



## PRÁCTICO

**Nombre del Grupo: “*Punteros Letales*”**

Jorge Luis Esteves Salas  
Fernando Navia  
Joel Dalton Montero  
Ana Laura Cuellar  
Weimar Valda  
Leonel Eguez Camargo  
ALEJANDRO HURTADO

***Dirigido por el docente:***  
***JIMMY REQUENA LLORENTY***

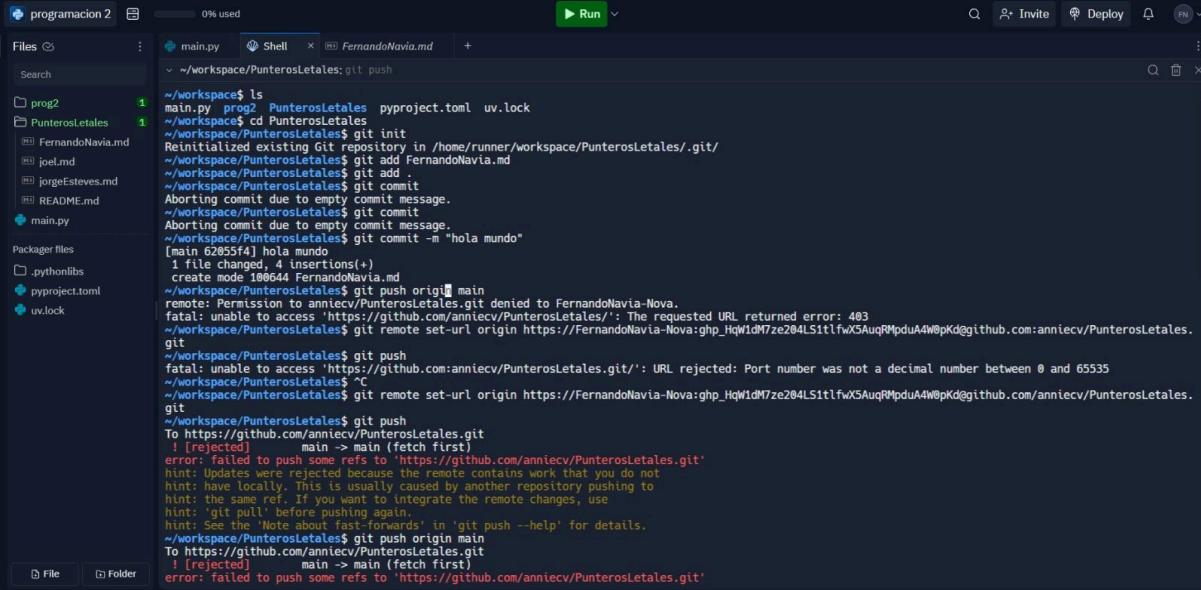
***Materia:***  
***Programación II***

## PRÁCTICO 1

JORGE ESTEVES

06/07/2025 17:30 APROXIMADAMENTE

PRIMER EJERCICIO REPLY



```
programacion 2 0% used Run
Files
Search
prog2
PunterosLetales
  FernandoNavia.md
  joel.md
  jorgeEsteves.md
  README.md
  main.py
Package files
  .pythonlibs
  pyproject.toml
  uv.lock

~/workspace/PunterosLetales: git push
~/workspace$ ls
main.py prog2 PunterosLetales pyproject.toml uv.lock
~/workspace$ cd PunterosLetales
~/workspace/PunterosLetales$ git init
Reinitialized existing Git repository in /home/runner/workspace/PunterosLetales/.git/
~/workspace/PunterosLetales$ git add FernandoNavia.md
~/workspace/PunterosLetales$ git add .
~/workspace/PunterosLetales$ git commit
Aborting commit due to empty commit message.
~/workspace/PunterosLetales$ git commit
Aborting commit due to empty commit message.
~/workspace/PunterosLetales$ git commit -m "hola mundo"
[main 62855f4] hola mundo
1 file changed, 4 insertions(+)
create mode 100644 FernandoNavia.md
