# PYTHON

Profesor:

Diego Serrano

[diegojserrano@gmail.com](mailto:diegojserrano@gmail.com)

Clases:

Jueves 18:30 a 21:30

8 clases (18/6, 25/6, 2/7, 16/7, 23/7, 30/7, 6/8, 13/8)

Ejercicios:

<https://github.com/diegojserrano/ISSD_Python>

## Plan de clases

1. Introducción. Estructuras condicionales y repetitivas
2. Funciones
3. Estructuras de datos
4. Modularización
5. E/S. Excepciones
6. POO
7. POO
8. Biblioteca estándar

## Introducción. Estructuras condicionales y repetitivas

* Lenguaje interpretado (más lento que compilado).
* El intérprete se descarga. Es interactivo.
* Fácil de programar, tiempo de desarrollo productivo.

### Tipo de datos

No hay declaración explícita de tipo de datos.

* Boolean (True,False)
* Numéricos
  + Int
  + (long solo v2)
* Secuencias (iterables):
  + String (arreglo de caracteres)
  + List (Vector de tamaño dinámico)
  + Tupla (par ordenado de distinto tipos de datos)  
    t = “Juan”,35 🡪 t (‘Juan’,35)  
    nombre,edad=tupla🡪 desarma asignación
* Conjuntos:
  + Set (elementos sin orden, sin repetidos)
  + Frozen set (no permite modificaciones)
* Mapas
  + Dict: (Diccionario) Estructura de datos que vincula par desordenados de clave y valor.  
    Escribe los datos en memoria, y no recorre toda la tabla, sino que va directo.

|  |  |
| --- | --- |
| Clave | Valor |
|  |  |
|  |  |

Ejemplos:

4\*\*3 🡪 4 al cubo

Print(“Hola mundo”) 🡪 Hola mundo

Print(Hola son las”,8) 🡪 Hola son las 8

a=6

print(a) 🡪 6

a 🡪 6

a = ‘hola’

print(a) 🡪 hola

“hola”+” mundo” 🡪 ‘hola mundo’

3\*\*0.5 🡪 raíz cuadrada

“w”\*8 🡪 ‘wwwwwwww’

### Operadores aritméticos

* +
* -
* \*
* \*\* 🡪 Potencia

### Operadores de cadenas

* + 🡪 Une cadenas
* \*
* []
* [:] 🡪 Slice

Ejemplos:

Nombre=”Juan”  
nombre[2]🡪’a’

nombre [1:4]🡪’uan’

nombre [:2]🡪’Ju’

nombre[:]🡪 ‘Juan’

nombre [-2:]🡪’an’

nombre [:-2]🡪’Ju’

nombre [::2]🡪’Ja’ 🡪 Salteador

nombre [::-1]🡪’nauJ’ 🡪 Ordena al revés

### Operadores lógicos y de comparación

* 🡪Mayor
* >= 🡪 Mayor e igual
* == 🡪 Igual
* != 🡪 Distinto
* and
* or
* not

### Estructuras condicionales

#### if

Única instrucción if (no existe el switch case)

If *condicion1*:  
código  
 *instrucción* 🡪 Se debe dejar al menos un espacio en las instrucciones  
elif *condicion2*: 🡪 else if  
 *instrucción*  
 *instrucción*  
else:  
 *instrucción*  
*instrucción* 🡪 para terminar, se debe volver a la misma columna del if (sin espacio)

### Estructuras repetitivas

#### while

While *condición*:  
 *instrucción* 🡪 Se debe dejar al menos un espacio en las instrucciones  
else:  
 *instrucción* 🡪 Se ejecuta al final del ciclo, sólo si no corta por break.

