Programación PYTHON
Hasta UNIDAD 04

Consideraciones y buenas prácticas que adoptamos para el desarrollo de los programas

- Usar nombre de variables estilo camelCase (ejemplos: fechaDeNacimiento, deudaOriginalActualizada, etc.)
- No utilizar acentos ni eñes en los nombres de variables, funciones ni programas
- Documentar el código con comentarios (#comentario """comentario""")
- Indentar (tabular) el código correctamente (utilizar la tecla TAB)
- Dejar espacio entre operadores y expresiones (ejemplo: precio=neto+iva debería codificarse precio = neto + iva)
- Usar funciones cuando se pida, y luego, aunque no se pida expresamente utilizarlas para modularizar el código
- Dar nombre a los proyectos y programas con la codificación TPXX-YY (XX = número de TP; YY número de ejercicio)
- Siempre escribir el código sobre la plantilla suministrada por el profesor
- En cada trabajo práctico sólo se deben aplicar las sentencias, funciones y métodos vistos hasta el momento y en esta materia.

INDICE

TRABAJO PRÁCTICO 01 | FUNCIONES

TP01-01 | SEGUNDOS A HORAS:MINUTOS:SEGUNDOS

Desarrollar una función que reciba un valor correspondiente a una cantidad de segundos obtenidos de una máquina que mide la cantidad de tiempo en ejecutarse un proceso. La función debe convertir dicho valor en horas, minutos y segundos y devolver el resultado en un string de la forma “hh:mm:ss”. Por ejemplo, 32130 segundos corresponden a “8:55:30”.

TP01-02 | FECHA VÁLIDA

Desarrollar una función que reciba tres números enteros positivos correspondientes al día, mes y año de una fecha, y verifique si corresponden a una fecha válida. Debe tenerse en cuenta la cantidad de días de cada mes, incluyendo los años bisiestos. La función debe devolver True o False según la fecha sea correcta o no. Realizar también un programa para verificar el comportamiento de la función.

TP01-03 | GASTO DE TRANSPORTE SUBTE

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 de 1 pasaje
1 a 20	Averiguar en internet el valor actualizado
21 a 30	20% de descuento
31 a 40	30% de descuento
41 o más	40% de descuento

TP01-04 | MÍNIMA CANTIDAD DE BILLETES

Para cerrar una cuenta corriente, un cajero de banco necesita un programa que le indique cuánto debe entregarle en efectivo a un cliente a partir de lo que tiene en la cuenta y lo que el cliente le debe al banco. Para eso se ingresan dos números enteros, uno correspondientes al saldo de la cuenta corriente y otro al monto que le corresponde al banco. Informar cuántos billetes de cada denominación debe entregarle al cliente como diferencia, de tal forma que se minimice la cantidad de billetes. Considerar que el banco dispone de billetes de $20.000, $10.000, $2.000, $1.000, $500, $200 y $100. Emitir un mensaje para el caso particular que el saldo fuera insuficiente, y otro para el caso particular que no haya que entregar dinero. Ejemplo: Si el banco debe retener $31.700 y el saldo en la cuenta es $58.300, el efectivo a entregar debe contener 1 billete de $20.000, 3 billetes de $2.000, 1 billete de $500, y 1 billetes de $100.

TP01-05 | OBLONGOS Y TRIANGULARES

Escribir dos funciones que reciban un número natural y devuelvan verdadero o falso según el número sea de alguna de las siguientes categorías:

Función oblongo(): Informa si un número es oblongo. Se dice que un número es oblongo cuando se puede obtener multiplicando dos números naturales consecutivos. Por ejemplo 6 es oblongo porque resulta de multiplicar 2 * 3.

Función triangular(): Informa si un número es triangular. Un número se define como triangular si puede expresarse como la suma de un grupo de números naturales consecutivos comenzando desde 1. Por ejemplo 10 es un número triangular porque se obtiene sumando 1+2+3+4.

TP01-06 | CONCATENAR BÁSICO

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, ni tampoco concatenar strings mediante la conversión de número a cadena.

TP01-07 | FECHA DÍA SIGUIENTE

Escribir una función fechaDiaSiguiente() que reciba como parámetro una fecha cualquiera expresada por tres enteros (correspondientes al día, mes y año) y calcule y devuelva un string (en formato “dd/mm/aaaa”) con la fecha del día siguiente a la fecha dada. Para probar la función, escribir un programa que pida los tres datos de una fecha (día, mes y año) y, aprovechando la función creada, devuelva la fecha del día siguiente al ingresado.

