo identificamos como nuestro módulo.

Otro uso de ese código es un código de testeo (el cod de abajo)

if \_\_name\_\_ == ‘\_\_main\_\_’:

dir devuelve los nombres activos en el directorio actual

Un paquete es un conjunto de módulos, y un paquete en el sistema es equivalente a un directorio que contiene módulos y contiene también un fichero especial que es el \_\_init\_\_.py, un directorio que lo contenga es un paquete el init puede contener código de iniciaciíon del paquete ese fichero se ejecutará la primera vez que se importe algún módulo del paquete

**EXCEPCIONES**

Una excepción es un error en tiempo de ejecución. Es el error que te sale. Están predefinidas como clases en python aunque también se pueden definir por nosotros. Cada vez que ocurre una excepción se levanta algo, esto que se dispara permite detectar estas excepciones anómalas y tratar de corregirlas en tiempo de ejecución. Python permite separar el código que queremos proteger y declarando el código que hay que hacer cuando ocurre esta excepción.

Si salta el error en el try se abandona el código y se va al except

try:

código

except ZeroDivisionError:

código

La cláusula else se pone para poner lo que se ejecuta en caso de que sea mentira

Los argumentos son lo de por ejemplo index out of range

**Formateado de texto**

* *str(obj)*
* *repr(obj) → Devuelve una representación del objeto*
* *string.rjust(int)* → Concatena un texto justificando a la derecha
* *string.ljust(int)*  → Concatena un texto justificando a la izquierda
* *string.center(int)* → Concatena un texto centrado
* *string.zfill(int)* → Rellena con ceros a la izquierda
* *string.format(v1,v2)* → Mete las variables 1 y 2 dentro del string, si no ponemos nada los mete en orden ….{}{} es la forma del format

**Lenguajes OOP → Object-Oriented Programming**

¿Qué es un objeto?

Lo entenderemos como la representación de un objeto real en un programa.

Un objeto tiene un estado, el cual será definido por el valor de sus atributos.

Entenderemos por clase en Python como una plantilla de construcción de objetos, desde la cual se especificarán los atributos, bien sean de datos o de comportamiento *(métodos)*

class Clase:

pass

Una instancia es un objeto de esa clase. Es decir que de una clase yo puedo construir muchas instancias. Normalmente cada objeto tiene los mismos atributos que su clase (Clase y tipo son lo mismo)

PROGRAMACIÓN ORIENTADA A OBJETOS

Un objeto en software es una representación de un objeto en el mundo real. Tiene una identidad (existe) y tiene un estado (definido por el valor de sus variables). Puede ser referenciado por muchos nombres, alias.

Una clase es una plantilla para construir objetos. Tiene una serie de atributos (variables) y además tienen métodos. Los atributos pueden ser de dos tipos:

- Atributos de Datos.

- Atributos de Comportamiento o métodos: definen acciones sobre ese objeto.

Una clase se define como:

class Clase: El nombre de las clases siempre en mayúscula.

Una instancia es un objeto de esa clase. De una clase yo puedo construir muchas instancias. Normalmente cada objeto tiene los mismos atributos que su clase. Tipo y clase es lo mismo.

x=Persona ()

crea una variable con la instancia de la clase. Esta acción se llama instanciar, que es crear una instancia de una clase determinada.

Se puede añadir directamente una variable a un objeto de una clase directamente con la notación p.x=1 (nombre del objeto . nombre del atributo)

Unos objetos pueden tener más variables que otros.

Las variable de instancia que tendrá esa clase será como self.variable

self es una referencia al propio objeto

self.nombre significa que cuando cree una clase persona va a tener un atributo que se va a llamar nombre. Kis nétodos se definen como funciones dentro de la clase.

El primer atributo que se hace es

def \_\_init\_\_(self,...)

barra baja significa que es un método especial del lenguaje.

Es normal que las clases que creemos tengan un método de iniciación. Ese método tiene como tarea inicializar el estado de las instancias. (Crear las variables de instancias de las clases y para darles valores.