#### For

For *variable* in *estructura*   
 *instrucción*

ejemplos  
for letra in nombre:  
 print(letra)  
🡪 J  
 u  
 a  
 n

For letra in “Hola":  
 print(letra,end=”,”)  
🡪H,o,l,a

For i in range(3) : print(i) 🡪 Con una instrucción se puede poner al lado  
🡪 1  
 2  
 3

For i in range (5,8): print(i)  
🡪 5  
 6  
 7  
 8

For i in range (5,10,2): print(i)  
🡪 5  
 7  
 9

### Otros comandos

#### Imprimir en pantalla

Print(*variable* o *valor*)

#### Saltos incondicionales:

* Break 🡪 interrumpe el ciclo
* continue 🡪 interrumpe la vuelta actual

#### Intercambio de variables

a,b=b,a 🡪 se le asigna el valor de *a* a *b*, y viceversa.

#### Entrada por consola

*variable*=input(“*mensaje*”) 🡪 Si recibe una instrucción de Python la debería ejecutar  
*variable* =int(input(“ “))  
*variable* =str(input(“ “))  
*variable* =float(input(“ “))

PEP 8 <https://www.python.org/dev/peps/pep-0008/>

## Funciones

Def 🡪 cabecera de la función

def *nombrefuncion*( *parámetrosformales* ):

* pass 🡪 indica un bloque vacío (por ejemplo, pongo un if sin saber aún que voy a poner de contenido, pongo pass)
* return 🡪 retorna el valor
* none 🡪 no retorna nada
* global 🡪 permite declarar variables globales  
  global i,j

### parámetros formales

* separados por coma:   
  def x1(a,b)
* pueden tener valores por defecto (generalmente constantes):   
  def x2(a=0,b=1)
* puede recibirse una lista variable de parámetros:  
  def x3(a,\*lista)  
  print(a,b,100,etc)🡪sería como def x(\*lista)
* puede recibirse un diccionario de parámetros:  
  def x4(a,\*\*dicc)

#### parámetros actuales

llamada normal 🡪 X1(3,5)

#### parámetros opcionales

llamada de un parámetro 🡪 X2(8)

**def** x(a=1,b=2):  
 print(a,b)  
x(3)

🡪 Reemplaza el primer parámetro (muestra 3 2)

x(b=3)

🡪 Reemplaza el parámetro asignado (muestra 1 3)

**def** suma(a=1,b=2):  
 print(a,end=**"----"**)  
 print(b,end=**"----"**)  
 **return** a+b  
s = suma(b=5,a=3)  
print(s)

🡪 Devuelve lo mostrado más lo calculado (muestra 3----5----8)

## Modularización

### Modulo

* Permite reutilizar funciones entre programas
* Consiste únicamente de un archivo de código

|  |  |  |
| --- | --- | --- |
| cálculos.py |  | programa.py |
| **def** f1:  ... **def** f2:  ... |  | **import** calculos calculos.f1()🡪 Llama directamente a la función  a=cálculos.f1()🡪 Devuelve el resultado de la función  f=cálculos.f1 🡪 Puntero de la función  b=f() |
|  |  | **from** calculos **import** \* a=f1()🡪 Corre riesgo de que ya se haya declarado la variable en la f. |

**import** sys  
print(sys.path)

🡪 Devuelve una lista de los directorios donde busca los módulos

sys.path.append(**"C:"**)

🡪 Busca en el directorio que le especifiquemos

Importar el paquete:

* Crear un archivo \_\_init\_\_.py (dos guiones bajos) en el directorio donde están los módulos a leer
* El archivo puede estar vacío
* De todos modos se deben importar los módulos

## Estructura de datos

### Secuencias

#### Cadenas

Lista inmutable de caracteres

n=**"Juan"**print(n[1])

🡪 u

#### Listas

* Lo más parecido a un arreglo
* No tiene limitaciones de tamaño ni tipo de datos

v = [1,2,4]

