



Introducción a la Programación Segura

Colecciones y librerías en Python.



INTRODUCCIÓN A LA PROGRAMACIÓN SEGURA

UNIDAD 3: COLECCIONES Y LIBRERÍAS EN PYTHON.

Contenidos

1. Uso e importación de módulos.
2. Módulos estándares.
3. Módulos y paquetes.
4. Errores, fallas y excepciones.
5. Caracteres y Cadenas.



UNIDAD DE APRENDIZAJE

UNIDAD 3: COLECCIONES Y LIBRERÍAS EN PYTHON.

Aprendizaje Esperado:

3.1. Utiliza estructuras de almacenamiento de datos de Python, para hacer más eficiente el código de programación, considerando el desarrollo de scripts y librerías asociadas a la seguridad.



A practicar...

Gestión de una Biblioteca

- **Descripción:** Se necesita desarrollar un programa para gestionar los libros de una biblioteca. El programa debe permitir agregar nuevos libros, prestar libros a los usuarios, devolver libros y mostrar el estado actual de los libros en la biblioteca.
- **Entrada :** Una lista de diccionarios, donde cada diccionario representa un libro y contiene las siguientes claves: "título", "autor", "disponible" (indicando si el libro está disponible para prestar o no). Opciones del usuario para agregar un nuevo libro, prestar un libro, devolver un libro o mostrar el estado actual de los libros.
- **Salida:** Dependiendo de la opción seleccionada por el usuario, se puede mostrar el estado actualizado de los libros en la biblioteca o un mensaje de confirmación de la operación realizada.

Pasos:

1. Utilizar un bucle iterativo para mostrar un menú de opciones al usuario.
2. Según la opción seleccionada por el usuario, realizar las operaciones correspondientes: agregar un nuevo libro, prestar un libro, devolver un libro o mostrar el estado actual de los libros.
3. Para agregar un nuevo libro, solicitar al usuario que ingrese el título y el autor del libro y agregarlo a la lista de libros con el estado "disponible".
4. Para prestar un libro, buscar el libro por su título y marcarlo como "no disponible".
5. Para devolver un libro, buscar el libro por su título y marcarlo como "disponible".
6. Para mostrar el estado actual de los libros, imprimir la lista de libros con su estado de disponibilidad.

Restricciones:

1. Los títulos de los libros deben ser únicos.
2. Los nombres de los autores pueden repetirse.
3. El programa debe manejar adecuadamente casos donde el usuario intente prestar un libro que ya está prestado o devolver un libro que ya está disponible.

Uso e importación de módulos:

¿Qué es un Módulo?

En Python, un módulo es **un archivo que contiene definiciones y declaraciones de funciones, variables y clases**. Los módulos permiten organizar el código en archivos separados y reutilizarlo en otros programas. Puedes utilizar módulos estándar de Python, módulos de la comunidad o incluso crear tus propios módulos personalizados.

```
import math  
  
resultado = math.sqrt(25)  
print(resultado)
```

5.0

En este caso la función **sqrt** está disponible gracias a la importación del **módulo math**.

Uso e importación de módulos:

Utilizando Módulos:

Para utilizar un módulo en Python, primero debes importarlo en tu programa:

```
import math

resultado = math.sqrt(25)
print(resultado)
```

Podemos **importar todo el contenido** de un módulo usando la **palabra clave import** seguida del **nombre del módulo**.

Luego, puedes acceder a las funciones, variables o clases del módulo usando la sintaxis **nombreModulo.elemento**

Podemos **importar solo funciones**, variables o clases **específicas** de un módulo **usando la palabra clave from**. Esto te permite acceder a esos elementos directamente sin usar la notación de punto.

```
from math import sqrt

resultado = sqrt(25)
print(resultado)
```

Uso e importación de módulos:

Utilizando Módulos:

Para utilizar un módulo en Python, primero debes importarlo en tu programa:

```
import math as m

resultado = m.sqrt(25)
print(resultado)
```

Podemos **importar todo el** módulo, pero al mismo tiempo asignarle un alias, es decir, un nombre más corto usando la palabra clave **as**.