TP01-08 | DÍA DE LA SEMANA

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.

```python
def diaDeLaSemana(dia, mes, anio):
    """
    Función para calcular a que día de la semana corresponde una fecha (0,1,2,3,4,5,6).
    PARÁMETROS:
        dia, mes, anio: fecha para la cual obtener el día de la semana.
    SALIDA:
        Entero indicando el día de la semana (0,1,2,3,4,5,6) (0 = domingo)
    """
    if mes < 3:
        mes = mes + 10
        anio = anio - 1
    else:
        mes = mes - 2
    siglo = anio // 100
    anio2 = anio % 100
    diaSem = (((26 * mes - 2) // 10) + dia + anio2 + (anio2 // 4) + (siglo // 4) - (2 * siglo)) % 7
    if diaSem < 0:
        diaSem = diaSem + 7
    return diaSem
```

La función deberá presentar el calendario en este formato:

```
IMPRESIÓN DE CALENDARIO
-----------------------
Ingrese el mes: 2
Ingrese el año: 2024

------------------------------------------------
MES 2 AÑO 2024
------------------------------------------------
D       L       M       X       J       V       S
------------------------------------------------
                                1       2       3
4       5       6       7       8       9       10
11      12      13      14      15      16      17
18      19      20      21      22      23      24
25      26      27      28      29
```

TP01-09 | CAJONES DE NARANJAS

Resolver el siguiente problema 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 será despachado por su alto costo.

TRABAJO PRÁCTICO 02 | LISTAS

TP02-01 | ELEMENTOS NUMÉRICOS

Desarrollar cada una de las siguientes funciones y escribir un programa que permita verificar su funcionamiento, imprimiendo lo que devuelve cada función luego de invocar a cada una de ellas:

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 el producto 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. No utilizar listas auxiliares.
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].

TP02-02 | ELEMENTOS REPETIDOS

Escribir funciones para:

a. Generar una lista de N números aleatorios del 1 al 100. El valor de N se ingresa a través del teclado.
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.

TP02-03 | ELEMENTOS AL CUADRADO

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.

TP02-04 | ELEMENTOS ELIMINADOS

Eliminar de una lista de números enteros aquellos valores que se encuentren en una segunda lista. Imprimir la lista original, la lista de valores a eliminar y la lista resultante. La función debe modificar la lista original sin crear una copia modificada.

TP02-05 | VERIFICAR ORDEN DE ELEMENTOS

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.

TP02-06 | LISTA NORMALIZADA

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].

TP02-07 | INTERCALACIÓN DE ELEMENTOS

Intercalar los elementos de una lista entre los elementos de otra. La intercalación podrá realizarse utilizando el método insert o mediante la técnica de rebanadas (slicing), y nunca 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]. Las listas pueden tener distintas longitudes.

TP02-08 | AZAR Y ELEMENTOS IMPARES

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. Imprimir las dos listas por pantalla. El proceso deberá realizarse utilizando listas por comprensión.

Posteriormente, comentar todo el código anterior, y dentro del mismo programa, desarrollar una variante de solución donde el proceso deberá realizarse utilizando la función Python incorporada filter(). (investigarla)

TP02-09 | REGISTRO DE VISITAS DE SOCIOS

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 03 | MATRICES

TP03-01 | FUNCIONES CON MATRICES (MENÚ)

Desarrollar un programa que presente el siguiente menú de opciones:

```
--------------------------
MENÚ DEL PROGRAMA
--------------------------
[1] Generar matriz
[2] Ordenar matriz
[3] Intercambiar dos columnas
[4] Intercambiar dos filas
[5] Transponer matriz
[6] Verificación de simetría diagonal principal.
--------------------------
[0] Salir del programa
--------------------------

Seleccione una opción:
```

Cada opción llamará a una función a desarrollar según las siguientes funcionalidades:

1. Cargar números enteros aleatorios de 0 a 99 en una matriz de N x N, ingresando el tamaño desde el teclado.
2. Ordenar en forma ascendente cada una de las filas de la matriz.
3. Intercambiar dos columnas dadas, cuyos números se reciben como parámetro.
4. Intercambiar dos filas, cuyos números se reciben como parámetro.
5. Trasponer la matriz sobre sí misma. (intercambiar cada elemento A[i][j] por A[j][i])
6. Determinar si la matriz es simétrica con respecto a su diagonal principal (Esta función sólo devolverá True o False)

Para operar el programa siempre primero se elegirá la opción 1 que llamará a una función para generar la matriz de trabajo. La matriz de trabajo debe mantenerse inalterada en el ámbito del programa principal para que pueda servir de argumento para otras funciones. Al elegir la opción 1 se debe mostrar por pantalla la matriz generada.