**for** a **in** v: 🡪 Recorre los valores de la secuencia  
 print(a)

**for** i **in** range(3): 🡪 Lo recorre como si fuese un vector  
 print(v[i])

u = [**'m'**,2]\*10 🡪 Multiplica el elemento  
**for** a **in** u:  
 print(a) 🡪 imprime ‘m’ y 2 diez veces

w1 = [1]\*5  
w2 = [2]\*5  
w = w1 + w2  
**for** a **in** w:  
 print(a)

v1=[1]\*5  
v2=[2]\*5  
v = v1 + v2  
v[3] = 6 🡪 reemplaza el valor al elemento de la posición 3  
v[::2] = [12,12,12,12,15] 🡪 de a par reemplaza los valores  
**for** a **in** v:  
 print(a)

Funciones de listas:

* Append: agrega un elemento al final

v.append()

* Pop: elimina el último elemento

v.pop()

Definición por comprensión

A 🡪 conjunto de nros pares

A = {X / X es par}

v2=[x **for** x **in** range(10)]  
**for** a **in** v2:  
 print(a)

🡪 imprime los nros del 0 al 9

v2=[x **for** x **in** range(10) **if** x%2==0] 🡪 Imprime del 0 al 9 los nros pares

#### Tuplas

* Par ordenado de elementos

Punto1= 1,1  
Punto2= 1.2,2  
x,y=punto2  
a=2  
B=3  
A,b=2,3  
x,y=y,x

#### Set

* Funcionan como listas que no aceptan repetidos
* Utilizan como sintaxis las llaves
* Guarda los datos en la tabla de hash 🡪 mezcla los elementos para garantizar búsquedas más rápidas para detectar la existencia o no de un elemento

Listas []  
Tuplas ()  
Set {}  
Diccionarios {:}

C1 = {3,76,7}  
**for** x **in** c1: print(x)

* Intersección del conjunto 🡪 interseccion = conj1 & conj2
* Diferencia 🡪 diferencia = conj1 - conj2
* Unión 🡪 union = conj1 | conj2
* Unión excluyente 🡪 xor = conj1 ^ conj2

conj1 = {x **for** x **in** range(9) **if** x % 2 == 0}  
conj2 = {x **for** x **in** range (3,8)}

#### Diccionarios

Asocia una tabla a datos

Datos atómicos

{3:”hola“,lista() : , : }

empleados = {1:**"Juan"**, 2:**"Pedro"**}  
print(empleados)  
  
*#indices***for** e **in** empleados:  
 print(e) 🡪 *1 y 2  
  
#valores***for** e **in** empleados.values():  
 print(e)🡪 *Juan y Pedro  
  
#testear si existe*print(1 **in** empleados) 🡪 *True  
  
#funciones*f = {x: x\*\*2 **for** x **in** range(4)}  
print(f)🡪 *{0: 0, 1: 1, 2: 4, 3: 9}*

*#testear si es menor*l1 = [1,3,6]  
l2 = [1,3,4]  
print(l1<l2)🡪 *False  
  
#testear si son iguales*l3 = [1,3,6,[5,7,4]]  
l4 = [1,3,6,[5,7,4]]  
print(l3==l4) 🡪 *True*

*#imprime el elemento con indice 2*  
print(empleados[2])🡪*#Pedro*

## E/S. Excepciones

### Lectura de archivos

#### Open

**open(nombre, modo)**

modos:

* r: lectura
* w: escritura
* a: (append) agregación
* rb: lectura (binaria)
* wb: escritura (binaria)
* ab: agregación (binaria)

#### Read

**readline**: lee una línea, hasta el retorno de carro

**rstrip()**: limpia los espacios del retorno de carro \n

Para saber si el archivo terminó, la línea viene vacía (sin retorno de carro)

f = open(**"numeros.txt"**)  
linea = f.readline()  
**while** linea != **""**:  
 print(linea.strip) *#le limpia los espacios de la punta* linea = f.readline()  
f.close()

Lee todo el archivo directamente:

f = open(**"numeros.txt"**)  
lista = f.readlines()  
print(lista)  
f.close()

Imprime los primeros 3 caracteres:

f = open(**"numeros.txt"**)  
texto = f.read(3)  
print(**"'"**,texto)  
f.close()

