FACULTAD DE INGENIERIA Y CIENCIAS EXACTAS
Departamento de Tecnología Informática

PROGRAMACIÓN I

Guía de Trabajos Prácticos

v21.03.19

Objetivos: Que el alumno pueda:

- Seleccionar la estrategia de resolución y las estructuras de datos más adecuadas para el problema planteado.
- Utilizar funciones como mecanismo para la modularización de soluciones informáticas.
- Manipular con fluidez distintas estructuras de datos, como arreglos, cadenas de caracteres, archivos, conjuntos y diccionarios.
- Reconocer en la recursión una herramienta para la resolución de problemas.

Bibliografía sugerida:

Básica

- Joyanes Aguilar, Luis: Fundamentos de Programación. Editorial Mc. Graw-Hill. ISBN 978-84-481-6111-8. EAN 9788448161118.

Complementaria

- **Tutorial de Python 3** [En línea] Python Software Foundation. Disponible gratuitamente bajo la licencia PSF. Buscar links en Google.
- Pilgrin, Mark: **Inmersión en Python 3** [En línea] Traducido por José Miguel González Aguilera. Disponible gratuitamente (bajo licencia Creative Commons 3.0). Buscar links en Google.
- Marzal Varó, A. - Gracia Luengo, I., García Sevilla, P.: **Introducción a la programación con Python 3** [En línea] Disponible bajo licencia Creative Commons. Buscar links en Google.

Autores:

Galati Verónica – Thompson Ricardo – Mazzitelli Patricia – Escalante Ramiro

Trabajo Práctico 1: Funciones

1. Desarrollar una función que reciba tres números positivos y devuelva el mayor de los tres, sólo si éste es único (mayor estricto). En caso de no existir el mayor estricto devolver -1. No utilizar operadores lógicos (and, or, not). Desarrollar también un programa para ingresar los tres valores, invocar a la función y mostrar el máximo hallado, o un mensaje informativo si éste no existe.

2. Desarrollar una función que reciba tres números enteros positivos y verifique si corresponden a una fecha válida (día, mes, año). Devolver True o False según la fecha sea correcta o no. Realizar también un programa para verificar el comportamiento de la función.

3. Una persona desea llevar el control de los gastos realizados al viajar en el subterráneo dentro de un mes. Sabiendo que dicho medio de transporte utiliza un esquema de tarifas decrecientes (detalladas en la tabla de abajo) se solicita desarrollar una función que reciba como parámetro la cantidad de viajes realizados en un determinado mes y devuelva el total gastado en viajes. Realizar también un programa para verificar el comportamiento de la función.

Cantidad de viajes	Valor del pasaje
1 a 20	Averiguar valor actualizado
21 a 30	20% de descuento sobre tarifa máxima
31 a 40	30% de descuento sobre tarifa máxima
Más de 40	40% de descuento sobre tarifa máxima

4. Un comercio de electrodomésticos necesita para su línea de cajas un programa que le indique al cajero el cambio que debe entregarle al cliente. Para eso se ingresan dos números enteros, correspondientes al total de la compra y al dinero recibido. Informar cuántos billetes de cada denominación deben ser entregados al cliente como vuelto, de tal forma que se minimice la cantidad de billetes. Considerar que existen billetes de $500, $100, $50, $20, $10, $5 y $1. Emitir un mensaje de error si el dinero recibido fuera insuficiente. Ejemplo: Si la compra es de $317 y se abona con $500, el vuelto debe contener 1 billete de $100, 1 billete de $50, 1 billete de $20, 1 billete de $10 y 3 billetes de $1.

5. Escribir dos funciones separadas para imprimir por pantalla los siguientes patrones de asteriscos, donde la cantidad de filas se recibe como parámetro:

```
**********
**********
**********
**********
**********
```

```
**
****
******
********
**********
```

6. Desarrollar una función que reciba como parámetros dos números enteros positivos y devuelva el número que resulte de concatenar ambos valores. Por ejemplo, si recibe 1234 y 567 debe devolver 1234567. No se permite utilizar facilidades de Python no vistas en clase.

7. Escribir una función diasiguiente(…) que reciba como parámetro una fecha cualquiera expresada por tres enteros (correspondientes al día, mes y año) y calcule y devuelva tres enteros correspondientes el día siguiente al dado.
   Utilizando esta función, desarrollar programas que permitan:

   a. Sumar N días a una fecha.
   b. Calcular la cantidad de días existentes entre dos fechas cualesquiera.