Uso e importación de módulos:

Utilizando Módulos:

Además de los módulos estándar de Python, también puedes usar módulos desarrollados por la comunidad de Python o módulos personalizados. **Para usar módulos de terceros, primero debes instalarlos en tu entorno Python.** Esto se puede hacer usando administradores de paquetes como **pip**.

```
Símbolo del sistema
C:\>pip install requests
```

Una vez que instalado el módulo de terceros, puedes importarlo en tu programa y utilizarlo de la misma manera que los módulos estándar de Python.

```
import requests

respuesta = requests.get("https://mindicador.cl/api/dolar/01-07-2023")
print(respuesta.json())
```

```
{'version': '1.7.0', 'autor': 'mindicador.cl', 'codigo': 'dolar', 'nombre': 'Dólar observado', 'unidad_medida': 'Pesos', 'serie': []}
```

Uso e importación de módulos:

Creando nuestros propios Módulos:

También podemos crear nuestros propios módulos personalizados para organizar y reutilizar nuestro código. Para hacerlo, simplemente **creamos un archivo con extensión .py** que contenga las definiciones de funciones, variables o clases. Luego, podremos **importar ese archivo como un módulo** en otros programas.

miModulo.py

```
miModulo.py > ...  
1  def saludar(nombre):  
2      return 'Hola ' + nombre + ' te saludo desde mi Módulo'  
3  
4  def hastaPronto(nombre):  
5      return 'hasta pronto ' + nombre + ' me despido desde mi Módulo'
```

archivoQueEjecutaElCódigo.py

```
import miModulo  
  
print(miModulo.saludar('Alex'))  
print(miModulo.hastaPronto('Alex'))
```

```
Hola Alex te saludo desde mi Módulo  
hasta pronto Alex me despido desde mi Módulo
```

Módulos estándares:

Son **conjuntos de bibliotecas y paquetes que vienen incluidos con la instalación predeterminada de Python**. Estos módulos proporcionan una amplia variedad de funcionalidades y herramientas para realizar tareas comunes como operaciones matemáticas, manejo de archivos, acceso a Internet, procesamiento de cadenas, entre muchas otras cosas.

Algunas de ellas son:

- Math
- Datetime
- Json
- Csv
- Os
- Random
- Urllib
- Re

Módulos estándares:

math: Proporciona funciones matemáticas avanzadas, como funciones trigonométricas, exponenciales, logarítmicas, entre otras.

```
import math

resultado = math.sqrt(25)
print(resultado)
```

5.0

os: Permite interactuar con el sistema operativo. Proporciona funciones para trabajar con archivos, directorios, rutas y otras operaciones relacionadas con el sistema.

```
import os

directorio_actual = os.getcwd()
print("Directorio actual:", directorio_actual)
```

Directorio actual: D:\Clases\TÉCNICAS DE PROGRAMACIÓN EN PYTHON\Codigos

datetime : Ofrece clases y funciones para trabajar con fechas y horas.

```
from datetime import datetime

fecha_hora_actual = datetime.now()
print("Fecha y hora actual:", fecha_hora_actual)
```

Fecha y hora actual: 2023-08-07 18:09:54.893680

Módulos estándares:

random: Proporciona funciones para generar números aleatorios.

```
import random

numero_aleatorio = random.randint(1, 10)
print("Número aleatorio entre 1 y 10:", numero_aleatorio)
```

Número aleatorio entre 1 y 10: 6

json: Permite trabajar con formato de datos JSON, que es comúnmente utilizado para el intercambio de datos en aplicaciones web y APIs.

```
import json

datos = {"nombre": "Teresa", "edad": 26, "ciudad": "La Serena"}

# Convertir el diccionario a formato JSON
json_datos = json.dumps(datos)
print("Datos en formato JSON:", json_datos)
```

Datos en formato JSON: {"nombre": "Teresa", "edad": 26, "ciudad": "La Serena"}

urllib : Facilita la apertura de URLs y la realización de solicitudes HTTP.

csv : Ofrece funcionalidades para trabajar con archivos CSV (Comma-Separated Values), que son muy comunes en el manejo de datos tabulares.

re : Proporciona operaciones de expresiones regulares para el procesamiento de cadenas.