### Escritura de archivos

#### Write

**write()**: escribe sin retorno de carro \n

f = open(**"salida.txt"**,**"w"**)  
f.write(**"prueba"**)

**writelines():** escribe una línea

#### Format

**format():** parecido a printf, el formato es el contenido de la variable

f = open(**"salida.txt"**,**"w"**)  
hora=**"20:40"**codigo=56776  
mensaje=**"error de red"**a=**"[{0}] - {1} : {2}"**f.write(a.format(hora,codigo,mensaje))  
f.close()

🡪escribe [20:40] - 56776 : error de red

equipos = []  
equipos.append((**"Boca"**,23,15,1.456))  
equipos.append((**"RiBer"**,20,19,1.823))  
equipos.append((**"BELGRANO"**,26,11,1.325))  
equipos.append((**"Galleres"**,1,-14,0.534))  
  
formato=**"{0:10}: Puntos {1:3d} - Dif.goles {2:3d} - Prom {3:1.5f}\n"**f = open(**"salida.txt"**,**"w"**)  
  
**for** equipo **in** equipos:  
 f.write(formato.format(equipo[0],equipo[1],equipo[2],equipo[3]))  
f.close()

🡪escribe

Boca : Puntos 23 - Dif.goles 15 - Prom 1.45600  
RiBer : Puntos 20 - Dif.goles 19 - Prom 1.82300  
Belgrano : Puntos 26 - Dif.goles 11 - Prom 1.32500  
Galleres : Puntos 1 - Dif.goles -14 - Prom 0.53400

**Dumps:** recibe una lista y lo convierte a json

f= open(**"salida.txt"**,**"r"**)  
contenido = f.read()  
**import** json  
equipos = json.load(contenido)  
f.close()  
**for** e **in** equipos:  
 print(e[0])

## POO

Programación estructurada, secuencial, repetitiva

Compra-Venta

Comprador Comprador

Artículos Brecha semántica Artículos

Ventas Ventas

p

Nombre documento edad

Definición la clase. Datos de objetos: atributos

* Constructor

**class** Persona:  
 **def** \_\_init\_\_(self):  
 self.nombre=**""** self.documento=0  
 self.edad=0

* Iniciación

p=Persona()  
p.nombre=**"Juan"**p.documento=2223  
p.edad=14

* Anotaciones

@property  
**def** edad(self):  
 **return** self.\_\_edad 🡪 el doble guión bajo lo asume como privado

No lo hace privado del todo, sino que le agrega coso

p.\_\_Persona\_\_edad

**class** Persona:  
 **def** \_\_init\_\_(self,nombre,apellido,documento,edad):  
 self.nombre=nombre  
 self.apellido=apellidp  
 self.documento=documento  
 self.edad=edad  
 @property  
 **def** edad(self):  
 **return** self.\_\_edad  
  
 @edad.setter  
 **def** edad(self,valor):  
 **if** valor >= 0 **and** valor <= 120:  
 self.edad = valor  
  
  
 **def** \_\_str\_\_(self):  
 **return "Persona: nombre = "** + self.nombre + **" - documento = "** + str(self.documento) + **" - edad = "** + str(self.edad)  
 **def** nombreCompleto(self):  
 **return** self.nombre + **" "** + self.apellido  
  
**def** agregarPersona(lista):  
 p.nombre = str(input(**"Nombre: "**))  
 p.apellido = str(input(**"Apellido: "**))  
 p.documento=int(input(**"Documento:"**))  
 p.edad=int(input(**"Edad:"**))  
 lista.append(p)

## Biblioteca estándar

### Librerias

#### os

Módulo del Sistema Operativo

**import** os

* **getcwd**: imprime el directorio actual  
  print(os.getcwd())
* **chdir**: cambia el directorio  
  os.chdir(**"/datos/Ejercicios"**)
* **mkdir**: crea un directorio
* **getenv**: imprime las variables de entornoprint(os.getenv**("PATH"**))
* **getlogin**: conocer el usuario logueado
* **putenv**: modifica el valor del entorno
* **uname:** datos del sistema operativo (sólo Unix)

name = os.uname()  
**for** x **in** name:  
 print(x)