8. La siguiente función permite averiguar el día de la semana para una fecha determinada. La fecha se suministra en forma de tres parámetros enteros y la función devuelve 0 para domingo, 1 para lunes, 2 para martes, etc. Escribir un programa para imprimir por pantalla el calendario de un mes completo, correspondiente a un mes y año cualquiera basándose en la función suministrada. Considerar que la semana comienza en domingo.

```
def diadelasemana(dia,mes,año):
    if mes < 3:
        mes = mes + 10
        año = año - 1
    else:
        mes = mes - 2
    siglo = año // 100
    año2 = año % 100
    diasem = (((26*mes-2)//10)+dia+año2+(año2//4)+(siglo//4)-(2*siglo))%7
    if diasem < 0:
        diasem = diasem + 7
    return diasem
```

9. Resolver el siguiente problema diseñando y utilizando funciones:

   Un productor frutihortícola desea contabilizar sus cajones de naranjas según el peso para poder cargar el camión de reparto. La empresa cuenta con N camiones, y cada uno puede transportar hasta media tonelada (500 kilogramos). En un cajón caben 100 naranjas con un peso entre 200 y 300 gramos cada una. Si el peso de alguna naranja se encuentra fuera del rango indicado, se clasifica para procesar como jugo. Se solicita desarrollar un programa para ingresar la cantidad de naranjas cosechadas e informar cuántos cajones se pueden llenar, cuántas naranjas son para jugo y si hay algún sobrante de naranjas que deba considerarse para el siguiente reparto. Simular el peso de cada unidad generando un número entero al azar entre 150 y 350.

   Además, se desea saber cuántos camiones se necesitan para transportar la cosecha, considerando que la ocupación del camión no debe ser inferior al 80%; en caso contrario el camión no serán despachado por su alto costo.

Trabajo Práctico 2: Listas

1. Desarrollar cada una de las siguientes funciones y escribir un programa que permita verificar su funcionamiento imprimiendo la lista luego de invocar a cada función:
   a. Cargar una lista con números al azar de cuatro dígitos. La cantidad de elementos también será un número al azar de dos dígitos.
   b. Calcular y devolver la sumatoria de todos los elementos de la lista anterior.
   c. Eliminar todas las apariciones de un valor en la lista anterior. El valor a eliminar se ingresa desde el teclado y la función lo recibe como parámetro.
   d. Determinar si el contenido de una lista cualquiera es capicúa, sin usar listas auxiliares. Un ejemplo de lista capicúa es [50, 17, 91, 17, 50].

2. Escribir funciones para:
   a. Generar una lista de 50 números aleatorios del 1 al 100.
   b. Recibir una lista como parámetro y devolver True si la misma contiene algún elemento repetido. La función no debe modificar la lista.
   c. Recibir una lista como parámetro y devolver una nueva lista con los elementos únicos de la lista original, sin importar el orden.

   Combinar estas tres funciones en un mismo programa.

3. Crear una lista con los cuadrados de los números entre 1 y N (ambos incluidos), donde N se ingresa desde el teclado. Luego se solicita imprimir los últimos 10 valores de la lista.

4. Eliminar de una lista de palabras las palabras que se encuentren en una segunda lista. Imprimir la lista original, la lista de palabras a eliminar y la lista resultante.

5. Escribir una función que reciba una lista como parámetro y devuelva True si la lista está ordenada en forma ascendente o False en caso contrario. Por ejemplo, ordenada([1, 2, 3]) retorna True y ordenada(['b', 'a']) retorna False. Desarrollar además un programa para verificar el comportamiento de la función.

6. Escribir una función que reciba una lista de números enteros como parámetro y la normalice, es decir que todos sus elementos deben sumar 1.0, respetando las proporciones relativas que cada elemento tiene en la lista original. Desarrollar también un programa que permita verificar el comportamiento de la función. Por ejemplo, normalizar([1, 1, 2]) debe devolver [0.25, 0.25, 0.50].

7. Intercalar los elementos de una lista entre los elementos de otra. La intercalación deberá realizarse exclusivamente mediante la técnica de rebanadas y no se creará una lista nueva sino que se modificará la primera. Por ejemplo, si lista1 = [8, 1, 3] y lista2 = [5, 9, 7], lista1 deberá quedar como [8, 5, 1, 9, 3, 7].

8. Utilizar la técnica de listas por comprensión para construir una lista con todos los números impares comprendidos entre 100 y 200.

