Introducción y Fundamentos de Programación en Python
Ejercicios y prácticas

A1 – Tipos básicos

1. Un país tiene monedas de 1 U, 5 U y 25 U. Si tenemos una cierta cantidad de dinero en una variable, diseña fórmulas para dar el cambio óptimo en las monedas descritas.

```
>>> dinero = 459
>>> mon_1 = dinero % 5
…
```

   Completa un programa que pida una cierta cantidad de dinero y dé una salida adecuada:

```
Dame la cantidad de dinero que deseas cambiar: 459
Desglose de 459 U:
    18 monedas de 25 U
    1 monedas de 5 U
    4 monedas de 1 U
```

2. En casi todas partes, es clásico el programa “Hola Mundo”, que simplemente escribe esta frase en la pantalla. Diséñalo, y haz luego dos versiones o tres más:

   - Una, que pregunta el nombre del usuario y lo saluda con dicho nombre:

```
¿Cómo te llamas? Edmundo
Hola, Edmundo
```

   - Otra, que pregunta también la edad y le contesta con su nombre, la edad y los años que cumplirá la próxima vez. (Esto es simplemente para que convierta la edad en un número…)

   - Una última versión que funciona desde la consola del sistema operativo.

```
if __name__ == "__main__":
    ...
```

```
c:\Users\cpareja\python_progrs>python hola_mundo.py
¿Cómo te llamas? Edmundo
Hola, Edmundo
```

3. Un nombre propio ha de escribirse con la primera letra mayúscula. Diseña instrucciones para que, en el caso de que se dé con minúscula, lo arregle:

```
>>> nombre_propio = “blacky”
...
>>> print(nombre_propio)
Blacky
```

En este ejercicio lo que te pido en realidad es que investigues la forma de hacerlo, anticipándote al uso de strings. Hay una función pensada para lograrlo (capitalize). Y también, de otro modo, puedes separar la inicial (cadena[0]) del resto de una cadena de caracteres(cadena[1:]) , poner la inicial con mayúscula (upper) y unir ambos (+).

4. Si tenemos los coeficientes (a, b, c) de una ecuación de segundo grado y sabemos que tiene dos raíces reales, expresa instrucciones para calcular dichas fórmulas.

5. (*) Ídem. Si sabemos que las dos raíces son complejas.[1]

6. ¡Magia!

¡Magia!

El ordenador es un mago

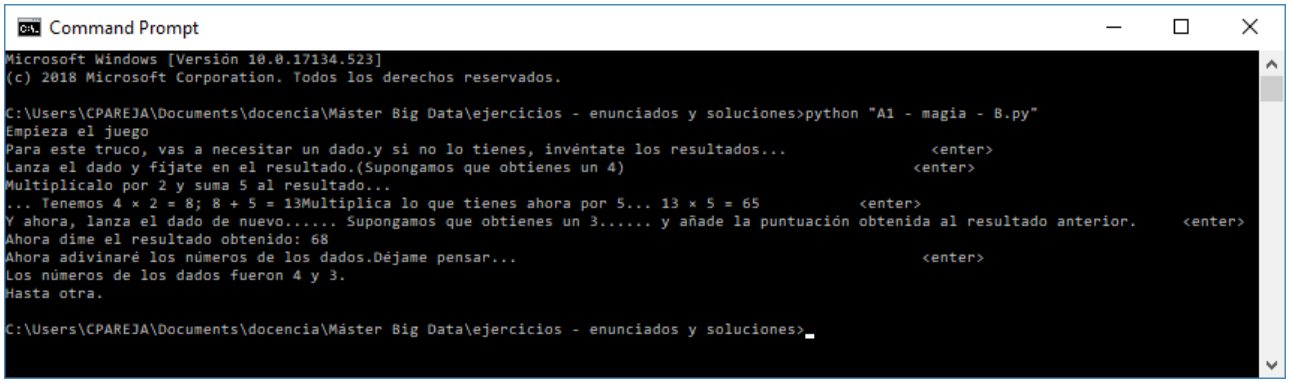
Vamos a entrenar al ordenador para efectuar un truco de magia. El programa "mago" pronunciará las siguientes palabras, poco a poco. A cada frase, el espectador irá pulsando la tecla <ENTER> tras los puntos suspensivos:

Esto de aquí debajo es el efecto visible	Y esto otro explica cómo marchan las cosas
Para este truco, vas a necesitar un dado...	Si no lo tienes, invéntate los resultados
Lanza el dado y fíjate en el resultado...	Supongamos que obtienes un 4
Multiplícalo por 2 y suma 5 al resultado...	Tenemos 4 × 2 = 8; 8 + 5 = 13
Multiplica lo que tienes ahora por 5...	13× 5 = 65
Y ahora, lanza el dado de nuevo...	Supongamos que obtienes un 3
y añade la puntuación obtenida	
al resultado anterior...	Obtenemos 65 + 3 = 68
Dime el resultado obtenido: 68	Le indicamos lo que tenemos
Ahora adivinaré los números de los dados.	
Déjame pensar...	
Los números de los dados fueron 4 y 3.	Aquí, lo típico es poner cara de sorpresa :-)
Hasta otra.	

Querrás saber el truco que ha empleado el ordenador ¿verdad? En realidad, es el espectador quien está efectuando la mayoría de las cuentas, como has visto. El trabajo secreto del ordenador se reduce a restar 25 al resultado (68) proporcionado, y en separar las dos cifras resultantes:

$$68 - 25 = 43$$
$$43 \rightarrow (4, 3)$$

Parte B: modifica tu programa para que funcione desde la consola del s.o.



[1] Los ejercicios marcados con un asterisco son prescindibles en un primer contacto con la programación.

A2 – Funciones

1. Reescribe los ejercicios del capítulo de “Tipos básicos” usando funciones. En cada caso, distingue entre funciones (devuelven un resultado) y procedimientos (realizan instrucciones). Documenta adecuadamente cada función.

2. Diseña una función que calcule el mínimo y el máximo de dos reales dados. Observa que, para este ejercicio, a lo mejor necesitas usar una expresión condicional, o una instrucción condicional. Seguro que sabes apañártelas aunque aún no se haya explicado esto, y seguro que sabes hacerlo con ambas posibilidades ☺.