* **listdir**: imprime una lista del directorio.print(os.listdir())
* **walk**: imprime una tupla del directorio y subdirectorios.  
  A  
  ├ B  
  └ C  
  [(A,[B,C],[,,]) , (B,[],[]) , (C,[],[])] os.chdir(**".."**)  
  recorrido=os.walk(**"."**)  
  **for** d **in** recorrido:  
   directorio,carpetas,archivos=d  
   print(**"Directorio="**+directorio)  
   print(**"Carpetas="**+str(carpetas))  
   print(**"Archivos="**+str(archivos))  
   **for** a **in** archivos:  
   print(os.path.join(directorio,a))
* **path:** maneja los directorios
  + **join:** une (ejemplo de arriba)
  + **size**

#### platform

Módulo del Sistema Operativo Windows

**import** platform

* **uname:** datos del sistema operativo   
  print (platform.uname())

devuelve

uname\_result(system='Windows', node='g42-03', release='7', version='6.1.7600', machine='AMD64', processor='Intel64 Family 6 Model 60 Stepping 3, GenuineIntel')

#### shutil

* **make\_archive**: comprime un archivo.

shutil.make\_archive(**"Datos\curso"**,**"zip"**)

* copy\_file

#### glob

Listar archivos con \* y ?

**import** glob

* **glob:**

print(glob.glob(**"M\*.py"**))

#### sys

Parámetros de línea de comando

**import** sys

* **argv:**

print(sys.argv)

#### urllib

Librería para manejar web

**from** urllib.request **import** urlopen  
  
respuesta = urlopen(**"http://www.lavoz.com.ar"**)  
**for** l **in** respuesta:  
 print(l)

imprime todo el contenido html de la página

#### datetime

Fecha del sistema

**from** datetime **import** \*

* **date**: obtiene fechas
* **timedelta**: permite hacer operaciones con las fechas

**from** datetime **import** \*  
**from** datetime **import** date  
**from** datetime **import** timedelta  
hoy = datetime.now()  
print(hoy)  
otra=date(2010,1,1)  
print(otra)  
print(hoy+timedelta(days=5,hours=10))

#### MySQLdb

**import** MySQLdb  
*# Open database connection*db = MySQLdb.connect(**"localhost"**,**"testuser"**,**"test123"**,**"TESTDB"** )  
*# prepare a cursor object using cursor() method*cursor = db.cursor()  
*# execute SQL query using execute() method.*cursor.execute(**"SELECT VERSION()"**)  
*# Fetch a single row using fetchone() method.*data = cursor.fetchone()  
print **"Database version : %s "** % data  
*# Fetch multiple row.*todo = cursor.fetchall()  
**for** fila **in** todo:  
 print(fila[0],fila[1])  
*# commmit, rollback*db.rollback()  
db.commit()  
*# disconnect from server*db.close()

## Ejercicios:

### Editor

1. JetBrains/PyCharm 🡪 Editor de texto para Python
2. Create New Project/
3. (Click derecho sobre el proyecto)/New/Python File
4. Crear configuración (única vez):
5. Run/Edit Configurations…
6. +/Python
7. Agregar el script

Nota: El editor interpreta el texto comentado, por lo tanto no se deben agregar caracteres españoles (acentos y ñ) debido a que está en inglés.