Como convenio el primer parámetro será self en el \_\_init\_\_

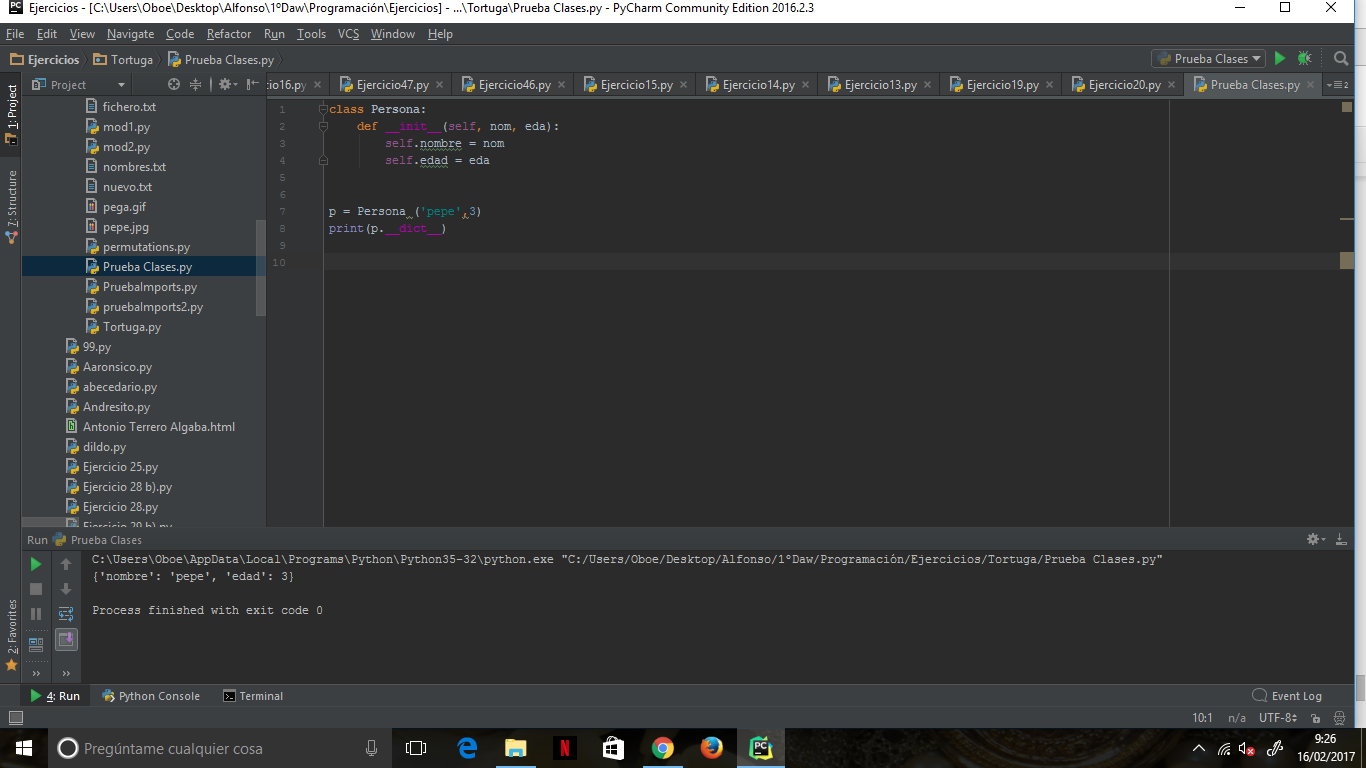
Self es una referencia a la misma instancia y se usará como primer parámetro.

def \_\_init\_\_ (self, x,y)

p= Persona (3,5)

p=self x=3 y=5

el \_\_dict\_\_ es como guarda el sistema los valores de la instancia



Los objetos no tienen nombre, se hacen referencias.

Un atributo de clase es aquel que es igual para todas las instancias de esa clase, se define no poniendo SELF al definirlo.Puede ocurrir un atributo de instancia oculte a un atributo de clase.

No se ponen nunca variables de clase ( Dentro de la clase pones x = 1)

Herencias

Es una característica de todos los lenguajes orientados a objetos que permite reutilizar y redefinir los atributos de otra clase.

class ClaseDerivada(ClaseBase)

La clase derivada hace que tenga todos los atributos de la clase base

El intérprete buscará un atributo primero en la clase derivada, si no lo encuentra, lo buscará en la clase padre. Buscándolo de izquierda a derecha

object → Es el objeto básico del que derivan todos los objetos de Python

Si yo pongo un método que ya existe en la clase padre.

**PROYECTO**

Que no haya nada que no hayamos dado en clase

y yo que se

sida

unos marciano

**ITERABLES**

Una clase (colección) es iterable cuando puede ser recorrida elemento por elemento (se le puede aplicar un for)

Un protocolo es un conjunto de métodos que contiene una clase.

El protocolo iterator dice que una clase que verifique eso tiene que tener 2 métodos especiales:

* \_\_next\_\_()
* \_\_iter\_\_()

Son métodos especiales para el intérprete

El for va a llamar a los métodos especiales, si el método next devuelve un objeto, el for lo captura en la i

El método iter debe devolver un iterador

**GENERADOR**

Un generador se hace con un yield valor (lo mismo que return valor)

Una función se reinicia cada vez que la llamas.

Un generador no, cuando lo llama el next (porque tiene un next interno) guarda el estado interno de las variables de la función

**EXPRESIÓN GENERADORA**

Una expresión generadora…, además produce un generador producido de la misma forma que las listas por comprensión. Al no crear los objetos físicamente la ocupación en memoria es muy baja

Si empleamos una lista en lugar de un generador nos encontramos que la memoria del PC estará en uso contínuo, además del tiempo de creación de esta.

\_\_new\_\_() es el verdadero constructor de objetos

comparadores

\_\_eq\_\_ == (si implementas uno no hace falta el ne y viceversa)

\_\_ne\_\_ !=

\_\_gt\_\_ >

\_\_ge\_\_ >=

\_\_lt\_\_ <

\_\_le\_\_ <=

\_\_bool\_\_ Return false or true

ColeccionClientes

dic\_clientes: dict <dni,Cliente>

Obtener\_Cliente(dni):Cliente

cargar\_clientes

El controlador de BBDD funcionará como una fachada en nivel de persistencia

Queremos definir la clase coleccion cliente con la estructura de antes