3. Diseña una función que calcule la raíz de un número, así: cuando no decimos el índice, se sobre entiende que es dos; esto es, la raíz cuadrada:

raiz(5)	→	2.23606797749979
raiz(5, índice=2)	→	2.23606797749979
raiz(5, índice=3)	→	1.7099759466766968

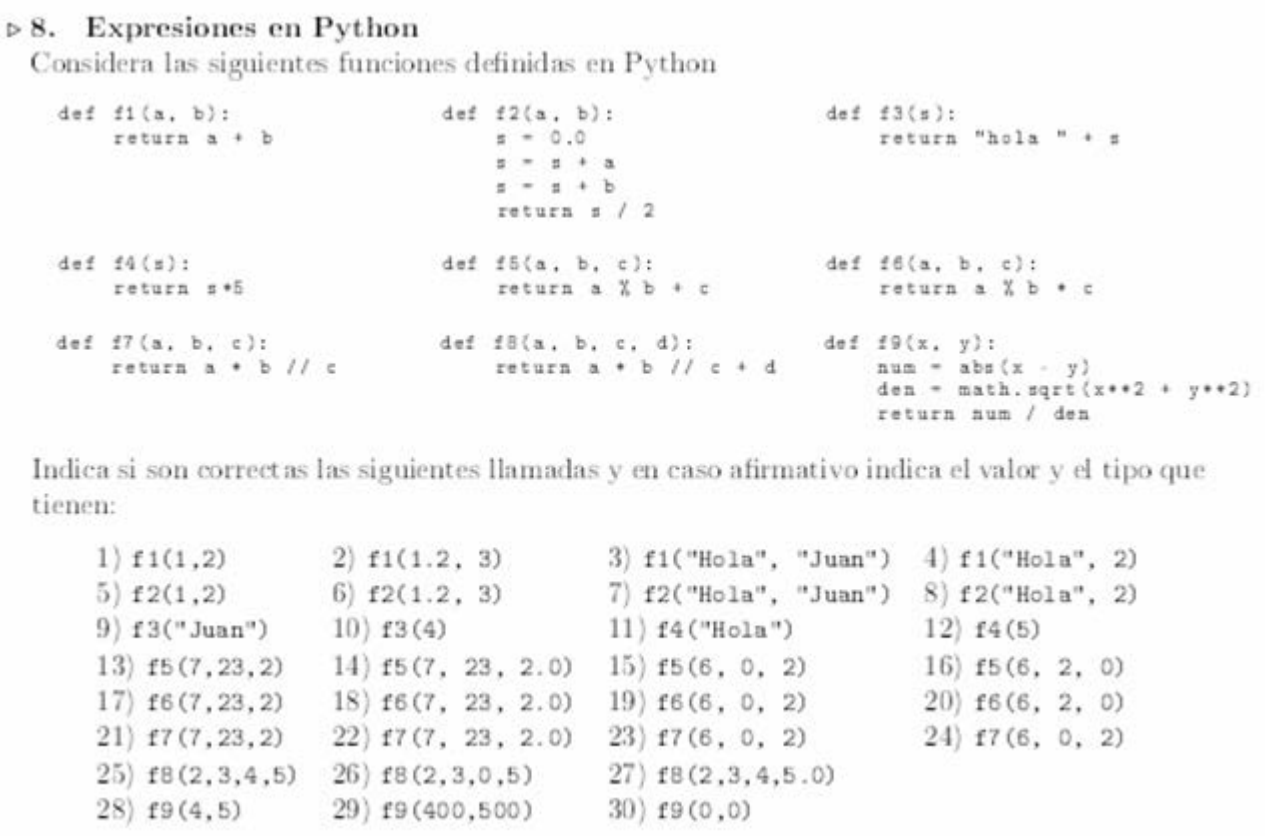
   Esta posibilidad recibe el nombre de “parámetros por defecto”.

4. Una moneda (cargada) cae de cara con una probabilidad de 0.7. Diseña una función aleatoria que lance dicha moneda, y responda “cara” o “cruz” con arreglo a las probabilidades dadas. Es decir, investiga el uso de “random”.

5. (*) Estudia las siguientes sucesiones y escribe una expresión que calcule el término general:

   a) 1, 3, 5, 7, 9, …
   b) 3, 8, 15, 24, …
   c) 1, -1, 1, -1, …
   d) 1, 1, 2, 3, 5, 8, …
   e) 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4…
   f) 1, 2, 4, 8, 16, 32, 64, …

6. El siguiente ejercicio está tomado del libro “Ejercicios creativos y recreativos en C++ (v. http://antares.sip.ucm.es/cpareja/libroCPP/).



▷ 8. Expresiones en Python

Considera las siguientes funciones definidas en Python

```
def f1(a, b):
    return a + b

def f2(a, b):
    s = 0.0
    s = s + a
    s = s + b
    return s / 2

def f3(s):
    return "hola " + s

def f4(s):
    return s*5

def f5(a, b, c):
    return a % b + c

def f6(a, b, c):
    return a % b * c

def f7(a, b, c):
    return a * b // c

def f8(a, b, c, d):
    return a * b // c + d

def f9(x, y):
    num = abs(x - y)
    den = math.sqrt(x**2 + y**2)
    return num / den
```

Indica si son correctas las siguientes llamadas y en caso afirmativo indica el valor y el tipo que tienen:

1) f1(1,2)	2) f1(1.2, 3)	3) f1("Hola", "Juan")	4) f1("Hola", 2)
5) f2(1,2)	6) f2(1.2, 3)	7) f2("Hola", "Juan")	8) f2("Hola", 2)
9) f3("Juan")	10) f3(4)	11) f4("Hola")	12) f4(5)
13) f5(7,23,2)	14) f5(7, 23, 2.0)	15) f5(6, 0, 2)	16) f5(6, 2, 0)
17) f6(7,23,2)	18) f6(7, 23, 2.0)	19) f6(6, 0, 2)	20) f6(6, 2, 0)
21) f7(7,23,2)	22) f7(7, 23, 2.0)	23) f7(6, 0, 2)	24) f7(6, 0, 2)
25) f8(2,3,4,5)	26) f8(2,3,0,5)	27) f8(2,3,4,5.0)	
28) f9(4,5)	29) f9(400,500)	30) f9(0,0)	

(La referencia mencionada puede resultarte útil para resolver muchos otros de los ejercicios planteados en este libro.)

A3 – Condicionales

1. Diseña una función que dé la calificación (“suspenso”, …, Matrícula de Honor”) correspondiente a una nota, exigiendo que la entrada sea un número real entre 0 y 10.

2. Ecuación de segundo grado. Diseña un programa que calcule las soluciones de una ecuación de la forma $ax^2 + bx + c = 0$. Debes tener en cuenta que los coeficientes pueden ser reales arbitrarios, nulos o no, etc., dando lugar a una ecuación sin solución, con una o con dos soluciones reales o imaginarias.

3. (*) Si un punto tiene cada una de sus coordenadas positivas o negativas, podrá estar en el cuadrante 1, 2, 3, 4, en alguno de los ejes abcisas u ordenadas o en incluso el origen. Diseña una función que da la posición de un punto según lo dicho.

4. Códigos de rotación.

▷ **5. Rotación del alfabeto**

Imaginemos las letras del alfabeto ordenadas y dispuestas en círculo. Esto es, a la derecha de la A se encuentra la B, luego la C y así sucesivamente hasta la Z; a la derecha de la Z se encuentra nuevamente la A.

Definimos una rotación de longitud n como aquella que lleva a una determinada letra n posiciones hacia su derecha.