Funciones de módulo math:

Devuelve la raíz cuadrada de x.

```
import math  
  
resultado = math.sqrt(25)  
print(resultado)
```

5.0

Devuelve el entero más pequeño mayor o igual que x.

```
import math  
  
numero = 7.8  
resultado = math.ceil(numero)  
print(resultado)
```

8

Devuelve el entero más grande menor o igual que x.

```
import math  
  
numero = 7.8  
resultado = math.floor(numero)  
print(resultado)
```

7

Devuelve x elevado a la potencia de y.

```
import math  
  
base = 2  
exponente = 8  
resultado = math.pow(base, exponente)  
print(resultado)
```

256.0

Devuelve el logaritmo natural de x.

```
import math  
  
numero = 10  
logaritmo_natural = math.log(numero)  
print(logaritmo_natural)
```

2.302585092994046

Funciones de módulo math:

Devuelven el seno, coseno y tangente de x.

```
import math

angulo_radianes = math.radians(45)
seno = math.sin(angulo_radianes)
coseno = math.cos(angulo_radianes)
tangente = math.tan(angulo_radianes)

print(seno)
print(coseno)
print(tangente)
```

```
0.7071067811865476
0.7071067811865476
0.9999999999999999
```

Devuelve el factorial de x.

```
import math

numero = 5
factorial = math.factorial(numero)
print(factorial)
```

```
120
```

Devuelve el máximo común divisor de a y b.

```
import math

num1 = 24
num2 = 36
maximo_comun_divisor = math.gcd(num1, num2)
print(maximo_comun_divisor)
```

```
12
```

Devuelve la raíz cuadrada entera de x.

```
import math

numero = 25
raiz_cuadrada_entera = math.isqrt(numero)
print(raiz_cuadrada_entera)
```

```
5
```

Funciones de módulo random:

Devuelve un número de punto flotante aleatorio en rango [0.0, 1.0).

```
import random

numero_aleatorio = random.random()
print(numero_aleatorio)
```

0.06511271175633793

Devuelve un número entero aleatorio en el rango [a, b], ambos inclusivos.

```
import random

numero_aleatorio = random.randint(1, 10)
print(numero_aleatorio)
```

7

Devuelve un elemento aleatorio de la lista dada.

```
import random

nombres = ['Pedro', 'Camila', 'Yanet', 'Oscar']
nombre_aleatoria = random.choice(nombres)
print(nombre_aleatoria)
```

Camila

Mezcla los elementos de la lista en su lugar, cambiando el orden de los elementos de manera aleatoria.

```
import random

nombres = ['Pedro', 'Camila', 'Yanet', 'Oscar']
random.shuffle(nombres)
print(nombres)
```

['Yanet', 'Oscar', 'Pedro', 'Camila']

Funciones de módulo random:

Devuelve una lista con elementos seleccionados al azar de la lista dada, sin repetición.

```
import random

nombres = ['Pedro', 'Camila', 'Yanet', 'Oscar']
nombres_aleatorios = random.sample(nombres, 2)
print(nombres_aleatorios)
```

['Pedro', 'Oscar']

Devuelve un número de punto flotante aleatorio en el rango [a, b).

```
import random

numero_aleatorio = random.uniform(1.0, 10.0)
print(numero_aleatorio)
```

6.114490235843501

Funciones de módulo platform:

Devuelve el nombre del sistema operativo.

```
import platform

sistema_operativo = platform.system()
print(sistema_operativo)
```

Windows

Devuelve la versión del sistema operativo.

```
import platform

version_sistema_operativo = platform.release()
print(version_sistema_operativo)
```

10

Devuelve la versión de Python que se está ejecutando.

```
import platform

version_python = platform.python_version()
print(version_python)
```

3.11.4

Funciones de módulo platform:

Devuelve el nombre del procesador.

```
import platform  
  
procesador = platform.processor()  
print(procesador)
```

```
Intel64 Family 6 Model 165 Stepping 2, GenuineIntel
```

Devuelve la arquitectura de la máquina.

```
import platform  
  
arquitectura = platform.machine()  
print(arquitectura)
```

```
AMD64
```

Módulos y paquetes:

Módulos:

Un módulo en Python **es un archivo que contiene definiciones de funciones, clases y variables** que se pueden utilizar en otros programas Python. Los módulos **permiten organizar y reutilizar código**, lo que ayuda a mantener los programas más limpios y legibles.

miModulo.py

```
miModulo.py > ...  
1 def saludar(nombre):  
2     return 'Hola ' + nombre + ' te saludo desde mi Módulo'  
3  
4 def hastaPronto(nombre):  
5     return 'hasta pronto ' + nombre + ' me despido desde mi Módulo'
```

archivoQueEjecutaElCódigo.py

```
import miModulo  
  
print(miModulo.saludar('Alex'))  
print(miModulo.hastaPronto('Alex'))
```

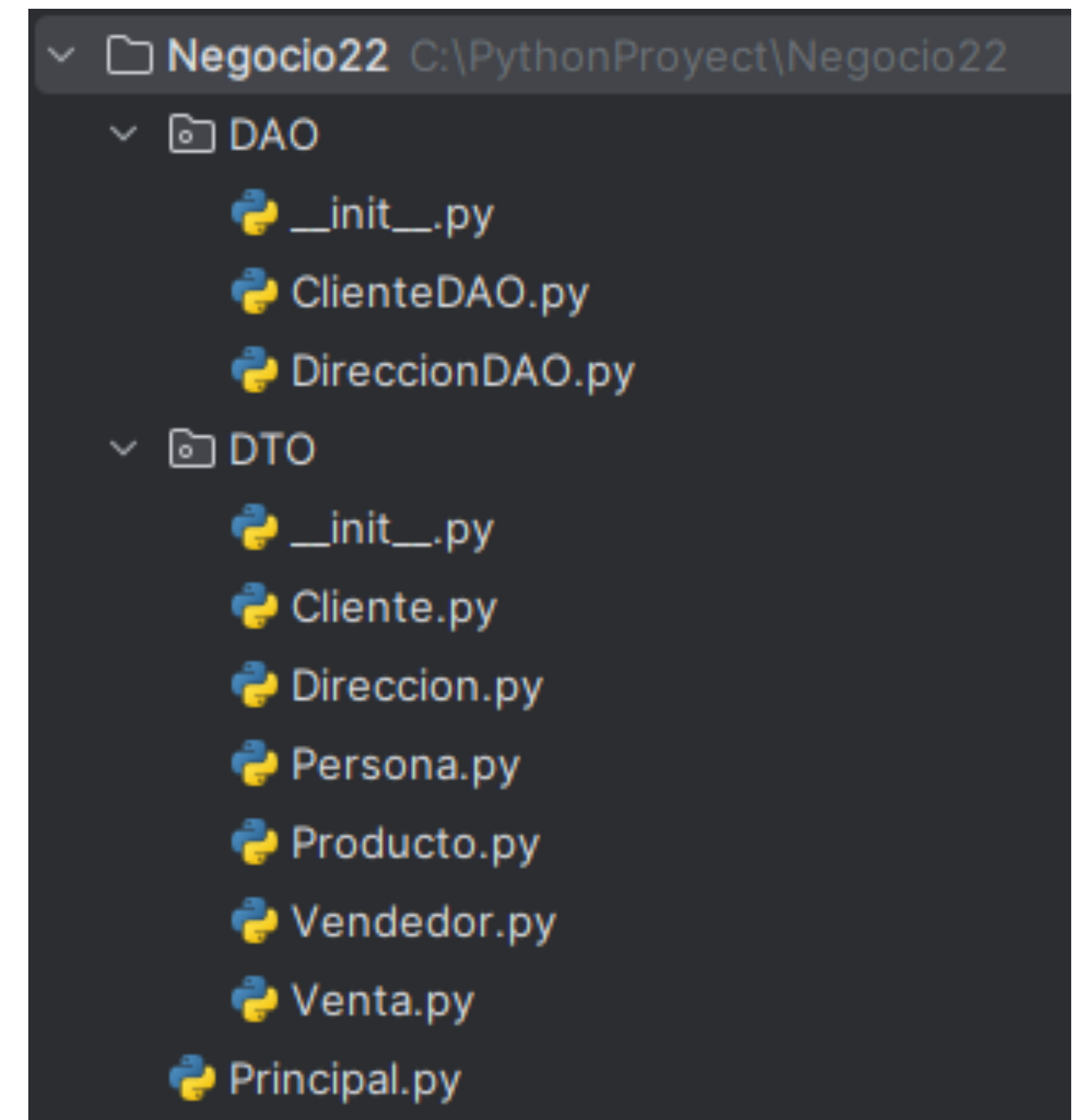
Hola Alex te saludo desde mi Módulo
hasta pronto Alex me despido desde mi Módulo