~/workspace/PunterosLetales$ git push origin main
remote: Permission to anniecv/PunterosLetales.git denied to FernandoNavia-Nova.
fatal: unable to access 'https://github.com/anniecv/PunterosLetales/': The requested URL returned error: 403
~/workspace/PunterosLetales$ git remote set-url origin https://FernandoNavia-Nova:ghp_HqW1dM7ze284LS1tlfwX5AuqRMpduA4W@pkd@github.com:anniecv/PunterosLetales.
git
~/workspace/PunterosLetales$ git push
fatal: unable to access 'https://github.com:anniecv/PunterosLetales.git/': URL rejected: Port number was not a decimal number between 0 and 65535
~/workspace/PunterosLetales$ ^C
~/workspace/PunterosLetales$ git remote set-url origin https://FernandoNavia-Nova:ghp_HqW1dM7ze284LS1tlfwX5AuqRMpduA4W@pkd@github.com:anniecv/PunterosLetales.
git
~/workspace/PunterosLetales$ git push
To https://github.com/anniecv/PunterosLetales.git
! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/anniecv/PunterosLetales.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
~/workspace/PunterosLetales$ git push origin main
To https://github.com/anniecv/PunterosLetales.git
! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/anniecv/PunterosLetales.git'
```

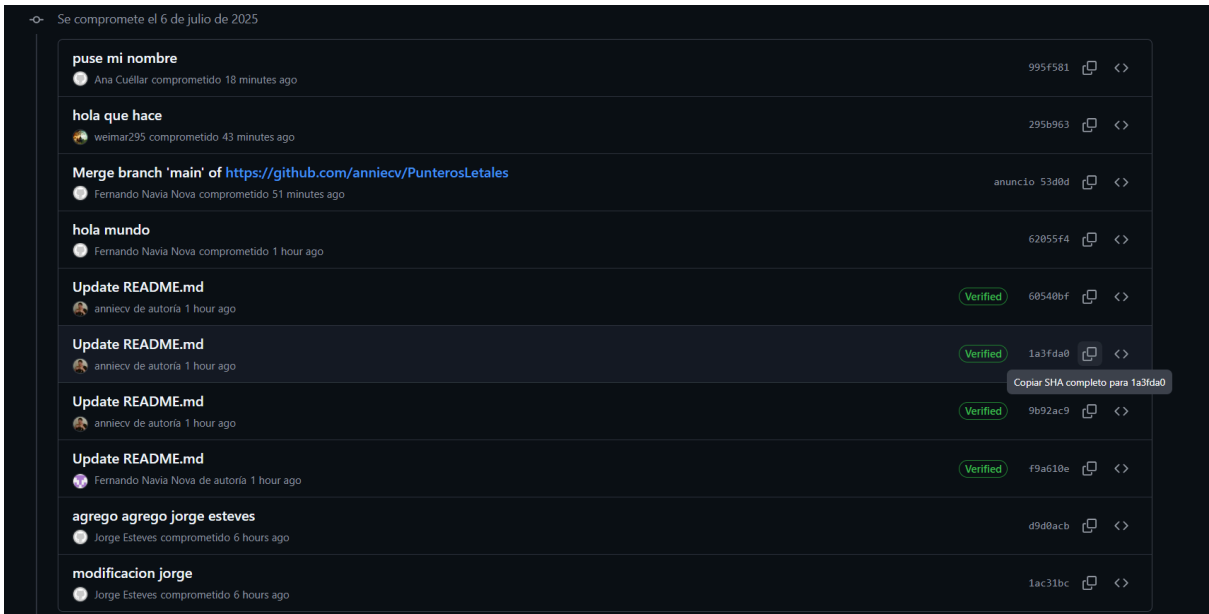
Se hicieron prueba con los diferentes códigos se de git como:

git init para inicializar x si no esta la extensión de git, git add

NOMBRE\_DE\_LO\_CREADO, git add . que nos dice que ya acabo de hacer cambios

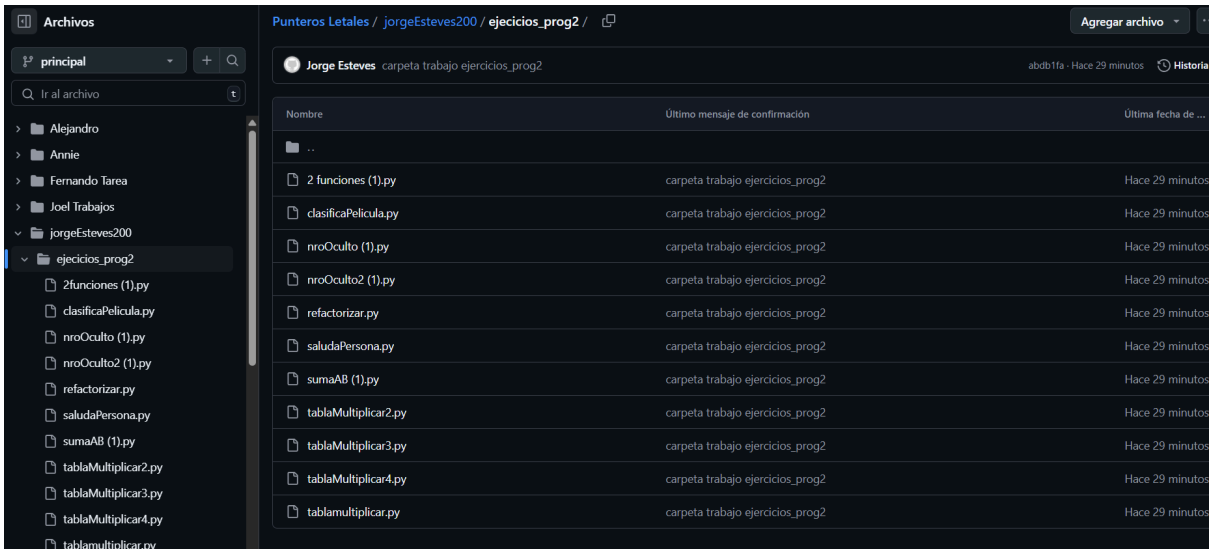
tuve varios errores con el token pero gracias al material de apoyo se encontró una

solución para esos pequeños errores .



Fecha y hora actual: PM-07-08 17:08:48

practica en clase



Se adiciono los diferentes archivos de y prácticas de clases en mi carpeta para seguimiento con el docente me quede sin internet y la sufrí tuvimos conflictos nuevamente

jorge ESTEVES r – 09/07/2025 – 17:53

Ejercicios practico p2uno,p2dos,p2tres

```
p2tablaMultiplicar.py +
PunterosLetales > jorgeEsteves200 > practicos > p2tablaMultiplicar.py > ...
1 num_tabla = int(input("Ingrese el número de la tabla de multiplicar: "))
2 print(f"---Tabla de multiplicar del {num_tabla}:---")
3 for i in range(1, 11):
4     resultado=num_tabla*i
5     print(f"{num_tabla} x {i} = {num_tabla * i}")
6 print ("---Fin del programa---jorge esteve---")
```

```
PunterosLetales > jorgeEsteves200 > practicos > p2HolaMundo.py
1 print ("hola soy Jorge Esteves" )
2
```

```
PunterosLetales > jorgeEsteves200 > practicos > p2adivinanza.py > ...
1
2 import random
3
4 numero_secreto = random.randint(1, 100)
5 num_azar = int(input("Ingresa un número al azar entre 1 y 100: "))
6
7 while num_azar != numero_secreto:
8     if num_azar < numero_secreto:
9         print("El número que ingresaste es menor al número secreto.")
10    else:
11        print("El número que ingresaste es mayor al número secreto.")
12
13    num_azar = int(input("Intenta de nuevo. Ingresa otro número: "))
14
15 print("¡Felicidades! Adivinaste el número.")
```

jorge esteves – 10/07/2025 – 18:05

## INVESTIGACIÓN

### QUÉ SON LOS ARGUMENTOS POR POSICIÓN VS LOS ARGUMENTOS POR NOMBRES

Son 2 formas diferentes de pasar valores a una función cada una diferente:

- Argumentos por Posición  
está definida la variable de acuerdo a su posición y orden en la función

está bien tipeada.