Ejemplo La rotación de longitud 1 lleva la A a la B, la V a la W y la Z a la A.
La rotación de longitud 3 lleva la A a la D, la V a la Y y la Z a la C.

Rotación de longitud 1 Escribe un programa en Python que permita calcular la rotación de longitud 1. Es decir, dada una letra del alfabeto el programa debería indicar la letra correspondiente a su rotación.

Rotación de longitud arbitraria Escribe un programa en Python que permita calcular una rotación de longitud n arbitraria. El valor n debe ser leído por el programa.

Este ejercicio requiere conocer funciones y piezas que no se han explicado, como muchas situaciones reales. La solución es ir desarrollando la iniciativa de buscar dichas piezas en Internet, lo que requiere saber formular las preguntas adecuadas.

A lo mejor eres capaz tú solo. Pero por si acaso, te ayudo un poco:

> ¿Cómo convertir en Python un carácter en un entero (su código)? ¿Y al revés? ¿Cómo saber si un carácter es una letra mayúscula o minúscula? ¿Cómo unir una lista de caracteres para formar un string? ¿Hay manera de tener la lista de letras mayúsculas (o minúsculas) de forma cómoda?

Posiblemente no necesites ya más pistas. Pero por si acaso…

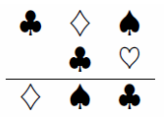
> ¿Cómo operan las funciones o métodos str, char, isalpha, isupper, join, ascii_lowercase…?

A4 y A5 – Estructuras iterativas: bucles

1. Uno de los algoritmos más antiguos es el que calcula el máximo común divisor mediante restas sucesivas. (a) Diséñalo. (b) También, mira a ver si existe una función en Python que lo calcule, sin tener que diseñarlo tú.

2. (*) El mínimo común múltiplo puede calcularse aumentando… hasta llegar a un múltiplo común. Diseña este algoritmo.

3. Se ha encontrado en Marte la siguiente operación de sumar, resuelta en una roca:



Se desea descifrar el significado (o sea, el valor) de esos símbolos, supuesto que se ha usado el sistema de numeración decimal.

Una posibilidad es tantear todas las combinaciones de cifras posibles y ver cuáles de ellas funcionan:

♣	♢	♠	♡
0	0	0	0
0	0	0	1
…	…	…	…
0	0	0	9
0	0	1	0
…	…	…	…
0	9	9	9
1	0	0	0
…	…	…	…

4. Programa de adivinación (del libro de “Ejercicios […] creativos […]”)

2.7 Juego de adivinación

Consideremos el siguiente juego entre los jugadores A (adivino) y P (pensador): P piensa un número comprendido entre 1 y N (digamos 1000, por ejemplo), y A trata de adivinarlo, mediante tanteos sucesivos, hasta dar con él. Por cada tanteo de A, P da una respuesta orientativa de entre las siguientes:

```
Fallaste.  El número pensado es menor que el tuyo.
Fallaste.  Mi número es mayor.
Acertaste al fin.
```

Naturalmente, caben diferentes estrategias para el adivino:

- Una posibilidad, si el adivino no tiene prisa en acabar, consiste en tantear números sucesivamente: primero el 1, después el 2, etc. hasta acertar.
- Otra estrategia, sin duda la más astuta, consiste en tantear el número central de los posibles de modo que, al fallar, se limiten las posibilidades a la mitad (por eso se llama *bipartición* a este modo de tantear).
- También es posible jugar caprichosamente, tanteando un número al azar entre los posibles. Al igual que la anterior, esta estrategia también reduce el intervalo de posibilidades sensiblemente.

Se plantea desarrollar tres programas independientes: uno deberá desempeñar el papel del pensador; otro el del adivino (usando la estrategia de bipartición), y un tercero deberá efectuar la simulación del juego, asumiendo ambos papeles (y donde el adivino efectúa los tanteos a capricho).

Para simular la elección de números al azar, puede consultarse el ejercicio 3.13.

5. (*) Diseña una función que calcule el cero de otra función según el método de bipartición.

Este ejercicio está tomado del libro “Ejercicios creativos y recreativos en C++” (v. http://antares.sip.ucm.es/cpareja/libroCPP/). En la página 227 puede encontrarse una breve explicación del mismo, quizá te resulte útil.

6. (*) Ídem, mediante el algoritmo de Newton-Raphson.

7. (*) Diseña un algoritmo que realice la descomposición clásica de un número en factores, a la manera clásica: se comienza dividiendo el número original entre el divisor más pequeño posible (2), se actualiza el dividendo y se continúa con ese divisor o con el siguiente, cuando haya de ser así:

```
60|2
30|2
15|3
5|5
1|
```

8. (*) Diseña una función que imprima todos los divisores de un número.

   Bueno, en realidad, lo de que “imprima”, no me gusta mucho, porque eso no es una función-función, sino una instrucción (un procedimiento). Modifícala para que devuelva la lista de los divisores. El fragmento de código que te muestro a la derecha puede resultarte útil para lograrlo.

```
In [1]: lista = []
In [2]: print(lista)
[]
In [3]: lista.append(33)
In [4]: print(lista)
[33]
In [5]: lista.append(666)
In [6]: print(lista)
[33, 666]
```

9. Diseña una función que tome una lista de enteros como parámetro y dé el mínimo y el máximo valor de dicha lista. ¿Existirán en Python estas funciones?

10. Diseña una función que filtra las palabras de una lista, quedándose con las que están escritas en minúscula.

11. (*) La función que se muestra a la derecha busca divisores de un número, para descartarlo como primo en caso de encontrar alguno.

```
def es_primo(n):
    if n<2:
        return False
    for i in range(2, int(math.sqrt(n)+1)):
        if n%i == 0:
            return False
    return True
```

Vemos en primer lugar la instrucción return dentro de un bucle for, asunto que se desaconseja en casi todos los lenguajes de Programación. En Python, no pasa nada: es la costumbre Pythónica.

En segundo lugar, este algoritmo limita la comprobación a los posibles divisores que no rebasan la raíz del dato… ¿te parece correcto?

B1 – Listas

1. a) MCD, generando las listas de divisores y seleccionando el mayor común.

   b) Este ejercicio puede hacerse de una manera más bonita usando la notación de listas por comprensión.

2. Función que toma una palabra y la limpia, de todos los caracteres que no son alfabéticos, por la izquierda o por la derecha:

   “quijote,” → “quijote”
   “ 4 quijote,” → “quijote”

   Puedes hacerlo tú mismo, pero también puedes inspeccionar las funciones “strip”, “lstrip”, “rstrip” en Python.

3. (*) Diseña un programa que simule la conocida criba de Eratóstenes.

4. (*) Tenemos un polinomio dado en una lista de los coeficientes, de menor a mayor orden, y con un cero para los coeficientes nulos, lógicamente. Diseña una función que calcule la lista correspondiente al polinomio derivada. Diseña también una función que calcule la derivada enésima, siendo el orden un parámetro que, si se omite, es uno.