Módulos y paquetes:

Paquetes:

Un paquete en Python es simplemente **una colección de módulos organizados en una jerarquía de directorios**. Los paquetes permiten organizar y distribuir módulos de manera más eficiente y estructurada. Un paquete **es un directorio que contiene un archivo especial llamado "init.py"**. Este **archivo** es **necesario para que Python reconozca** el directorio **como un paquete**.

Dentro de un paquete, puedes tener módulos y subpaquetes (otros directorios que contienen su propio "init.py" y otros módulos). Los paquetes pueden ser útiles cuando tienes un conjunto de módulos relacionados que deseas agrupar y organizar.



Errores, fallas y excepciones:

Los errores, fallas y excepciones son situaciones en las que el programa no puede continuar su ejecución normal debido a problemas en el código. Estos pueden ocurrir por diversos motivos, como errores de sintaxis, problemas lógicos o situaciones excepcionales durante la ejecución.

Errores de Sintaxis:

Ocurren cuando el código no respeta las reglas de sintaxis del lenguaje. Python no puede comprender el código incorrecto y mostrará un mensaje de error indicando la ubicación del problema. Un error de sintaxis suele ser detectado antes de que el programa se ejecute.

```
print "Bienvenido"
```

```
print "Bienvenido"  
^^^^^^^^^^^^^^^^  
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

Errores, fallas y excepciones:

Fallas y Excepciones:

Las fallas son errores más graves que causan que el programa termine inesperadamente. Estas pueden ser causadas por problemas en el sistema, como falta de recursos, interrupciones inesperadas o fallos en la interpretación del lenguaje. Por otro parte, tenemos las excepciones, que son situaciones anormales que pueden ocurrir durante la ejecución de un programa, como divisiones entre cero, acceso a índices inválidos en listas, etc.

Python proporciona un sistema de manejo de excepciones para capturar y manejar estas situaciones, evitando que el programa se detenga abruptamente.

Sin manejar la
Excepción

```
resultado = 8 / 0  
print(resultado)
```

```
resultado = 8 / 0  
~~~~~  
ZeroDivisionError: division by zero
```

Manejando la
Excepción

```
try:  
    resultado = 8 / 0  
    print(resultado)  
except ZeroDivisionError:  
    print("Error: No puedes dividir entre cero")
```

```
Error: No puedes dividir entre cero
```


Errores, fallas y excepciones:

Manejo de excepciones:

Se produce cuando una función recibe un argumento con el tipo correcto pero un valor inapropiado.

```
try:
    numero = int("tres")
except ValueError:
    print("Error: No se puede convertir a entero")
```

Error: No se puede convertir a entero

Se produce cuando se intenta acceder a un índice inválido en una lista, tupla o cadena.

```
try:
    lista = [15, 8, 24]
    elemento = lista[6]
except IndexError:
    print("Error: Índice fuera de rango")
```

Error: Índice fuera de rango

Se produce cuando se realiza una operación o función en un objeto de un tipo no adecuado.

```
try:
    resultado = "12" + 8
except TypeError:
    print("Error: No se puede realizar la operación con estos tipos")
```

Error: No se puede realizar la operación con estos tipos

Errores, fallas y excepciones:

Manejo de excepciones:

Se produce cuando se intenta acceder a una clave que no existe en un diccionario.

```
try:
    diccionario = {"nombre": "Alex", "edad": 35}
    valor = diccionario["fono"]
except KeyError:
    print("Error: La clave no existe en el diccionario")
```

Error: La clave no existe en el diccionario

Se produce cuando se intenta abrir un archivo que no existe.

```
try:
    archivo = open("archivo_no_existente.txt", "r")
except FileNotFoundError:
    print("Error: El archivo no se encuentra")
```