Python asocia cada valor con el parámetro **según su posición**.

ejemplo

```
def saludar(nombre, edad):  
    print(f"Hola, {nombre}. Tienes {edad} años.")
```

```
saludar("jorge", 25) # Argumentos por posición
```

- Argumento por nombre

son aquellas que al momento de nombrar se le da el valor especificado sin importar el orden

ejemplo

```
saludar(edad=25, nombre="Ana") # Argumentos por nombre
```

**Puedes combinar ambos (con cuidado)**

Primero van los argumentos por posición, luego los argumentos **por nombre**.

**CUAL ES LA DIFERENCIA Y CUANDO HAY QUE USAR UNO DE ESTOS**

**diferencias principales de cada uno**

Característica	Argumentos por posición	Argumentos por nombre (keyword)
Orden	Importa el orden	No importa el orden
Forma de escritura	Solo el valor	Se indica el nombre del parámetro

Claridad	Puede ser menos clara	Es más legible y autoexplicativa
Uso común	Rápido para pocos valores	Mejor para funciones con muchos parámetros o valores por defecto

jorge ESTEVES r – 09/07/2025 – 18:15

### Practica clase p3uno,p3dos,p3tres

```

Shell p3uno.py p3tres.py +
PunterosLetales > jorgeEsteves200 > practicos > p3uno.py > ...

1 #Lista donde se guardan las comidas
2 comida_fav = ["milaneza", "brownie", "ramen"]
3 print(comida_fav)
4
5 print("\nMi segunda comida favorita es " + comida_fav[1])
6 print("\nCambiar tu primera comida favorita")
7 nueva_comida = input("\nIngresa tu nueva comida favorita:")
8 comida_fav[0] = nueva_comida
9 print(comida_fav)
10 cantidad_comida = len(comida_fav)
11 print(f"\nLa cantidad de comidas favoritas que tienes es de {cantidad_comida}")

```

```

Shell p3uno.py p3tres.py +
PunterosLetales > jorgeEsteves200 > practicos > p3tres.py > ...

1 lista_nom = ["nom1", "nom2", "nom3", "nom4", "nom5"]
2
3 for i in lista_nom:
4     print(f"Bienvenido al equipo {i} ")
5

```

```

PunterosLetales > jorgeEsteves200 > practicos > p3dos.py > ...

1  mis_notas = [10, 10, 10, 10, 10]
2  sum_total = 0
3  for i in mis_notas:
4      sum_total = sum_total + i
5
6  promedio = sum_total / len(mis_notas)
7  print(f"La suma total de todas las notas es de: {sum_total}")
8  print(f"El promedio de las notas es de: {promedio}")

```

tuvimos problemas con la lógica de los ejercicios pero alejandro nos ayudó entender