### Ejercicios

#### Ingresar 5 números por teclado y mostrar el mayor

*#Ingresar 5 numeros por teclado y mostrar el mayor*mayor=0  
print(**"Ingrese 5 numeros"**)  
**for** i **in** range(5):  
 n=int(input())  
 **if** i == 0 **or** n > mayor:  
 mayor = n  
print(**"El mayor es"**,mayor)

#### The 3n + 1 problem

<https://uva.onlinejudge.org/external/1/100.html>

<http://www.programming-challenges.com/spanish/pdfs/110101.pdf>

*#The 3n + 1 problem  
  
#Seteo de variables*n=0  
m=0  
t=0  
v=0  
maxn=0  
maxm=0  
maxv=0  
  
*#Ingreso de comandos*print(**"Ingrese el valor para N:"**)  
n=int(input())  
**while** n<1 **or** n>999999:  
 **if** n<1:  
 print(**"El valor debe ser mayor a 0:"**)  
 **if** n>999999:  
 print(**"El valor debe ser menor a 1.000.000:"**)  
 n=int(input())  
print(**"Ingrese el valor para M:"**)  
m=int(input())  
**while** m<1 **or** m>999999 **or** m<=n:  
 **if** m<=n:  
 print(**"El valor de m debe ser mayor a n:"**)  
 **else**:  
 **if** m<1:  
 print(**"El valor debe ser mayor a 0:"**)  
 **if** m>999999:  
 print(**"El valor debe ser menor a 1.000.000:"**)  
 m=int(input())  
  
*#logica***for** i **in** range (n,m+1):  
 v = 1  
 t=i  
 **while** t != 1:  
 **if** t%2==0:  
 t=t/2  
 **else**:  
 t=t\*3+1  
 v = v + 1  
 **if** v > maxv:  
 maxv=v  
print(n,m,maxv)

#### Funciones

\_\_author\_\_ = **'gab4alto'  
def** longitud(x):  
 c=0  
 **while** x != 1:  
 **if** x%2 == 1:  
 x = x\*3+1  
 **else**:  
 x=x/2  
 c +=1  
 **return** c  
  
**def** longitudMaxima(i,j):  
 mayor = 0  
 **for** n **in** range(i,j+1):  
 c=longitud(n)  
 **if** c>mayor: mayor = c  
 **return** mayor  
  
**def** cargarDatos():  
 *#global i, j* i = int(input())  
 j = int(input())  
 **return** i, j  
  
datos = cargarDatos()  
i, j= datos  
**while** (i!=0) **and** (j!=0):  
 mayor = longitudMaxima(i,j)  
 print(mayor)  
 datos = cargarDatos()

(incompleto)

#### Programar una función que calcule números de Fibonacci en forma recursiva

Fn 🡪 n=1 🡪 1

🡪n=2 🡪 1

🡪 n>2 🡪 Fn-1 + Fn-2

*#1 1 2 3 5 8 13 21 34 55***def** fibonacci(n):  
 i=1  
 j=1  
 k=0  
 **while** i < n:  
 print(i)  
 k = j  
 j = i + j  
 i = k  
  
print(**"Ingrese el valor para N:"**)  
n=int(input())  
fibonacci(n)

*#0 1 2 3 4 5 6 7 8 9 10 11 12 13 14  
#1 1 2 3 5 8 13 21 34 55 89 144 233 377 610***def** fibonacci(n):  
 **if** n<2:  
 **return** 1  
 **return** fibonacci(n-1)+fibonacci(n-2)  
  
print(**"Ingrese el valor para N:"**)  
n=int(input())  
print(fibonacci(n))

#### Programar una función que busque el mayor de una lista y retorne el valor de su posición

*#Programar una funcion que busque el mayor de una lista y retorne el valor de su posicion***def** mayor(lista):  
 m=0  
 i=1  
 p=0  
 **for** j **in** lista:  
 **if** j > m:  
 p = i  
 m = j  
 i+=1  
 print(**"El mayor es:"**,m,**"Posicion:"**,p)  
  
l = []  
n=1  
**while** n > 0:  
 n = int(input(**"Ingrese un numero (0 - termina)"**))  
 l.append(n)  
mayor(l)

#### Calcular la distancia de un punto por Pitágoras

**import** math  
**def** distanciaAlOrigen(punto):  
 x = punto[0]  
 y = punto[1]  
 **return** math.sqrt(x\*\*2+y\*\*2)  
  