Luego, al elegir cualquiera de las demás opciones del menú, se solicitarán datos de ser necesario, y luego se llamará a la correspondiente función, para finalmente presentar la matriz original junto al resultado de la función invocada para poder

comprobar los cambios. Por ejemplo, luego de llamar a la función 2, en el programa principal se mostrarán los datos de esta forma:

```
Matriz original:
[32, 8, 72, 48, 7]
[79, 99, 33, 63, 33]
[33, 97, 8, 15, 57]
[73, 60, 82, 68, 79]
[11, 77, 8, 36, 33]

Matriz procesada:
[7, 8, 32, 48, 72]
[33, 33, 63, 79, 99]
[8, 15, 33, 57, 97]
[60, 68, 73, 79, 82]
[8, 11, 33, 36, 77]
```

NOTA: No incluir los input() ni print() dentro de las funciones. Tanto input() como print() sólo deben codificarse dentro de main(), es decir, se debe enviar a las funciones los datos del input vía argumentos/parámetros, y recibir los resultados a imprimir desde las funciones vía return.

TP03-02 | GENERACIÓN DE MATRICES CON PATRONES (MENÚ)

Desarrollar un programa que presente el siguiente menú de opciones:

```
---------------------------
MENÚ DEL PROGRAMA
---------------------------
[1] Matriz patrón A (diagonal principal)
[2] matriz patrón B (diagonal secundaria)
[3] matriz patrón C (triángulo izquierda)
[4] matriz patrón D (franjas horizontales)
[5] matriz patrón E (salteado cada 2)
[6] matriz patrón F (triángulo derecha)
---------------------------
[0] Salir del programa
---------------------------

Seleccione una opción:
```

Cada opción llamará a una función a desarrollar que deberá generar cada una de las siguientes matrices:

Luego de llamar a cada función se debe mostrar por pantalla la matriz obtenida. El tamaño de las matrices debe establecerse como N x N solicitando el valor por teclado, y las funciones deben servir para cualquier valor ingresado por teclado.

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

NOTA: No incluir los input() ni print() dentro de las funciones. Tanto input() como print() sólo deben codificarse dentro de main(), es decir, se debe enviar a las funciones los datos del input vía argumentos/parámetros, y recibir los resultados a imprimir desde las funciones vía return.

TP03-03 | FÁBRICA DE BICICLETAS (MENÚ)

Una empresa que fabrica bicicletas guarda en una matriz la cantidad de unidades producidas durante una semana para cada una de sus 10 fábricas. De este modo, cada columna representa el día de la semana (Lunes columna 0, Martes columna 1, etc.) y cada fila representa cada una de sus 10 fábricas. Similar al siguiente ejemplo:

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

Se solicita desarrollar un programa con un menú de 5 opciones que llame a 5 funciones a desarrollar y que permitan:

A. Función **crearMatriz()**: Creará una matriz con datos generados al azar para las 10 fábricas y para una semana de trabajo, considerando que la capacidad máxima de fabricación es de 150 unidades por día y puede suceder que en ciertos días no se fabrique ninguna. La función devolverá la matriz creada. Mostrar la matriz por pantalla (de forma rectangular, como la imagen anterior).
B. Función **totalPorFabrica()**: La función recibirá la matriz generada y devolverá una lista con la cantidad total de bicicletas fabricadas por cada fábrica en toda la semana. Mostrar por pantalla un listado con los datos obtenidos.
C. Función **fabricaTop()**: Recibirá la matriz y devolverá cuál es la fábrica que más produjo en un solo día y cuál fue el día. Mostrar los datos por pantalla (detallar día y fábrica).
D. Función **diaMayorProduccion()**: Recibirá la matriz y devolverá cuál es el día más productivo, considerando la producción de todas las fábricas. Mostrar el resultado por pantalla.
E. Función **bajaProduccion()**: Recibirá la matriz y creará, mediante lista por comprensión, una lista que contenga la menor cantidad fabricada por cada fábrica. La función devolverá la lista. Mostrar por pantalla un listado con los datos obtenidos.

NOTA: Salvo que se pida explícitamente, NO incluir los input() ni print() dentro de las funciones. Tanto input() como print() sólo deben codificarse dentro de main(), es decir, se debe enviar a las funciones los datos del input vía argumentos/parámetros, y recibir los resultados a imprimir desde las funciones vía return.