Error: El archivo no se encuentra

Caracteres y Cadenas:

Métodos de Cadenas:

Convierten una cadena a mayúsculas.

```
saludo = 'Buenas tardes Teresa'  
print(saludo.upper())
```

BUENAS TARDES TERESA

Convierten una cadena a minúsculas.

```
saludo = 'Buenas tardes Teresa'  
print(saludo.lower())
```

buenas tardes teresa

Divide la cadena en una lista de subcadenas usando un separador.

```
cadena = "Manzana,Naranja,Plátano"  
print(cadena.split(","))
```

['Manzana', 'Naranja', 'Plátano']

Elimina espacios en blanco al principio y al final de la cadena

```
saludo = '    Buenas tardes Teresa    '  
print(saludo.strip() )
```

Buenas tardes Teresa

Caracteres y Cadenas:

Cadenas en Acción:

Las cadenas se pueden comparar utilizando operadores de comparación, como `==`, `<`, `>`.

```
cadena1 = "manzana"  
cadena2 = "Manzana"  
  
print(cadena1 == cadena2)
```

False

Las cadenas se pueden ordenar alfabéticamente.

```
frutas = ["Plátano", "Naranja", "Manzana"]  
frutas.sort()  
  
print(frutas)
```

['Manzana', 'Naranja', 'Plátano']

```
nombre = "Teresa"  
apellido = "Rojas"  
concatenacion = nombre + " " + apellido  
multiplicacion = nombre * 3  
  
print(concatenacion)  
print(multiplicacion)
```

Teresa Rojas
TeresaTeresaTeresa

Ten en cuenta que las operaciones matemáticas en cadenas tienen un significado diferente. Por ejemplo, la concatenación y la multiplicación.



A practicar...

Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

1. Escribe un programa que importe el módulo math y calcule la raíz cuadrada de un número ingresado por el usuario. Luego, muestra el resultado por pantalla.

```
import math

numero = int(input('Ingrese un número entero: '))
raiz_cuadrada = math.sqrt(numero)
print("La raíz cuadrada de", numero, "es:", raiz_cuadrada)
```

```
Ingrese un número entero: 16
La raíz cuadrada de 16 es: 4.0
```



Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

2. Crea un programa que importe el módulo datetime y muestre la fecha y hora actual en el siguiente formato: "Año-Mes-Día
Hora : Minuto : Segundo".

```
import datetime  
  
fecha_actual = datetime.datetime.now()  
print("La fecha y hora actual es:", fecha_actual)
```

```
La fecha y hora actual es: 2023-08-07 22:09:06.217272
```



Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

3. Desarrolla un programa que utilice el módulo math para calcular el seno de un ángulo en grados. El programa debe solicitar el ángulo al usuario y mostrar el resultado.

```
import math

angulo = int(input('Ingrese el valor del ángulo: '))
seno = math.sin(math.radians(angulo))
print("El seno de", angulo, "grados es:", seno)
```

```
Ingrese el valor del ángulo: 45
El seno de 45 grados es: 0.7071067811865476
```



Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

4. Crea un programa que importe el módulo platform y muestre el sistema operativo en el que se está ejecutando.

```
import platform

sistema_operativo = platform.system()
print("Sistema operativo:", sistema_operativo)
```