9. Generar e imprimir una lista por comprensión entre A y B con los múltiplos de 7 que no sean múltiplos de 5. A y B se ingresar desde el teclado.

10. Generar una lista con números al azar entre 1 y 100 y crear una nueva lista con los elementos de la primera que sean impares. El proceso deberá realizarse utilizando listas por comprensión. Imprimir las dos listas por pantalla.

11. Resolver el siguiente problema, diseñando las funciones a utilizar:

    Una clínica necesita un programa para atender a sus pacientes. Cada paciente que ingresa se anuncia en la recepción indicando su número de afiliado (número entero de 4 dígitos) y además indica si viene por una urgencia (ingresando un 0) o con turno (ingresando un 1). Para finalizar se ingresa -1 como número de socio. Luego se solicita:

    a. Mostrar un listado de los pacientes atendidos por urgencia y un listado de los pacientes atendidos por turno en el orden que llegaron a la clínica.

    b. Realizar la búsqueda de un número de afiliado e informar cuántas veces fue atendido por turno y cuántas por urgencia. Repetir esta búsqueda hasta que se ingrese -1 como número de afiliado.

12. Resolver el siguiente problema, utilizando funciones:

    Se desea llevar un registro de los socios que visitan un club cada día. Para ello, se ingresa el número de socio de cinco dígitos hasta ingresar un cero como fin de carga. Se solicita:

    a. Informar para cada socio, cuántas veces ingresó al club (cada socio debe aparecer una sola vez en el informe).

    b. Solicitar un número de socio que se dio de baja del club y eliminar todos sus ingresos. Mostrar los registros de entrada al club antes y después de eliminarlo. Informar cuántos ingresos se eliminaron.

Trabajo Práctico 3: Matrices

1. Desarrollar cada una de las siguientes funciones y escribir un programa que permita verificar su funcionamiento, imprimiendo la matriz luego de invocar a cada función:

   a. Cargar números enteros en una matriz de N x N, ingresando los datos desde teclado.
   b. Ordenar en forma ascendente cada una de las filas de la matriz.
   c. Intercambiar dos filas, cuyos números se reciben como parámetro.
   d. Intercambiar dos columnas dadas, cuyos números se reciben como parámetro.
   e. Trasponer la matriz sobre si misma. (intercambiar cada elemento A_{ij} por A_{ji})
   f. Calcular el promedio de los elementos de una fila, cuyo número se recibe como parámetro.
   g. Calcular el porcentaje de elementos con valor impar en una columna, cuyo número se recibe como parámetro.
   h. Determinar si la matriz es simétrica con respecto a su diagonal principal.
   i. Determinar si la matriz es simétrica con respecto a su diagonal secundaria.
   j. Determinar qué columnas de la matriz son palíndromos (capicúas), devolviendo una lista con los números de las mismas.

   NOTA: El valor de N debe leerse por teclado. Las funciones deben servir cualquiera sea el valor ingresado.

2. Para cada una de las siguientes matrices, desarrollar una función que la genere y escribir un programa que invoque a cada una de ellas y muestre por pantalla la matriz obtenida. El tamaño de las matrices debe establecerse como N x N, y las funciones deben servir aunque este valor se modifique.

a:

1	0	0	0
0	3	0	0
0	0	5	0
0	0	0	7

b:

0	0	0	27
0	0	9	0
0	3	0	0
1	0	0	0

c:

4	0	0	0
3	3	0	0
2	2	2	0
1	1	1	1

d:

8	8	8	8
4	4	4	4
2	2	2	2
1	1	1	1

e:

0	1	0	2
3	0	4	0
0	5	0	6
7	0	8	0

f:

0	0	0	1
0	0	3	2
0	6	5	4
10	9	8	7

g:

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

h:

1	2	4	7
3	5	8	11
6	9	12	14
10	13	15	16

i:

1	2	6	7
3	5	8	13
4	9	12	14
10	11	15	16

3. Desarrollar un programa para rellenar una matriz de N x N con números enteros al azar comprendidos en el intervalo $[0,N^2)$, de tal forma que ningún número se repita. Imprimir la matriz por pantalla.