jorge esteves – 10/07/2025 – 18:33

```

MIPrimeraChamba 0% used Run Invite Deploy

Files Search
  .git
  Alejandro
  Annie
  FernandoTarea
  joel-trabajos
  jorgeEsteves200
    ejercicios_prog2
      2funciones (1).py
      clasificaPelicula.py
      encontrarMayor.py
      invertirLista.py
      invertirListaSlicing.py
      nroOculto (1).py
      nroOculto2 (1).py
      refactorizar.py
      saludaPersona.py
      sumaAB (1).py
      sumaNotas.py
      tablamultiplicar.py
      tablaMultiplicar2.py

Shell
~/workspace/PunterosLetales/jorgeEsteves200/ejercicios_prog2: python invertirLista.py
~/workspace$ cd PunterosLetales/
~/workspace/PunterosLetales$ cd jo
joel-trabajos/ jorgeEsteves200/
~/workspace/PunterosLetales$ cd jorgeEsteves200/
~/.../PunterosLetales/jorgeEsteves200$ cd ejercicios_prog2/
~/.../jorgeEsteves200/ejercicios_prog2$ py invertirLista
/home/runner/workspace/.pythonlibs/bin/python3.11: can't open file '/home/runner/workspace/PunterosLetales/jorgeEsteves200/ejercicios_prog2/invertir
Lista': [Errno 2] No such file or directory
~/.../jorgeEsteves200/ejercicios_prog2$ python invertirLista
python: can't open file '/home/runner/workspace/PunterosLetales/jorgeEsteves200/ejercicios_prog2/invertirLista': [Errno 2] No such file or directory
~/.../jorgeEsteves200/ejercicios_prog2$ python invertirLista
python: can't open file '/home/runner/workspace/PunterosLetales/jorgeEsteves200/ejercicios_prog2/invertirLista': [Errno 2] No such file or directory
~/.../jorgeEsteves200/ejercicios_prog2$ python invertirListaSlicing.py

Probando invertir lista con slicing...
¡Pruebas con slicing jorge esteves ✓
~/.../jorgeEsteves200/ejercicios_prog2$ python invertirLista
invertirLista.py
~/.../jorgeEsteves200/ejercicios_prog2$ python invertirListaSlicing.py
~/.../jorgeEsteves200/ejercicios_prog2$ python invertirLista.py

Probando invertir_lista...
¡Pruebas para invertir_lista jorge esteves ✓
~/.../jorgeEsteves200/ejercicios_prog2$
Generate