   La función derivada admite una solución usando la función “enumerate” y otra sin ella. También se puede definir con listas intensionales y sin ellas. Diseña ahora una segunda versión de la derivada.

5. Python proporciona distintas maneras de ordenar una lista. Nos interesa especialmente poder hacerlo usando una función arbitraria que compare los elementos.

   Tenemos una colección de puntos del plano, esto es, una lista de pares de reales. La llamamos, sencillamente “puntos”.

   a) Para hacer los siguientes apartados, crea una lista de puntos del plano restringida a $-10 \leq x \leq 10$, $-10 \leq y \leq 10$, esto es, $(x, y) \in [-10, 10)^2$.

   b) Diseña una función que da la distancia de un punto al origen de coordenadas, esto es, al punto (0, 0). Esta función se puede usar para ordenar la lista de puntos generada, por orden de más cercano a más lejano al (0, 0).

   c) Más difícil todavía: tenemos un punto especial “P”. Deseamos diseñar una función que ordene la lista de puntos de más cercano a más lejano a “P”, esto es, de menor a mayor distancia a “P”, usando la función “sort” predefinida. (Para resolver esto, tendrás que manejar alguna función de orden superior.)

6. Dado un string, deseamos cambiar sus vocales por un carácter punto, “.” Ejemplo:

'Tengo que decir que esto es sencillo.' → 'T.ng. q.. d.c.r q.. .st. .s s.nc.ll..'

Piezas útiles. Te doy algunas pistas. A ver si sabes buscar tú solo en Internet la información que necesites.

- ¿Puede recorrerse un string igual con “for c in cadena”, igual que una lista? ¿Haría falta convertir el string en una lista? Esto puede hacerse usando la función list, pero a lo mejor no hace falta.
- Predicado que dice si un carácter es una vocal (minúscula o mayúscula).
- Expresión o función que convierte un carácter en “.” si es vocal, o lo deja igual, si no.
- Lista que hace lo dicho con todos los caracteres de una lista. Para una cadena de caracteres, list(cadena), aunque, ¿hace falta?
- Se puede unir todos los caracteres de una lista para formar un string, con un bucle o con la función “insert”. (V. librería functools.) Trata de hacerlo de las dos formas.
- Una tercera forma: observa esta manera de unir caracteres:
  >>> "-".join("Hola")
  E investiga cómo usarlas para pasar de ['H', 'o', 'l', 'a'] a 'Hola'
- Luego está la notación de listas por comprensión:

```
lista_de_letras =  ["." if …… else c for c in …]
resultado =  …… lista_de_letras ……
```

B2 – Conjuntos y diccionarios

1. Ella declara sus aficiones (una lista de strings) y luego lo hace él. Entre los dos, diseñan un programa que calcula sus aficiones comunes, las que tiene él y no ella, las que tiene ella y no él, las que tienen entre ambos. Y para ello, usan la notación de conjuntos.

2. (*) Diseña el algoritmo para el mcd con conjuntos y con la notación intensional.

3. Agenda básica. Tenemos una lista de líneas de texto como la siguiente:

   ```
   "639232983 # blaky@ucm.es # calle BB 45 8-3-3 Madrid"
   ```

   Cada una contiene un número de teléfono, una dirección de email y una dirección física. El separador es la cadena de tres caracteres " # ". Diseña una función cuya entrada es una lista de líneas como la descrita y forma un diccionario, donde cada clave es un número de teléfono, y el valor asociado, un nuevo diccionario con dos elementos: el email y la dirección.

   Nota: usa el método split(…) para descomponer cada línea.

4. Para ver el funcionamiento del programa anterior, diseña también una función que dé un listado legible de la agenda, una vez leída en un diccionario.

5. (Ejercicio de investigación.) A menudo, usamos un diccionario para contabilizar las apariciones de ciertos ítems. El seudocódigo sería éste:

   ```
   inicio del diccionario vacío
   si un elemento está en el diccionario:
       contabilizar una unidad más del elemento
   si no:
       poner el elemento con el valor 1
   ```

   Pero deseamos hacerlo con algo más sencillo:

   ```
   inicio del diccionario vacío
   contabilizar una unidad más del elemento
   ```

   Para lograrlo, usamos un diccionario con un valor por defecto (el cero). Busca la manera de hacerlo y pon un ejemplo de funcionamiento.

B3 – Archivos

1. La agenda básica del apartado anterior se puede mejorar, asumiendo que los datos de la agenda están almacenados en un archivo de disco, y que el parámetro de la función es el nombre de dicho archivo.

   Nota: usa la función rstrip() para eliminar las marcas de fin de línea.

2. Diseña una función que lee las líneas de un archivo como el siguiente:

```
En un lugar de la Mancha
de cuyo nombre no quiero acordarme
vivía un hidalgo
…
```

y las escribe, numerándolas:

```
1. En un lugar de la Mancha
2. de cuyo nombre no quiero acordarme
3. vivía un hidalgo
…
```

3. Con el archivo anterior, diseña una función que calcule cuántas palabras tiene una línea en promedio.

4. He aquí un programa (entrada_salida.py) listo para funcionar en línea de comandos. Adapta uno de los dos programas anteriores para que funcione desde la línea de comandos, dando como parámetro el archivo dato.

```
import sys

def main():
    if len(sys.argv) != 2:
        print ('uso correcto: python ./programa.py' \
            + '<nombre archivo agenda>')
        sys.exit(1)

    <Siguientes instrucciones>

if __name__ == '__main__':
    main()
```

Pruébalo así, desde la consola del sistema operativo,

```
…> python programa.py agenda.txt
…> python programa.py
```

y saca tus conclusiones.

5. A partir de un archivo de texto (ej.: “texto.txt”), podemos formar:

   a) La lista de las palabras del texto
   b) Para que no se repitan, podemos usar un diccionario (poniéndolas en minúscula tal vez)
   c) Un diccionario que, para cada palabra, dé su frecuencia
   d) Un diccionario que, para cada frecuencia, dé el conjunto de palabras que tienen dicha frecuencia)

6. El ejercicio codificar mediante códigos de rotación puede reescribirse para que funciones desde la consola del sistema operativo, así:

```
…> python codifica.py “archivo_in” “archivo_out” “Natural”
```

B4 – Objetos mutables e inmutables

1. Queremos hacer algo para modificar un entero, o un par de enteros, como los términos de una fracción, por ejemplo. Pero lo siguiente no funciona:

```
def incr(l):
    l[0] = l[0]+1

>>> n = 4
>>> incr([n])
>>> print(n)
4

>>> lista = [n]

>>> incr(lista)
>>> print(n)
4
```

2. Y no debe estar del todo mal, porque…

```
>>> n = 4
>>> lista = [n]
>>> incr(lista)
print(lista)
 [5]
```

   Ayúdame a entender qué pasa y a arreglarlo para poder usar funciones que modifican sus parámetros, enteros.