4. Una fábrica de bicicletas guarda en una matriz la cantidad de unidades producidas en cada una de sus plantas durante una semana. De este modo, cada columna representa el día de la semana (Lunes columna 0, Martes columna 1…) y cada fila representa a una de sus fábricas. Ejemplo:

	(Lunes) 0	(Martes) 1	(Miércoles) 2	(Jueves) 3	(Viernes) 4	(Sábado) 5
(Fábrica 1) 0	23	150	20	120	25	150
(Fábrica 2) 1	40	75	80	0	80	35
(Fábrica 3) 2	100	120	140	120	180	120
(. . .)	. . .	. . .	. . .	. . .	. . .	. . .
(Fábrica n) 3	80	80	80	80	80	80

Se solicita:

   a. Crear una matriz con datos generados al azar de N fábricas durante una semana, considerando que la capacidad máxima de fabricación es de 150 unidades por día y puede suceder que en ciertos días no se fabrique ninguna.

   b. Mostrar la cantidad total de bicicletas fabricadas por cada fábrica.

   c. Cuál es la fábrica que más produjo en un solo día (detallar día y fábrica).

   d. Cuál es el día más productivo, considerando todas las fábricas combinadas.

   e. Crear una lista por comprensión que contenga la menor cantidad fabricada por cada fábrica.

5. Desarrolle un programa que permita realizar reservas en una sala de cine de barrio de N filas con M butacas por cada fila. Desarrollar las siguientes funciones y utilizarlas en un mismo programa:

   mostrar_butacas: Mostrará por pantalla el estado de cada una de las butacas del cine por pantalla. Se deberá mostrar antes de que el usuario realice la reserva y se volverá a mostrar luego de la misma, con los estados actualizados.

   reservar: Deberá recibir una matriz y la butaca seleccionada, y actualizará la matriz en caso de estar disponible dicha butaca. La función devolverá **True**/**False** si logró o no reservar la butaca.

   cargar_sala: recibirá una matriz como parámetro y la cargará con valores aleatorios para simular una sala con butacas ya reservadas.

   butacas_libres: Recibirá como parámetro la matriz y retornará cuántas butacas desocupadas hay en la sala.

   butacas_contiguas: Buscará la secuencia más larga de butacas libres contiguas en una misma fila y devolverá las coordenadas de inicio de la misma.

Trabajo Práctico 4: Cadenas de caracteres

1. Desarrollar una función que determine si una cadena de caracteres es capicúa, sin utilizar cadenas auxiliares ni rebanadas. Escribir además un programa que permita verificar su funcionamiento.

2. Leer una cadena de caracteres e imprimirla centrada en pantalla. Suponer que la misma tiene 80 columnas.

3. Los números de claves de dos cajas fuertes están intercalados dentro de un número entero llamado "clave maestra", cuya longitud no se conoce. Realizar un programa para obtener ambas claves, donde la primera se construye con los dígitos ubicados en posiciones impares de la clave maestra y la segunda con los dígitos ubicados en posiciones pares. Los dígitos se numeran desde la izquierda. Ejemplo: Si clave maestra fuera 18293, la clave 1 sería 123 y la clave 2 sería 89.

4. Escribir una función que reciba como parámetro un número entero entre 0 y 3999 y lo convierta en un número romano, devolviéndolo en una cadena de caracteres. ¿En qué cambiaría la función si el rango de valores fuese diferente?

5. Escribir una función filtrar_palabras() que reciba una cadena de caracteres conteniendo una frase y un entero N, y devuelva otra cadena con las palabras que tengan N o más caracteres de la cadena original. Escribir también un programa para verificar el comportamiento de la misma. Hacer tres versiones de la función, para cada uno de los siguientes casos:

    a. Utilizando sólo ciclos normales
    b. Utilizando listas por comprensión
    c. Utilizando la función filter

6. Desarrollar una función que extraiga una subcadena de una cadena de caracteres, indicando la posición y la cantidad de caracteres deseada. Devolver la subcadena como valor de retorno. Escribir también un programa para verificar el comportamiento de la misma. Ejemplo, dada la cadena "El número de teléfono es 4356-7890" extraer la subcadena que comienza en la posición 25 y tiene 9 caracteres, resultando la subcadena "4356-7890". Escribir una función para cada uno de los siguientes casos:

    a. Utilizando rebanadas
    b. Sin utilizar rebanadas

7. Escribir una función para eliminar una subcadena de una cadena de caracteres, a partir de una posición y cantidad de caracteres dadas, devolviendo la cadena resultante. Escribir también un programa para verificar el comportamiento de la misma. Escribir una función para cada uno de los siguientes casos:

    a. Utilizando rebanadas
    b. Sin utilizar rebanadas

8. Escribir una función que reciba como parámetro una cadena de caracteres en la que las palabras se encuentran separadas por uno o más espacios. Devolver otra cadena con las palabras ordenadas alfabéticamente, dejando un espacio entre cada una.