```

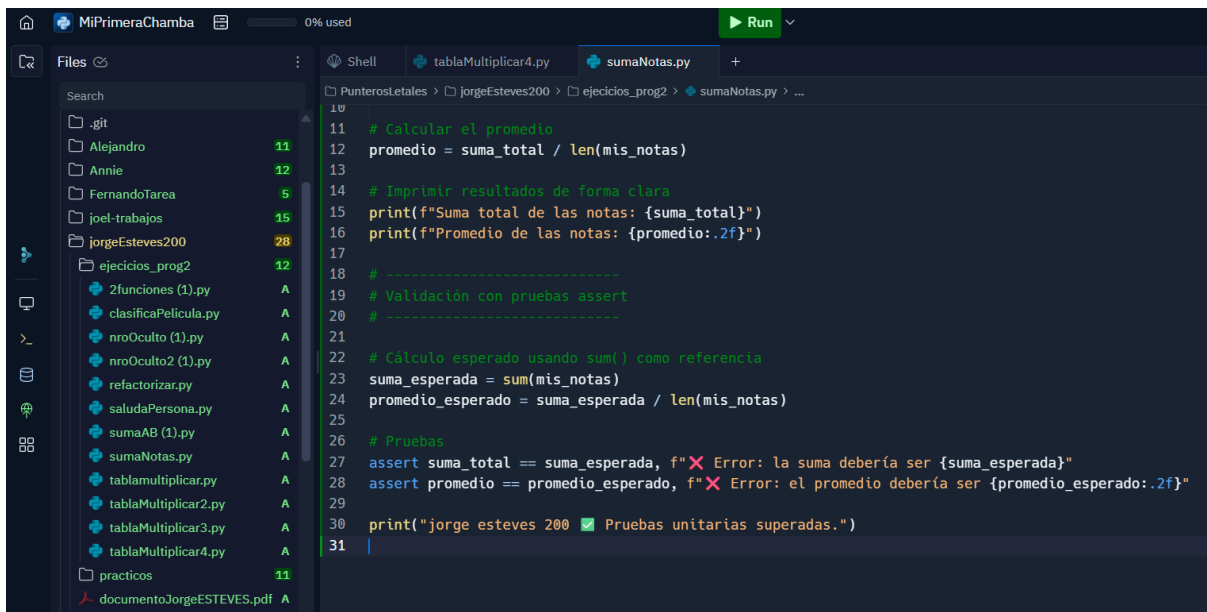
```

Shell sumaNotas.py +

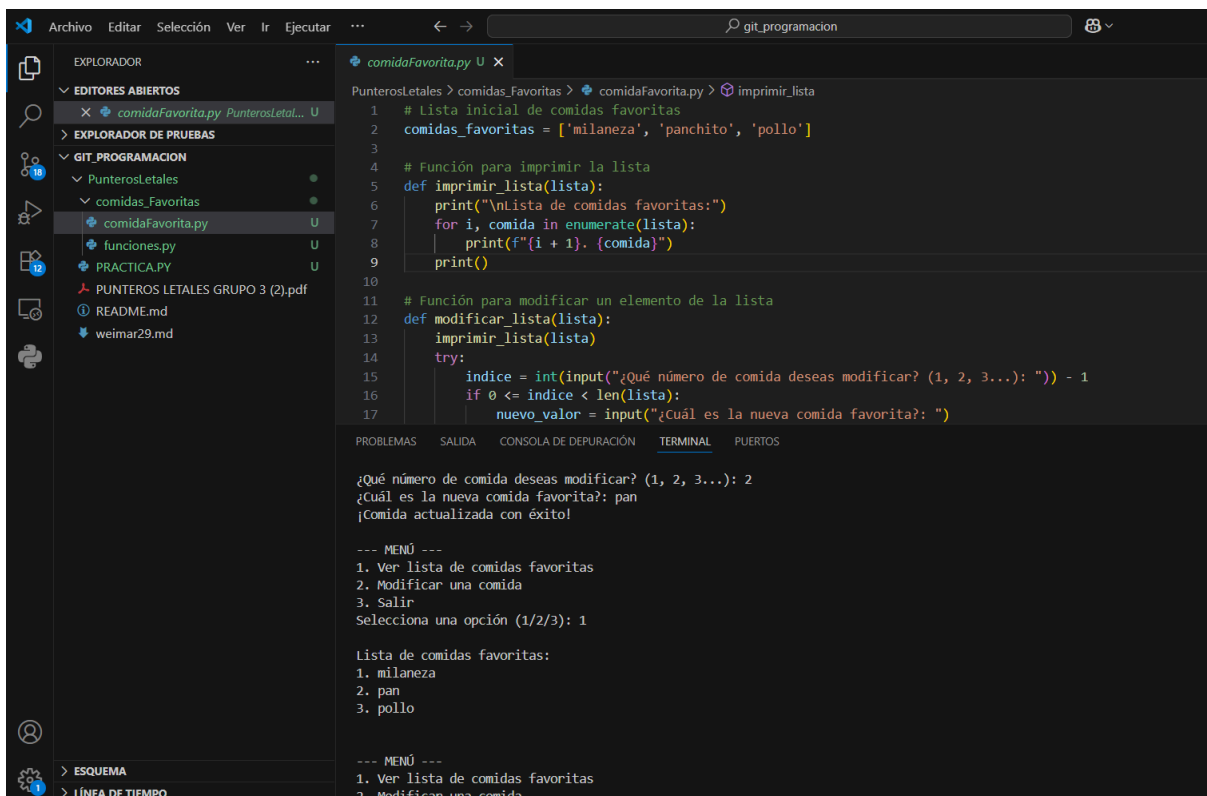
~/workspace/PunterosLetales/jorgeEsteves200/ejercicios_prog2: py sumaNotas.py

~/workspace$ cd PunterosLetales/
~/workspace/PunterosLetales$ cd jorgeEsteves200/
~/.../PunterosLetales/jorgeEsteves200$ cd ejercicios_prog2/
~/.../jorgeEsteves200/ejercicios_prog2$ py suma
sumaAB (1).py sumaNotas.py
~/.../jorgeEsteves200/ejercicios_prog2$ py suma
sumaAB (1).py sumaNotas.py
~/.../jorgeEsteves200/ejercicios_prog2$ py sumaNotas.py
py: command not installed, but was located via Nix.
package: python-launcher 1.0.0 An implementation of the `py` command for Unix-based platforms
Would you like to run py from Nix and add it to your project? [Yn]: y
python-launcher 1.0.0 An implementation of the `py` command for Unix-based platforms
Adding python-launcher to .replit
Suma total de las notas: 519.0
Promedio de las notas: 86.50
jorge esteves 200 ✓ Pruebas unitarias superadas.
~/.../jorgeEsteves200/ejercicios_prog2$