12. Una fracción se puede simplificar calculando el máximo común divisor de sus términos… Diseña una función que da la fracción simplificada y otro que simplifica “in place” una fracción dada. Este asunto puede ilustrar el concepto de objetos mutables e inmutables.

B5 – Excepciones

1. En el ejercicio de la agenda (v. B3, archivos), cuando el archivo no está presente en el disco, se puede añadir una excepción, evitando así que el programa dé un error de ejecución.

   Este uso de las excepciones para identificar la inexistencia de archivos mencionados en un programa es muy corriente en los proyectos de programación.

2. (*) Además de las excepciones, tenemos la función assert, para asegurarnos de que los requisitos de una función se cumplen.

   Diseña un par de funciones con la instrucción assert y con excepciones para calcular el mcd de dos números, pensando en un usuario que pueda algún dato nulo, o negativo… y compara su funcionamiento cuando la función es llamada desde otra.

B6 – Objetos

1. (*) Define un módulo “Complejo” basado en la representación binómica y con operaciones de creación (en forma binómica o polar), para obtener la parte real, imaginaria, módulo y argumento de un complejo, así como operaciones de escritura (en forma binómica o polar). Operaciones con complejos.

2. Define el objeto “automóvil”, que únicamente puede estar orientado según las direcciones de los puntos cardinales, N, S, E, O, y cuyas velocidades son números naturales, con el estado inicial siguiente…

   - motor_encendido: False
   - velocidad: 0
   - dirección: “N”

   … y los métodos siguientes:

   - encender_motor()
   - apagar_motor(), que únicamente funciona si la velocidad es nula.
   - acelerar(), decelerar(), frenar()
   - giro_dcha(), giro_izda()
   - Operación de escritura, para operar con print.

3. Define la ficha de una persona es una clase con su nombre, fecha de nacimiento, edad, estatura y aficiones.

   Define ahora un diccionario cuya clave es el email de una persona y sus datos, los definidos en la clase anterior.

   Completa la tarea con unas pocas funciones de prueba.

D5 – PF y orden superior

1. Genera las siguientes listas, usando listas intensionales o usando las funciones de orden superior map, filter, etc.

   - La lista con los cuadrados de los 100 primeros números
   - Ídem, pero únicamente de los números pares.

2. (*) Diseña funciones para los siguientes cálculos:
   - La derivada de una función (derivable) en un punto
   - El método de bipartición para calcular el cero de una función en un intervalo supuesto que...
   - Newton-Raphson, partiendo de un punto y supuesto que...
   - Dado el término general de una sucesión a_n de reales (que en realidad es una función $a : N \rightarrow R$), define la función que da la lista de términos a_i para $i \in \{a_1, \ldots, a_n\}$, aplicando la función a cada término de la lista [1, …, n] mediante la función map.
   - Expresa la función “sumatorio”, que suma los términos de una sucesión entre dos límites dados, esto es, usando lambda expresiones.

3. (*) Explica lo que hace la siguiente función:

   ```
   fun = lambda a, b: lambda c : (a(b(c)), b(a(c)))
   ```

4. La función máximo se puede definir mediante un reduce:

   ```
   def maximo(lista):
       return reduce(lambda..., lista)
   ```

   ¿Serás capaz de completar los puntos suspensivos con una expresión lambda?

5. La función factorial toma un parámetro y calcula el producto de los números desde 1 hasta el dato. Escribe esto con lambda expresiones: f = λ n : ...

6. Dada una lista de nombres de persona, tenemos una función que selecciona los que tienen una longitud menor o igual a una cantidad, dada. Funciona así, por ejemplo:

   ```
   >>>l = ['Ana', 'Marta', 'Patricia', 'Alba', 'Silvia', 'Gloria', 'Lara']
   >>>short_names(l, 5), short_names(l, 3)
   (['Ana', 'Marta', 'Alba', 'Lara'], ['Ana'])
   ```

   Te pido que definas la función short_names así:

   ```
   >>>sort_names = lambda ... : list(filter ...)
   ```

7. Define la función select_multiplos(n, k), que genera los números desde 1 hasta n que son múltiplos de k. Hazlo usando listas por comprensión.

8. Define una instrucción que actúe como la siguiente,

```
>>> print(list(filter(es_primo, range(2, 100))))
```

pero usando una lista definida por comprensión:

```
>>> print([... for ... if ...])
```

9. (*) Genera una lista con 25 pares de enteros aleatorios, entre 1 y 10: son las coordenadas de 25 puntos del plano discreto. Almacenamos esta lista en una variable lista_inicial, y en otra lista_de_trabajo, con la que vamos a trabajar. Ahora, define una función que reciba dos puntos del plano discreto (dos pares de enteros) y calcule la distancia euclídea entre dichos puntos.

   Define ahora una función de orden superior tal que, dado un punto P, dé la función $dist_P$, que calcula la distancia (a P) de un punto: $dist_P(Q) = \|\overline{PQ}\|$.

   Define una función que, dado un punto y una lista de puntos, devuelve la lista de puntos dada, pero ordenada de menor a mayor distancia a P.

E1 – numpy[2]

1. [Producto de matrices] Sean A y B dos matrices cuadradas de dimensión n. Definimos la matriz C, producto de las matrices dadas, A y B, así:

$$C[i,j] = \sum_{k=0}^{n-1} A[i,k] * B[k,j]$$

2. [Determinante de una matriz]

2. Regla del corazón para determinantes de orden tres

Necesitamos unas definiciones previas que enunciamos en general. Dado el determinante

$$|A| = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2,n-1} & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} & a_{nn} \end{vmatrix}$$

llamamos **corazón** al determinante que resulta de quitar la primera y última fila y la primera y última columna, es decir

$$\heartsuit = \begin{vmatrix} a_{22} & \cdots & a_{2,n-1} \\ \vdots & \ddots & \vdots \\ a_{n-1,2} & \cdots & a_{n-1,n-1} \end{vmatrix}$$

y llamamos **esquinas** a los que resultan de quitar una fila y una columna, primeras y últimas, es decir

$$\ulcorner = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1,n-1} \\ a_{21} & a_{22} & \cdots & a_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n-1} \end{vmatrix} \qquad \lrcorner = \begin{vmatrix} a_{22} & \cdots & a_{2,n-1} & a_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n-1,2} & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,2} & \cdots & a_{n,n-1} & a_{nn} \end{vmatrix}$$

$$\llcorner = \begin{vmatrix} a_{21} & a_{22} & \cdots & a_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n-1} \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} \end{vmatrix} \qquad \urcorner = \begin{vmatrix} a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ a_{22} & \cdots & a_{2,n-1} & a_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n-1,2} & \cdots & a_{n-1,n-1} & a_{n-1,n} \end{vmatrix}.$$

Proposición 2 *Se verifica que*

$$|A| \cdot \heartsuit = \ulcorner \cdot \lrcorner - \llcorner \cdot \urcorner$$

Si es $\heartsuit \neq 0$, podemos calcular el determinante en la forma

$$|A| = \frac{1}{\heartsuit} (\ulcorner \cdot \lrcorner - \llcorner \cdot \urcorner)$$

EJEMPLO 3.