9. Desarrollar una función que devuelva una subcadena con los últimos N caracteres de una cadena dada. La cadena y el valor de N se pasan como parámetros.

10. Desarrollar una función para reemplazar todas las apariciones de una palabra por otra en una cadena de caracteres y devolver la cadena obtenida y un entero con la cantidad de reemplazos realizados. Tener en cuenta que sólo deben reemplazarse palabras completas, y no fragmentos de palabras. Escribir también un programa para verificar el comportamiento de la función.

11. Escribir un programa que cuente cuántas veces se encuentra una subcadena dentro de otra cadena, sin diferenciar mayúsculas y minúsculas. Tener en cuenta que los caracteres de la subcadena no necesariamente deben estar en forma consecutiva dentro de la cadena, pero sí respetando el orden de los mismos. Ejemplo:

    Cadena:
    Platero es pequeño, peludo, suave; tan blando por fuera, que se diría todo de algodón, que no lleva huesos. Sólo los espejos de azabache de sus ojos son duros cual dos escarabajos de cristal negro.

    Sub-cadena: *UADE*
    Cantidad encontrada: 4 (Las coincidencias están resaltadas en la cadena siguiente)

    Platero es pequeño, peludo, suave; tan blando por fuera, que se diría todo de algodón, que no lleva huesos. Sólo los espejos de azabache de sus ojos son duros cual dos escarabajos de cristal negro.

12. Escribir un programa para crear una lista por comprensión con los naipes de la baraja española. La lista debe contener cadenas de caracteres. Ejemplo: ["1 Oros", "2 Oros"...]. Imprimir la lista por pantalla.

13. Muchas aplicaciones financieras requieren que los números sean expresados también en letras. Por ejemplo, el número 2153 puede escribirse como "dos mil ciento cincuenta y tres". Escribir un programa que utilice una función para convertir un número entero entre 0 y 1 billón (1.000.000.000.000) a letras. ¿En qué cambiaría la función si también aceptara números negativos? ¿Y números con decimales?

14. Se solicita crear un programa para leer direcciones de correo electrónico y verificar si representan una dirección válida. Por ejemplo usuario@dominio.com.ar. Para que una dirección sea considerada válida el nombre de usuario debe poseer solamente caracteres alfanuméricos, la dirección contener un solo carácter @, el dominio debe tener al menos un carácter y tiene que finalizar con .com.ar.
    Repetir el proceso de validación hasta ingresar una cadena vacía. Al finalizar mostrar un listado de todos los dominios, sin repetirlos y ordenados alfabéticamente, recordando que las direcciones de mail no son case sensitive.

Trabajo Práctico 5: Manejo de excepciones

1. Desarrollar una función para ingresar a través del teclado un número natural. La función rechazará cualquier ingreso inválido de datos utilizando excepciones y mostrará la razón exacta del error. Controlar que se ingrese un número, que ese número sea entero y que sea mayor que 0. Devolver el valor ingresado cuando éste sea correcto. Escribir también un programa que permita probar el correcto funcionamiento de la misma.

2. Realizar una función que reciba como parámetros dos cadenas de caracteres conteniendo números reales, sume ambos valores y devuelva el resultado como un número real. Devolver -1 si alguna de las cadenas no contiene un número válido, utilizando manejo de excepciones para detectar el error.

3. Desarrollar una función que devuelva una cadena de caracteres con el nombre del mes cuyo número se recibe como parámetro. Los nombres de los meses deberán obtenerse de una lista de cadenas de caracteres inicializada dentro de la función. Devolver una cadena vacía si el número de mes es inválido. La detección de meses inválidos deberá realizarse a través de excepciones.

4. Todo programa Python es susceptible de ser interrumpido mediante la pulsación de las teclas Ctrl-C, lo que genera una excepción del tipo `KeyboardInterrupt`. Realizar un programa para imprimir los números enteros entre 1 y 100000, y que solicite confirmación al usuario antes de detenerse cuando se presione Ctrl-C.

5. La raíz cuadrada de un número puede obtenerse mediante la función `sqrt()` del módulo `math`. Escribir un programa que utilice esta función para calcular la raíz cuadrada de un número cualquiera ingresado a través del teclado. El programa debe utilizar manejo de excepciones para evitar errores si se ingresa un número negativo.