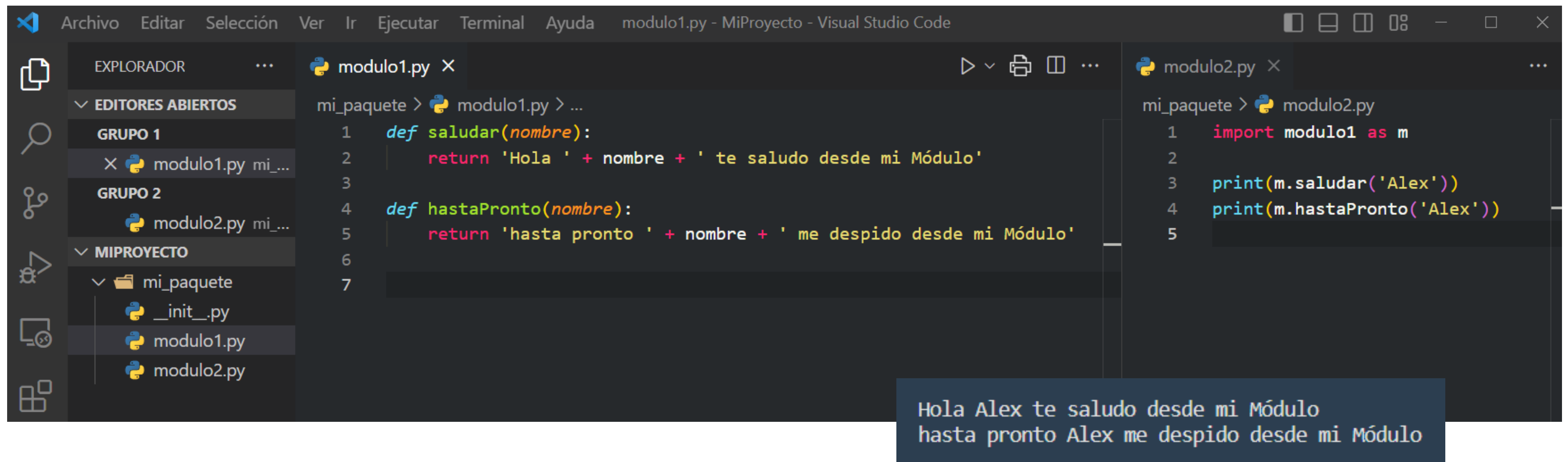
```
Sistema operativo: Windows
```



Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

5. Organiza un paquete llamado "mi_paquete" que contenga dos módulos: "modulo1.py" y "modulo2.py". En cada módulo, define una función simple y luego importa y utiliza estas funciones en un programa principal.



The screenshot shows the Visual Studio Code interface with a project named "MiProyecto". The Explorer sidebar on the left shows the file structure: a folder named "mi_paquete" containing three files: "__init__.py", "modulo1.py", and "modulo2.py". The Editor pane shows two files open. The active file is "modulo1.py", which contains the following code:

```
mi_paquete > Python modulo1.py > ...
1 def saludar(nombre):
2     return 'Hola ' + nombre + ' te saludo desde mi Módulo'
3
4 def hastaPronto(nombre):
5     return 'hasta pronto ' + nombre + ' me despido desde mi Módulo'
6
7
```

The second file, "modulo2.py", contains the following code:

```
mi_paquete > Python modulo2.py
1 import modulo1 as m
2
3 print(m.saludar('Alex'))
4 print(m.hastaPronto('Alex'))
5
```

Below the code editor, a dark blue box displays the output of the program:

```
Hola Alex te saludo desde mi Módulo
hasta pronto Alex me despido desde mi Módulo
```



Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

6. Crea un programa que tome una cadena como entrada y la convierta a mayúsculas utilizando para luego mostrar la cadena en mayúsculas.

```
cadena = input('Ingrese una cadena de texto: ')\n\nmayusculas = cadena.upper()\nprint("Mayúsculas:", mayusculas)
```

```
○ Ingrese una cadena de texto: teclado\n  Mayúsculas: TECLADO
```



Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

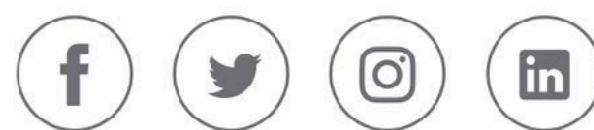
7. Desarrolla un programa que solicite al usuario el ingreso de dos cadenas, las compare y determine si son iguales o no. Luego, muestra el resultado de la comparación por pantalla.

```
cadena1 = input('Ingrese la primera cadena de texto: ')\ncadena2 = input('Ingrese la segunda cadena de texto: ')\n\ncomparacion = cadena1 == cadena2\n\nprint("¿Las cadenas son iguales?", comparacion)
```

```
Ingrese la primera cadena de texto: Python\nIngrese la segunda cadena de texto: python\n¿Las cadenas son iguales? False
```

```
Ingrese la primera cadena de texto: Python\nIngrese la segunda cadena de texto: Python\n¿Las cadenas son iguales? True
```





inacap.cl