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 4 \\ 3 & 2 & 1 \end{vmatrix} = \frac{1}{5} \left(\begin{vmatrix} 1 & 2 \\ 4 & 5 \end{vmatrix} \cdot \begin{vmatrix} 5 & 4 \\ 2 & 1 \end{vmatrix} - \begin{vmatrix} 4 & 5 \\ 3 & 2 \end{vmatrix} \cdot \begin{vmatrix} 2 & 3 \\ 5 & 4 \end{vmatrix} \right) =$$

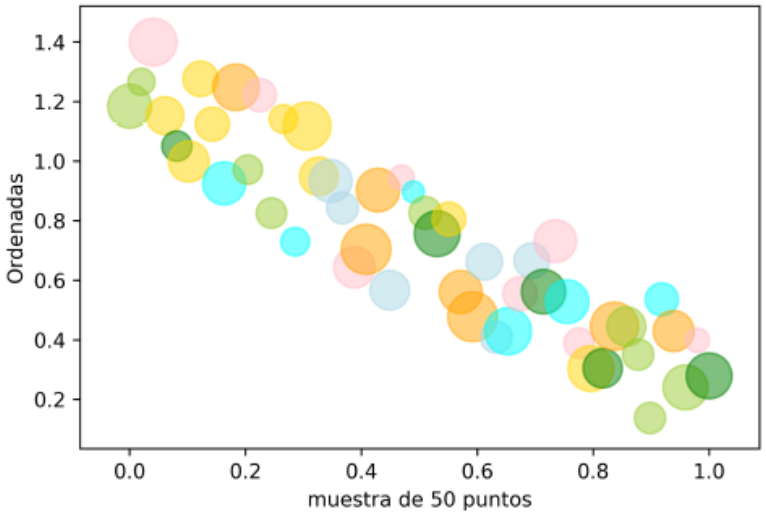
$$= \frac{1}{5} [(-3)(-3) - (-7)(-7)] = \frac{-40}{5} = -8.$$

[2] Lógicamente, esta librería tiene una utilidad eminentemente matemática.

E2 - matplotlib

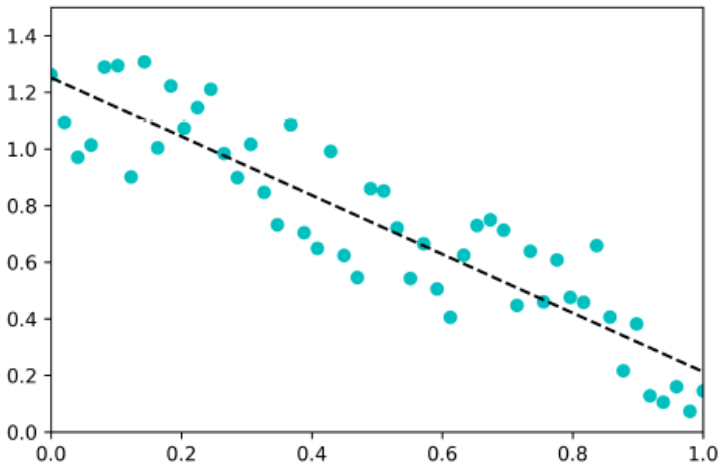
1. Nube de puntos en torno a una recta

Se pide la generación de una nube de N = 50 puntos aleatorios, en torno a la recta y = 1 - x. Digamos por ejemplo que se crean con una dispersión vertical extraída de [-0.5, 0.5] uniformemente. Los puntos deben generarse con tamaños y colores aleatorios.



2. Recta de regresión

Deseamos ahora ver la recta de regresión que aproxima la nube de puntos generados en el ejercicio anterior. Esta vez cambiamos la apariencia, porque los colores y tamaños realmente no afectan a la recoa de regresión.

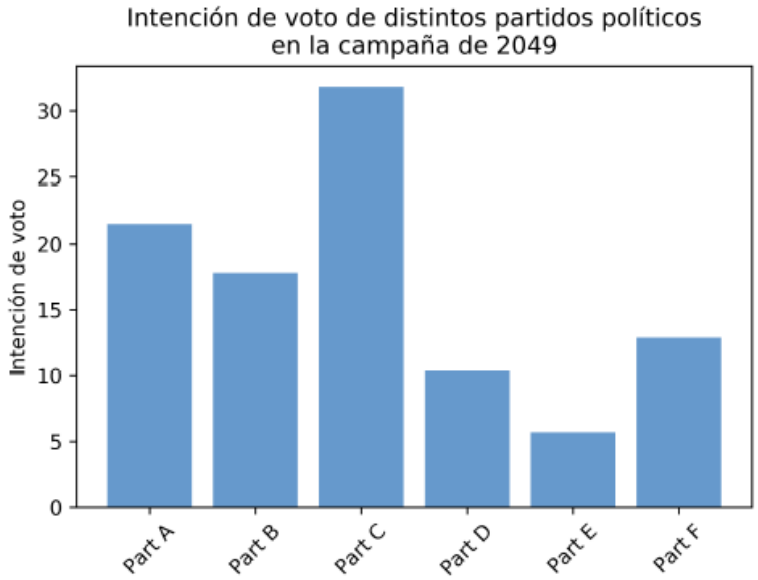


3. Diagrama de barras

Tras un sondeo a pocos días de unas elecciones, se ha registrado la intención de voto que sigue, donde se han camuflado los nombres de los partidos políticos para no influir en tu opinión personal:

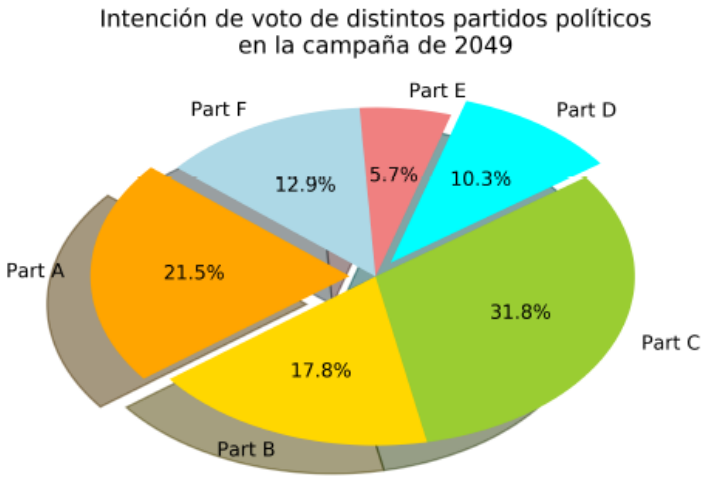
```
partidos = ['Part A', 'Part B', 'Part C', 'Part D', 'Part E', 'Part F']
intencion_voto = [21.5, 17.8, 31.8, 10.3, 5.7, 12.9]
```

