CARRERA DE: INGENIERÍA INFORMÁTICA
MATERIA: PARADIGMA DE PROGRAMACIÓN II
FORMACIÓN TEÓRICA: GUÍA DE EJERCICIOS

GUÍA DE EJERCICIOS

TEMA: INTRODUCCIÓN

OBJETIVO: Familiarizarse con el lenguaje utilizando los conceptos mas importante del paradigma estructurado. Mostrar su utilidad y comprobar su funcionamiento.

1- Crear un modulo llamado fechas que contenga una función crear_mes que reciba tres parámetros: nombre, abreviatura y cantidad de días. Cada vez que se invoca, dicha función debe imprimir por pantalla la información. Por ejemplo:

```
crear_mes(“Enero”, “ENE”, 31)


Se creó el mes Enero, que se abrevia ENE y contiene 31 días
```

2- Modificar la función anterior de forma tal que en vez de imprimir, genere y devuelva el string formateado. Luego, al ser invocada, imprimir el resultado. (no debería sufrir ningún cambio visual).

```
print(crear_mes(“Enero”, “ENE”, 31)


Se creó el mes Enero, que se abrevia ENE y contiene 31 días
```

3- Crear un programa que importe el modulo creado hasta el momento (fechas) y le pida al usuario los 3 datos: nombre del mes, abreviatura y cantidad de días. Una vez obtenida la información por parte del usuario.

Tip: función input y analizar que tipo de variable es con type

```
Ingrese el nombre del mes: Enero
Ingrese la abreviatura: ENE
Ingrese la cantidad de días: 31
Se creó el mes Enero, que se abrevia ENE y contiene 31 días.
```

4- Crear un programa que reciba del usuario la información de una fecha en formato dd/mm/aaaa y luego mostrar la información desglosada.

Tip: analizar la función split aplicada a un string, donde se le especifica un separador. Como resultado, se obtiene una lista (se puede verificar con type).

```
Ingrese una fecha en formato dd/mm/aaaa: 6/12/1986
El día ingresado fue:  6
El mes ingresado fue:  12
El año ingresado fue:  1986
```

5- Realizar un programa que permita el usuario ingresar las coordenadas (x,y) de puntos del plano (se indica el fin con el par 0,0). Luego, imprimir las coordenadas ingresadas y la distancia al origen, ordenadas de menor a mayor por distancia. Utilizar funciones y módulos.

Tips: el modulo math contiene la función sqrt que calcula la raíz cuadradra. Para la obtención de los datos, se puede trabajar con la función split vista anteriormente.

$$|a,b| = \sqrt{a^2 + b^2}$$

```
$ python3 ej05.py

Por favor, ingrese el punto en formato x,y : 1,0
La distancia al origen del punto (1,0) es: 1.0
Por favor, ingrese el punto en formato x,y : 1,1
La distancia al origen del punto (1,1) es: 1.4142135623730951
Por favor, ingrese el punto en formato x,y : 0,1
La distancia al origen del punto (0,1) es: 1.0
Por favor, ingrese el punto en formato x,y : 1231,123123
La distancia al origen del punto (1231,123123) es: 123129.15369643373
Por favor, ingrese el punto en formato x,y : 0,0
La distancia al origen del punto (0,0) es: 0.0
Programa finalizado
```

6- Un supermercado requiere un sistema de facturación que pueda calcular los totales de las facturas que realizan las cajas registradoras. Cada caja ingresa el código del producto (numérico) y tiene una base de productos, ya codificada en la lista contenida del archivo `productos.py`

Se pide realizar los siguientes requerimientos solicitados por el cliente:

a. Presentar un menú permitiendo al usuario elegir una de las siguientes opciones: 1) Buscar producto 2) Ingresar nueva factura 3) Ingresar nuevo producto al catalogo 3) Salir

```
$ python ej06.py

     1) Buscar producto
     1) Ingresar nueva factura
     2) Ingresar nuevo producto
     3) Salir

    Por favor, seleccione la opcion:
    3
```

b. Realizar función llamada buscar en el modulo productos.py de forma tal de recibir por parámetro un código numérico de producto y retornar el mismo, utilizando la lista de diccionarios implementada. Por ejemplo

```
import productos

print(productos.buscar(1233))
```

De como resultado: {'codigo': 1233, 'desc': 'Arroz Integral 1 kilo', 'precio': 66.33}

c. Juntar los dos puntos anteriores en el programa, de forma tal de implementar correctamente la opción 1, que permita al usuario buscar un producto y mostrar la información de este (se permite hacer `print`, pero puede también formatearse mediante `format`).

d. Continuar implementando la opción 2, la cual debe solicitar al usuario cada código de producto. Si el producto se encuentra en la lista, preguntar la cantidad. Si no se encuentra en el catalogo, informarlo por pantalla para volver a reingresar otro código de producto. Si ingresa 'F' se considera que finaliza la factura. Al finalizar, mostrar la factura, producto por producto, con la cantidad y el total de la factura.

   Tips: Repasar el tipo de datos lista (mediante la creación con `list()` o con `[]`). Cada factura debe ser una lista nueva. Y mediante las operaciones de agregado (append) ir completándola.

e. Con la opción de agregar nuevo producto al catalogo, debe solicitarle los datos de descripción, y precio unitario con dos decimales. Luego agregarla al catalogo. Se aprecia si para este se crea una nueva función agregar en el modulo de productos.

Utilizar módulos permitiendo la reutilización del código y no debe encontrarse toda la lógica en un solo archivo.

Tips: Se agrega a continuación un ejemplo de obtención del catalogo dentro del modulo productos y su utilización, imprimiendo toda la información del mismo y utilizando la función `format` para incluir cada campo de los productos

```
import productos

for i in productos.catalogo:
    print("Cod: {} - Desc: {} - Precio: {}".format(i['codigo'], i['desc'], i['precio']))
```