**def** distancia(punto1,punto2):  
 dx = punto1[0] - punto2[0]  
 dy = punto

(incompleto)

#### Hacer un programa que solicite los datos de los equipos de una liga de fútbol.

De cada equipo se ingresa:  
-Nombre  
-Puntos  
-Diferencia de gol.  
Informar el nombre del equipo que va primero, si corresponde los de los que posean el mismo puntaje.

equipo1 = **"Boca"**,23,18  
equipo2 = **"River"**,23,13  
equipo3 = **"Racing"**,22,12  
equipo4 = **"San Lorenzo"**,23,12  
equipos = [equipo1,equipo2,equipo3,equipo4]  
  
m=0  
**for** i **in** equipos:  
 **if** i[1]>m:  
 m=i[1]  
  
**for** i **in** equipos:  
 **if** i[1]==m:  
 print(i)

#### Hacer un programa que permita ingresar números en dos listas diferentes y a continuación muestre los números existentes en ambas

**def** iguales(lis1,lis2):  
 **for** i **in** lis1:  
 **for** j **in** lis2:  
 **if** i == j:  
 print(**"Esta en las dos listas:"**,j)  
  
lista1 = []  
n=1  
**while** n > 0:  
 n = int(input(**"Lista 1 - Ingrese un numero (0 - termina)"**))  
 **if** n > 0:  
 lista1.append(n)  
  
lista2 = []  
m=1  
**while** m > 0:  
 m = int(input(**"Lista 2 - Ingrese un numero (0 - termina)"**))  
 **if** m > 0:  
 lista2.append(m)  
iguales(lista1,lista2)

#### Modificar para que muestre los elementos que estan en el primero pero no en el segundo

*#modificar para que muestre los elementos que estan en el primero pero no en el segundo***def** cargarConj(conj):  
 cantidad = int(input(**"Ingrese cantidad:"**))  
 **for** i **in** range(cantidad):  
 conj.add(int(input(**"Ingrese un numero:"**)))  
  
conj1 = set()  
conj2 = set()  
  
cargarConj(conj1)  
cargarConj(conj2)  
  
interseccion=set()  
**for** a **in** conj1:  
 **if** a **not in** conj2:  
 interseccion.add(a)  
print(interseccion)

#### Ejercicio de archivos

* Abrir CP.txt.
* Cargar todo el contenido en una estructura de datos adecuada:
  + Lista de tuplas
  + Diccionario de tuplas
* Permitir consultar
  + Las localidades de cierto código
  + El código de alguna localidad

**def** leerCodigos():  
 archivo = open(**"CP.txt"**)  
 codigos = []  
  
 linea = archivo.readline()  
 **while** linea != **""**:  
 provincia = linea[0]  
 codigo = int(linea[2:6])  
 localidad = linea[7:].rstrip()  
 codigos.append((provincia,codigo,localidad))  
 linea = archivo.readline()  
  
 **return** codigos  
  
  
**def** buscarCP(lista, codigo):  
 salida = []  
 **for** c **in** lista:  
 **if** c[1] == codigo:  
 salida.append(c[2])  
  
 **return** salida  
  
  
**def** buscarLocalidad(lista, nombre):  
 salida = []  
 a = **""  
  
 for** c **in** lista:  
 **if** c[2].find(nombre) != -1:  
 salida.append(c)  
  
 **return** salida  
  
lista = leerCodigos()  
  
  
codigoBuscado = int(input(**"Ingrese el codigo a buscar"**))  
print(buscarCP(lista,codigoBuscado))  
  
localidad = input(**"Ingrese la localidad"**)  
print(buscarLocalidad(lista,localidad))

#### Modulo

Programar un módulo llamado funciones que incluya dos funciones sumar y restar. Programar un programa que use esas funciones

Funciones.py

**def** fibonacci(n):  
 **if** n<2:  
 **return** 1  
 **return** fibonacci(n-1)+fibonacci(n-2)

Programa.py

**import** Funciones  
print(**"Ingrese el valor para N:"**)  
n=int(input())  
print(Funciones.fibonacci(n))

#### Buscador de texto

Indexador

* Abre un archivo de texto
* Inserta todas las palabras en un índice (set)