Se pide mostrar estos datos en un diagrama de barras como el que sigue:



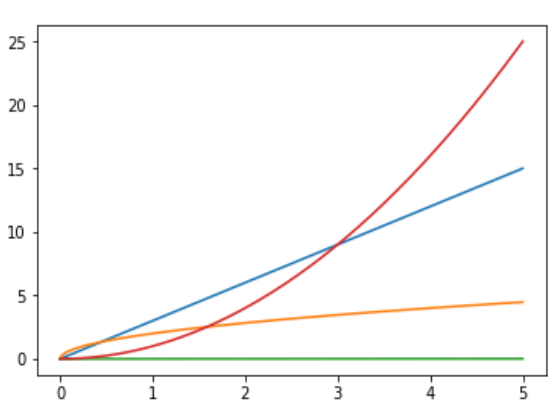
4. Diagrama de sectores

Para el sondeo anterior, deseamos ahora ver los datos recogidos en un diagrama de sectores, con los datos de los partidos A y D detacados.



1. Representa las funciones de complejidad siguientes en un gráfico único:
    a) Lineal
    b) raíz cuadrada
    c) exponencial
    d) cuadrática

Elige un rango de valores del eje de abcisas y coeficientes adecuados para mostrar las diferencias.



E3 – Archivos csv y formato JSON

1. En las páginas del INE puede encontrarse distintos archivos Excel con los nombres de persona más frecuentes:

   https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736177009&menu=resultados&idp=1254734710990

   Uno de ellos incluye todos los **nombres** con frecuencia igual o mayor a 20 personas. Contiene dos tablas, una con los masculinos y otra con los femeninos. Te pido lo siguiente:

   a) Generar dos archivos csv, uno con los nombres masculinos y otro con los femeninos.

   b) Diseña ahora un programa en Python que cargue ambos archivos en sendas listas, y las amplíe, así: la lista con los nombres femeninos puede ampliar cada elemento con un valor adicional representativo del sexo femenino, una "F". Resp., los nombres masculinos con una "M".

   c) En mismo programa puede ahora fundir en una sola, y desearíamos que se ordenara esto alfabéticamente.

   d) Finalmente, te pido que tu programa grabe esto en un archivo único, con todos los nombres de persona, en una sola tabla, que tendrá las mismas columnas que las anteriores, ampliadas con una columna adicional para recoger el sexo.

E4 – Pandas y Dataframes

1. A partir del archivo csv de viviendas en venta, deseamos localizar los que tienen más de 1000 pies cuadrados, tres dormitorios y dos baños, situados en una ciudad distinta de "Sacramento" y tener un precio entre 100.000$ y 150.000$. De ellos, obtén una estadística:

10. El listado completo
11. Un listado por ciudades y por número de baños
12. El número total de pisos de nuestra tabla
13. La media de los pisos de nuestra tabla

E6 – web scrapping

1. Diseña una función que, a partir de la dirección de una página web, extraiga todas las imágenes con formato "png".

2. Diseña una función que, a partir de la dirección de una página web, extraiga todas las imágenes dando una lista de formatos posibles: ["png", "jpg", "gif"].

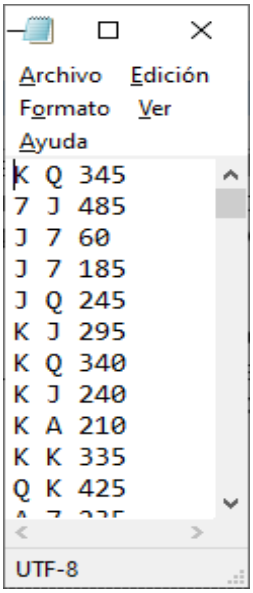
3. Diseña una función que, a partir de la dirección de una página web, extraiga todos los enlaces a otras páginas web.

4. (*) Diseña una función que, a partir de la dirección de una página web, extrae todos los enlaces a otras páginas web y todos los enlaces en el nivel siguiente.

5. (*) Diseña una función que, a partir de la dirección de una página web, extrae todos los enlaces a otras páginas web y todos los enlaces en el nivel siguiente, y en el siguiente... hasta el nivel que se indique.

E7 – map-reduce

1. Estamos en el casino. Dos jugadores lanzan un dado cada uno, y sospechamos que los dados están cargados. El casino nos encarga una aplicación para descubrirlo, y para ponerla a prueba empezamos por diseñar algunos juegos de datos.



   a) En primer lugar, debemos generar un archivo “casino.txt” con una muestra de 10000 lanzamientos, de dos dados, cada uno con probabilidades arbitrarias. V. figura.

   b) Ahora, hay que diseñar un programa que totalice el número de veces que sale cada cara de los dados, juntando los lanzamientos de ambos jugadores, para simplificar.

   Esto último, se ha de hacer con la técnica de map-reduce.

   El valor de los dados, de mayor a menor, es el siguiente:

   ["A", "K", "Q", "J", "8", "7"]

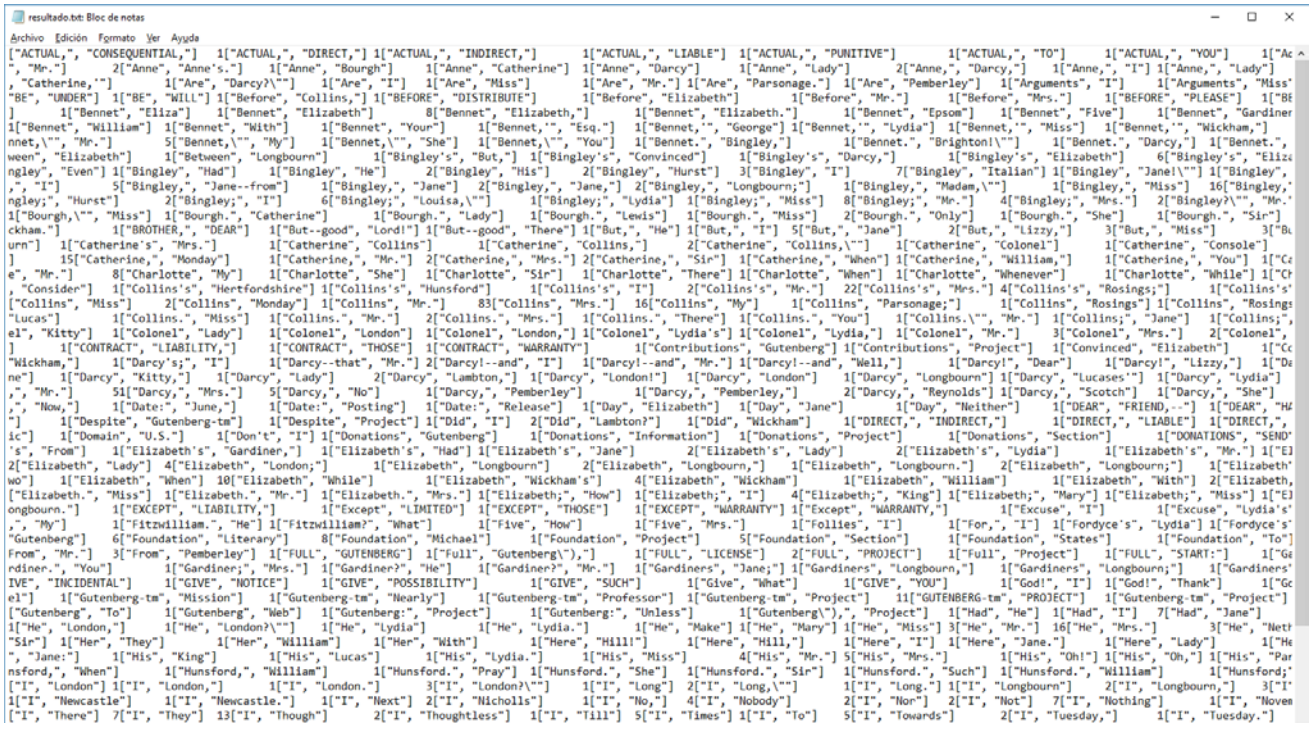
2. Tenemos un archivo de texto plano (ej.: "pride_and_prejudice.txt"). No tiene formato de líneas; esto es, cada párrafo está en una única línea.