6. El método `index` permite buscar un elemento dentro de una lista, devolviendo la posición que éste ocupa. Sin embargo, si el elemento no pertenece a la lista se produce una excepción de tipo `ValueError`. Desarrollar un programa que cargue una lista con números enteros ingresados a través del teclado (terminando con -1) y permita que el usuario ingrese el valor de algunos elementos para visualizar la posición que ocupan, utilizando el método `index`. Si el número no pertenece a la lista se imprimirá un mensaje de error y se solicitará otro para buscar. Abortar el proceso al tercer error detectado. No utilizar el operador `in` durante la búsqueda.

7. Escribir un programa que juegue con el usuario a adivinar un número. El programa debe generar un número al azar entre 1 y 500 y el usuario debe adivinarlo. Para eso, cada vez que se introduce un valor se muestra un mensaje indicando si el número que tiene que adivinar es mayor o menor que el ingresado. Cuando consiga adivinarlo, se debe imprimir en pantalla la cantidad de intentos que le tomó hallar el número. Si el usuario introduce algo que no sea un número se mostrará un mensaje en pantalla y se lo contará como un intento más.

Trabajo Práctico 6: Archivos

1. Desarrollar un programa para eliminar todos los comentarios de un programa escrito en lenguaje Python. Tener en cuenta que los comentarios comienzan con el signo # (siempre que éste no se encuentre encerrado entre comillas simples o dobles) y que también se considera comentario a las cadenas de documentación (docstrings).

2. Escribir un programa que permita grabar un archivo los datos de lluvia caída durante un año. Cada línea del archivo se grabará con el siguiente formato:

   <dia>;<mes>;<lluvia caída en mm> por ejemplo 25;5;319

   Los datos se generarán mediante números al azar, asegurándose que las fechas sean válidas. La cantidad de registros también será un número al azar entre 50 y 200. Por último se solicita leer el archivo generado e imprimir un informe en formato matricial donde cada columna represente a un mes y cada fila corresponda a los días del mes. Imprimir además el total de lluvia caída en todo el año.

3. Una institución deportiva necesita clasificar a sus atletas para inscribirlos en los próximos Juegos Panamericanos. Para eso encargó la realización de un programa que incluya las siguientes funciones:

   GrabarRangoAlturas() Graba en un archivo las alturas de los atletas de distintas disciplinas, los que se ingresan desde el teclado. Cada dato se debe grabar en una línea distinta. Ejemplo:

   <Deporte 1>
   <altura del atleta 1>
   <altura del atleta 2>
   < . . . >
   <Deporte 2>
   <altura del atleta 1>
   <altura del atleta 2>
   < . . . >

   GrabarPromedio() Graba en un archivo los promedios de las alturas de los atletas presentes en el archivo generado en el paso anterior. La disciplina y el promedio deben grabarse en líneas diferentes. Ejemplo:

   <Deporte 1>
   <Promedio de alturas deporte 1>
   <Deporte 2>
   <Promedio de alturas deporte 2>
   < . . . >

   MostrarMasAltos() Muestra por pantalla las disciplinas deportivas cuyos atletas superan la estatura promedio general. Obtener los datos del segundo archivo.

4. Escribir un programa que lea un archivo de texto conteniendo un conjunto de apellidos y nombres en formato "Apellido, Nombre" y guarde en el archivo ARMENIA.TXT los nombres de aquellas personas cuyo apellido terminan con la cadena "IAN", en el archivo ITALIA.TXT los terminados en "INI" y en el archivo ESPAÑA.TXT los terminados en "EZ". Descartar el resto. Ejemplo:

   Arslanian, Gustavo -> ARMENIA.TXT
   Rossini, Giuseppe -> ITALIA.TXT
   Pérez, Juan -> ESPAÑA.TXT
   Smith, John -> descartar

   El archivo de entrada puede ser creado mediante el Block de Notas o el IDLE. No escribir un programa para generarlo.

5. Se dispone de tres formatos diferentes de archivos de texto en los que se almacenan datos de empleados. Los formatos se indican más abajo. Desarrollar un programa para cada uno de los formatos suministrados, que permitan leer cada uno de los archivos y grabar los datos obtenidos en otro archivo de texto con formato CSV.

   Archivo 1: Todos los campos son de longitud fija.

```
          1         2         3         4         5         6
012345678901234567890123456789012345678901234567890123456789012
Pérez Juan         20080211 Corrientes 348
González Ana M    20080115 Juan de Garay 1111 3er piso dto A
```

   Archivo 2: Los campos están separados por el signo #.

```
Pérez Juan#20080211#Corrientes 348
González Ana M#20080115#Juan de Garay 1111 3er piso Dto A
```

   Archivo 3: Todos los campos de este archivo están precedidos por un número de dos dígitos que indica la longitud del campo a leer.

```
10Pérez Juan08200802111 4Corrientes 348
14González Ana M0820080115 33Juan de Garay 1111 3er piso dto A
```

   NOTA: Los espacios que se encuentren al final de las cadenas en el archivo 1 deberán ser eliminados.