TP03-04 | CINE DE BARRIO (MENÚ)

Desarrollar un programa con un menú de 5 opciones que permita realizar reservas en una sala de cine de barrio que tiene 30 filas con 15 butacas por cada fila. Desarrollar las siguientes funciones y utilizarlas en el mismo programa:

A. Función **cargarSala()**: Primero generar en el programa principal una matriz con valores 0. Luego, la función recibirá la matriz como parámetro y la cargará aleatoriamente con valores 1 para simular una sala con butacas ya reservadas (es decir, 0 = libre; 1 = reservada)
B. Función **mostrarButacas()**: Recibirá la matriz y mostrará por pantalla (de forma rectangular) el estado de las butacas del cine. Para una mejor representación visual, cuando se trate de un 0 mostrar un * (asterisco) y cuando se trate de un 1 mostrar una R.
C. Función **reservar()**: Primero, desde el programa principal se deberá invocar a la función anterior para mostrar las butacas que están libres. Luego se deberá solicitar la posición a reservar. Entonces se llamará a la función reservar() que recibirá la matriz y las coordenadas de la butaca seleccionada, y actualizará la matriz en caso de estar disponible dicha butaca. La función devolverá además True/False si logró o no reservar la butaca, es decir si estaba libre o no. Si se logró reservar se llamará nuevamente a la función

mostrarButacas() para comprobar visualmente la reserva. Si no se logró reservar (porque estaba ocupada) entonces se mostrará un mensaje advirtiendo sobre esta situación.

D. Función **butacasLibres()**: Recibirá como parámetro la matriz y retornará cuántas butacas desocupadas hay en la sala. Mostrar por pantalla un mensaje con el dato obtenido.

E. Función **butacasContiguas()**: Esta función buscará un espacio con la secuencia más larga de butacas libres contiguas en una misma fila y devolverá las coordenadas de inicio del espacio y de cuantas butacas se trata. Mostrar por pantalla un mensaje con los datos obtenidos.

NOTA: Salvo que se pida explícitamente, NO incluir los input() ni print() dentro de las funciones. Tanto input() como print() sólo deben codificarse dentro de main(), es decir, se debe enviar a las funciones los datos del input vía argumentos/parámetros, y recibir los resultados a imprimir desde las funciones vía return.

TRABAJO PRÁCTICO 04 | CADENAS de CARACTERES

TP04-01 | CONTRASEÑAS INTERCALADAS

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.

TP04-02 | RELOJ

Desarrollar un programa que pida un valor de hora, un valor de minuto, y un valor de segundo. A partir de esos valores mostrar un reloj digital en formato de display HH:MM:SS (cada valor siempre en 2 dígitos). El display deberá avanzar cada 1 segundo como cualquier reloj digital (es decir que cuando los segundos superen los 59 volverán a 00 y se agregará un minuto, etc. Y lo mismo entre los minutos y las horas)

TP04-03 | TEMPORIZADOR

Desarrollar un programa que pida un valor de minuto, y un valor de segundo. A partir de esos valores mostrar un reloj digital en formato de display MM:SS (cada valor siempre en 2 dígitos). El display deberá ir en cuenta regresiva cada 1 segundo hasta llegar a 00:00. Cuando llegue a cero deberá detenerse y mostrar el mensaje “<<<< TIEMPO >>>>”

TP04-04 | FILTRADO DE PALABRAS

Escribir una función filtrarPalabras() 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 ciclos normales y slicing. Sin utilizar el método split()
B. Utilizando el método split() y ciclos normales
C. Utilizando el método split() y listas por comprensión

TP04-05 | PUNTO CADA 3 DÍGITOS

Escribir una función que reciba una cadena que contiene un número entero de muchos dígitos y devuelva una cadena con el mismo número, pero con los puntos de las separaciones de miles. Por ejemplo, si recibe 1234567890, debe devolver 1.234.567.890

TP04-06 | FUNCIÓN ELIMINAR SUBCADENA

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

TP04-7 | PALABRAS DE FRASE ORDENADAS

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.

TP04-8 | VOCALES ARRIBA

Se está desarrollando una importante app para tratamiento de texto y nos piden que desarrollemos una función para una de las opciones de la app. La función consiste en poner en mayúscula todas las vocales de una frase, por ejemplo, si la función recibe el texto “frase de prueba para el nuevo programa de tratamiento de texto” debe devolver “frAsE dE prUEbA pArA El nUEvO prOgrAmA dE trAtAmIEntO dE tExtO”. Probar la función desde un programa principal

TP04-9 | VERIFICACIÓN DE DIRECCIÓN DE EMAIL

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.