   Se plantea diseñar un programa que contabiliza cada par de palabras con mayúscula que aparecen en un mismo párrafo, con la esperanza de que esta contabilidad dé la relación entre los personajes de una obra literaria...

   Hazlo usando la técnica de map-reduce. El programa se usará así:

```
c:\...>python cuenta_pares.py < pride_and_prejudice.txt > resultado.txt
```

   Observa el resultado que he obtenido yo:

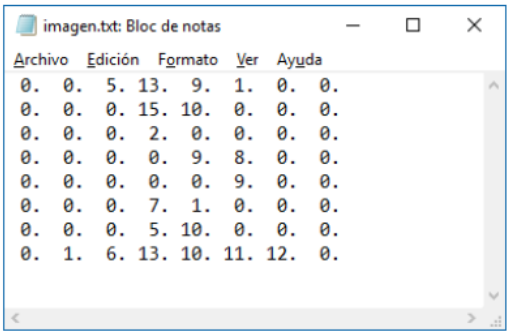


E8 – scikit-learn[3]

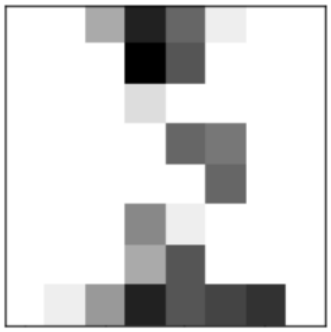
Ejercicio - máquinas de soporte vectorial

Clasificación de una imagen

Considera una matriz como la siguiente, que se encuentra en el archivo "matriz.txt":



Si interpretamos sus valores como intensidades de gris, esta imagen podría representar uno de los dígitos que tenemos en nuestra librería:



Quizás un 1 o un 2...

Lo que se pide en este ejercicio es, simplemente, usar un algoritmo de tipo svm para clasificarla, contrastándola con los dígitos de la librería de dígitos de sklearn.

Para ello, puedes completar el siguiente script:

```
# Cargar la imagen del archivo (y comprobar la carga):
# ...

# Visualizarla, con ayuda de matplotlib.pyplot:
# ...

# Visualizarla, con ayuda de matplotlib.pyplot:
# ...

# Cargar la librería de imágenes de dígitos:
from sklearn import datasets
digits = datasets.load_digits()
# ...

# Cargar y el clasificador svm:
# ...

# y ajustarlo para predecir con todos los ejemplares de entrenamiento:
# ...

# Efectuar la predicción:
# ...
```

[3] Este tema está ampliamente estudiado en muchos otros módulos y asignaturas. Se proporciona únicamente para facilitar el uso de aplicaciones de *machine learning* desde Python.

F2 - SciPy[4]

Librería scipy. Ejercicio

Este ejercicio se debe realizar consultando la librería "integrate", de scipy. V. por ejemplo el contenido de la siguiente url:

https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html

---

Integración con parámetros

Deseamos calcular la siguiente integral:

$$I(a,b) = \int_0^t (ax^2 + b)dx$$

lo que es, claramente una función de a y b (y de t, pero esto vendrá luego).

Si por ejemplo, $a = 3$ y $b = 2$, $I(3,2)$ sería una función $R \to R$, concretamente la definida como $\lambda t : \int_0^t (3x^2 + 2)dx = t^3 + 2t$.

Define la función $I(a,b)$ con arreglo a la descripción proporcionada. El resultado deberá poderse aplicar de la manera siguiente, por ejemplo:

```
>>> F = I(3, 2)
>>> print(F(2))
```

F3 – expresiones regulares

1. **Números de teléfono.** Tenemos que identificar los números de teléfono que hay en un documento, como podría ser el siguiente:

> *Te mando a continuación los teléfonos que me has pedido. Perdona que estén en formatos distintos, no he tenido tiempo de unificar esto:*
>
> *Los de los socios son: Lola, 68512345, Javier, 91-394-12-56 y 874 973 567 es el móvil, y su hermano Luis, (892) 956-56-43-67.*
>
> *Por si también lo necesitas, querrás tener los del restaurante que tanto te gusta: La Canoa de Ciguatanejo: (+034)91 657 98 23.*

Estos números están en lugares arbitrarios del texto, aunque podemos dar por sentado que un número de teléfono es una secuencia de cifras, separadas tal vez por espacios o por guiones, y precedido a veces por el prefijo del país entre paréntesis, siendo este prefijo una secuencia de dígitos, a veces con un signo + delante.

Necesitamos una expresión regular que recoja esta descripción.

[4] Lógicamente, esta librería tiene una utilidad eminentemente matemática.

2. **Extracción de imágenes.**[5] Diseña una expresión regular que capture y extraiga el nombre de un archivo de imagen (esto es, .img, .jpg, *.png) de una url. Por ejemplo, en las siguientes:

   http://www.domain.com/static/js/imagen1.jpg
   http://www.domain.com/index.php?a=imagen2.img
   http://www.domain.com/index.php?a=/imagen3.png
   http://www.domain.com/?v=http://domain.com/static/imagen4.mpg
   http://www.domain.com/?v=http://domain.com/static/?v=http://www.domain/src/este.js

   Detalles:

   - El nombre del archivo estará precedido por el carácter "/" o "=".
   - Luego, caracteres distintos del carácter "/" o "=", seguidos de un punto y una extensión válida. Ésta será la cadena de caracteres que nos interesa capturar.

[5] https://es.stackoverflow.com/questions/160945/expresi%C3%B3n-regular-en-pythom-ficheros-en-url.