Trabajo Práctico 7: Recursividad

1. Escribir una función que devuelva la cantidad de dígitos de un número entero, sin utilizar cadenas de caracteres.

2. Desarrollar una función que reciba un número binario y lo devuelva convertido a base decimal.

3. Escribir una función que devuelva la suma de los N primeros números naturales.

4. Desarrollar una función que devuelva el producto de dos números enteros por sumas sucesivas.

5. Realizar una función que devuelva el resto de dos números enteros, utilizando restas sucesivas.

6. La función de Ackermann A(m,n) se define de la siguiente forma:

n+1	si m = 0
A(m-1,1)	si n = 0
A(m-1,A(m,n-1))	de otro modo

   Imprimir un cuadro con los valores que adopta la función para valores de m entre 0 y 3 y de n entre 0 y 7.

7. Dados dos números enteros no negativos, devolver el resultado de calcular el Máximo Común Divisor (también llamado Divisor Común Mayor) basándose en las siguientes propiedades:

   MCD(X, X) = X
   MCD(X, Y) = MCD(Y, X)
   Si X > Y => MCD(X, Y) = MCD(X–Y, Y).

   Utilizando la función anterior calcular el MCD de todos los elementos de una lista de números enteros, sabiendo que MCD(X,Y,Z) = MCD(MCD(X,Y),Z). No se permite utilizar iteraciones en ninguna etapa del proceso.

8. Realizar la implementación recursiva del método de selección para ordenar una lista de números enteros. Sugerencia: Colocar el elemento más pequeño en la primera posición, y luego ordenar el resto de la lista con una llamada recursiva.

9. Realizar una función recursiva para imprimir una matriz de M x N.

10. Escribir una función que sume todos los elementos de una matriz de M x N y devuelva el resultado.

11. Desarrollar una función que devuelva el mínimo elemento de una matriz de M x N.

Trabajo Práctico 8: Tuplas, conjuntos y diccionarios

1. Desarrollar las siguientes funciones utilizando tuplas para representar fechas y horarios, y luego escribir un programa para verificar su comportamiento:

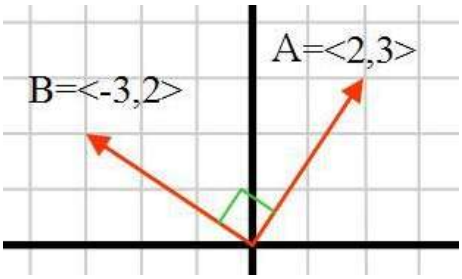
   a. Ingresar una fecha desde el teclado, verificando que corresponda a una fecha válida.
   b. Sumar N días a una fecha.
   c. Ingresar un horario desde teclado, verificando que sea correcto.
   d. Calcular la diferencia entre dos horarios. Si el primer horario fuera mayor al segundo se considerará que el primero corresponde al día anterior. En ningún caso la diferencia en horas puede superar las 24 horas.

2. Escribir una función que reciba como parámetro una tupla conteniendo una fecha (día,mes,año) y devuelva una cadena de caracteres con la misma fecha expresada en formato extendido. Por ejemplo, para (12,10,17) devuelve "12 de Octubre de 2017". Escribir también un programa para verificar su comportamiento.

3. Desarrollar un programa que utilice una función que reciba como parámetro una cadena de caracteres conteniendo una dirección de correo electrónico y devuelva una tupla con las distintas partes que componen dicha dirección. Ejemplo: alguien@uade.edu.ar -> (alguien, uade, edu, ar).

4. Escribir una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en dos tuplas, por ejemplo: (3, 4) y (5, 4). La función devuelve True o False. Escribir también un programa para verificar su comportamiento.

5. En geometría un vector es un segmento de recta orientado que va desde un punto A hasta un punto B. Los vectores en el plano se representan mediante un par ordenado de números reales (x, y) llamados *componentes*. Para representarlos basta con unir el origen de coordenadas con el punto indicado en sus componentes:



   Dos vectores son *ortogonales* cuando son perpendiculares entre sí. Para determinarlo basta calcular su producto escalar y verificar si es igual a 0. Ejemplo:

   A = (2,3) y B = (-3,2) => 2 * (-3) + 3 * 2 = -6 + 6 = 0 => Son ortogonales

   Escribir una función que reciba dos vectores en forma de tuplas y devuelva un valor de verdad indicando si son ortogonales o no. Desarrollar también un programa que permita verificar el comportamiento de la función.