Buscador

* Serialice el índice
* Deserialice el índice
* Ingrese por teclado una palabra e informe si está

**def** indexar(nombre):  
 archivo = open(nombre)  
 indice = list()  
  
 linea = archivo.readline()  
  
 **while** linea != **""**:  
 palabras = linea.split()  
 **for** palabra **in** palabras:  
 **if not** palabra **in** indice:  
 indice.append(palabra)  
 linea = archivo.readline()  
 archivo.close()  
  
 print(**"cantidad de palabras distintas: "**,len(indice))  
  
**def** guardarIndice(indice,nombre):  
 **import** json  
 archivo = open(nombre,**"w"**)  
 json.dumps(indice,archivo)  
 archivo.close()  
  
index = indexar(**"pg21231.txt"**)  
guardarIndice(index,**"pg21231.ind"**)

#### Crear la clase personas

**class** Persona:  
 **def** \_\_init\_\_(self):  
 self.nombre=**""** self.documento=0  
 self.edad=0  
  
**def** agregarPersona(lista):  
 p=Persona()  
 p.nombre = str(input(**"Nombre: "**))  
 p.documento=int(input(**"Documento:"**))  
 p.edad=int(input(**"Edad:"**))  
 lista.append(p)  
  
listaPersonas=[]  
n=int(input(**"ingrese la cantidad de personas: "**))  
**for** j **in** range(n):  
 agregarPersona(listaPersonas)  
**for** i **in** listaPersonas:  
 print(**"Nombre:"**,i.nombre)  
 print(**"Documento:"**,i.documento)  
 print(**"Edad:"**,i.edad)

#### Programar una clase Equipo que represente a un equipo de futbol durante un campeonato con forma de liga. De cada equipo se deben mantener los siguientes datos:

* Nombre
* Cant. De partidos jugados
* Cant. de partidos ganados
* Cant. De partidos empatados
* Cant. De goles recibidos
* Cant. De goles realizados

Incluir métodos para calcular:

* Puntos
* Cant. De partidos perdidos
* Diferencia de goles
* **class** Equipo:  
   **def** \_\_init\_\_(self):  
   self.nombre=**""** self.jugados=0  
   self.ganados=0  
   self.empatados=0  
   self.realizados=0  
   self.recibidos=0  
    
   **def** puntos(self):  
   **return** self.ganados \* 3 + self.empatados \* 1  
    
   **def** perdidos(self):  
   **return** self.jugados - self.ganados - self.empatados  
    
   **def** diferencia(self):  
   **return** self.realizados - self.recibidos  
    
   **def** \_\_str\_\_(self):  
   **return "Equipo: "** + self.nombre + **" - J :"** + str(self.jugados) + **" - P:"** + str(self.puntos()) + **" - G:"** + str(self.ganados) + **" - E:"** + str(self.empatados)+ **" - P:"** + str(self.perdidos())+ **" - GF:"** + str(self.realizados)+ **" - GC:"** + str(self.recibidos)+ **" - D:"** + str(self.diferencia())  
    
  listaEquipos=[]  
  n=int(input(**"ingrese la cantidad de equipos: "**))  
  **for** j **in** range(n):  
   e=Equipo()  
   print(**"-------------------"**)  
   e.nombre = str(input(**"Nombre: "**))  
   e.jugados = int(input(**"Jugados: "**))  
   e.ganados=int(input(**"Ganados:"**))  
   e.empatados=int(input(**"Empatados:"**))  
   e.realizados=int(input(**"Realizados:"**))  
   e.recibidos=int(input(**"Recibidos:"**))  
   listaEquipos.append(e)  
  **for** i **in** listaEquipos:  
   print(i)

#### Hacer un programa que cargue los datos de algunos equipos e informe todos los datos de quien lidera la tabla