6. Ingresar una frase desde el teclado y usar un conjunto para eliminar las palabras repetidas, dejando un solo ejemplar de cada una. Finalmente mostrar las palabras ordenadas según su longitud.

7. Definir un conjunto con números enteros entre 0 y 9. Luego solicitar valores al usuario y eliminarlos del conjunto mediante el método `remove`, mostrando el contenido del conjunto luego de cada eliminación. Finalizar el proceso al ingresar -1. Utilizar manejo de excepciones para evitar errores al intentar quitar elementos inexistentes.

8. Generar e imprimir un diccionario donde las claves sean números enteros entre 1 y 20 (ambos incluidos) y los valores asociados sean el cuadrado de las claves.

9. Escribir una función que reciba un número entero N y devuelva un diccionario con la tabla de multiplicar de N del 1 al 12. Escribir también un programa para probar la función.

10. Desarrollar una función eliminarclaves() que reciba como parámetros un diccionario y una lista de claves. La función debe eliminar del diccionario todas las claves contenidas en la lista, devolviendo el diccionario modificado y un valor de verdad que indique si la operación fue exitosa. Desarrollar también un programa para verificar su comportamiento.

11. Crear una función contarvocales(), que reciba una palabra y cuente cuántas letras "a" contiene, cuántas "e", cuántas "i", etc. Devolver un diccionario con los resultados. Desarrollar un programa para leer una frase e invocar a la función por cada palabra que contenga la misma. Imprimir cada palabra y la cantidad de vocales hallada.

12. Una librería almacena su lista de precios en un diccionario. Diseñar un programa para crearlo, incrementar los precios de los cuadernos en un 15%, imprimir un listado con todos los elementos de la lista de precios e indicar cuál es el ítem más costoso que venden en el comercio.

13. Escribir una función buscarclave() que reciba como parámetros un diccionario y un valor, y devuelva una lista de claves que apunten ("mapeen") a ese valor en el diccionario. Comprobar el comportamiento de la función mediante un programa apropiado.

Trabajo Práctico 9: Tipos Abstractos de Datos

Utilizar el módulo pilacola para resolver los siguientes problemas. Se pueden utilizar pilas y colas auxiliares, pero no se permite el uso de listas adicionales.

Pilas

1. Desarrollar un programa que sume los elementos de una pila, conservando intacta la misma al terminar el proceso.

2. Invertir el orden de los elementos de una pila.

3. Ordenar una pila, utilizando los métodos tradicionales de ordenamiento (Selección, Intercambio o Inserción). No se permite utilizar el método sort.

4. Escribir una función reemplazar(<pila>, <nuevo>, <viejo>) que reemplace todas las apariciones de <viejo> por <nuevo> dentro de la pila suministrada.

5. Un conductor viaja de un pueblo origen a un pueblo destino, pasando por varios pueblos intermedios. Una vez en el destino, el conductor debe regresar a casa por el mismo camino. Desarrollar un programa que permita al conductor registrar cada pueblo visitado y al finalizar el viaje le indique el camino de regreso.

Colas

6. Una universidad cuenta con una cola de estudiantes y graduados, donde cada persona se representa mediante una tupla (<situación>, <DNI>) (la situación se codifica como 0: estudiante, 1: graduado) dividirla en dos colas según el nivel educativo de cada integrante.

7. Se tiene una cola en la que se han repartido números con el orden de atención. Cada integrante se representa mediante una tupla (<número de orden>, <DNI>). Sin embargo hay muchos "colados" en la misma, los que carecen de número ("None"). Por eso se le ha ordenado al personal de seguridad que retire a todos aquellos que no tienen número. Mostrar la cola inicial, los DNI de quienes fueron retirados de la cola y la cola final.

8. Escribir un programa que devuelva la concatenación de dos colas de números enteros, respetando el orden original de cada una.

9. Escribir un programa que devuelva la concatenación de dos colas de números enteros, respetando valor de los elementos de cada una. La cola final debe quedar ordenada de menor a mayor.

10. Cargar desde el teclado la cola DADA. Imprimir un mensaje indicando si la cola DADA es capicúa.