```



```
10
11 # Calcular el promedio
12 promedio = suma_total / len(mis_notas)
13
14 # Imprimir resultados de forma clara
15 print(f"Suma total de las notas: {suma_total}")
16 print(f"Promedio de las notas: {promedio:.2f}")
17
18 # -----
19 # Validación con pruebas assert
20 # -----
21
22 # Cálculo esperado usando sum() como referencia
23 suma_esperada = sum(mis_notas)
24 promedio_esperado = suma_esperada / len(mis_notas)
25
26 # Pruebas
27 assert suma_total == suma_esperada, f"❌ Error: la suma debería ser {suma_esperada}"
28 assert promedio == promedio_esperado, f"❌ Error: el promedio debería ser {promedio_esperado:.2f}"
29
30 print("jorge esteves 200 ✅ Pruebas unitarias superadas.")
31
```



```
1 # Lista inicial de comidas favoritas
2 comidas_favoritas = ['milaneza', 'panchito', 'pollo']
3
4 # Función para imprimir la lista
5 def imprimir_lista(lista):
6     print("\nLista de comidas favoritas:")
7     for i, comida in enumerate(lista):
8         print(f"{i + 1}. {comida}")
9     print()
10
11 # Función para modificar un elemento de la lista
12 def modificar_lista(lista):
13     imprimir_lista(lista)
14     try:
15         indice = int(input("¿Qué número de comida deseas modificar? (1, 2, 3...): ")) - 1
16         if 0 <= indice < len(lista):
17             nuevo_valor = input("¿Cuál es la nueva comida favorita?: ")
18             lista[indice] = nuevo_valor
19     except ValueError:
20         print("Error: El índice debe ser un número entero válido.")
21
22 # Función para buscar una comida en la lista
23 def buscar_comida(lista):
24     print("\nBuscando comida favorita...")
25     palabra = input("Ingresa la palabra que quieras buscar: ")
26     encontrado = False
27     for i, comida in enumerate(lista):
28         if comida.lower() == palabra.lower():
29             encontrado = True
30             print(f"La comida '{palabra}' se encuentra en la posición {i + 1}.")
31     if not encontrado:
32         print(f"No se encontró la comida '{palabra}' en la lista.")
33
34 # Función para salir del programa
35 def salir():
36     print("¡Comida actualizada con éxito!")
37     print("\n--- MENÚ ---")
38     print("1. Ver lista de comidas favoritas")
39     print("2. Modificar una comida")
40     print("3. Salir")
41     print("Selecciona una opción (1/2/3): ")
42
43 # Función principal
44 def main():
45     imprimir_lista(comidas_favoritas)
46     while True:
47         opcion = input("Ingresa el número de la opción que deseas realizar: ")
48         if opcion == "1":
49             imprimir_lista(comidas_favoritas)
50         elif opcion == "2":
51             modificar_lista(comidas_favoritas)
52         elif opcion == "3":
53             salir()
54         else:
55             print("Opción no válida. Por favor, ingresa un número del 1 al 3.")
56
57 # Ejecutar la función principal
58 if __name__ == "__main__":
59     main()
```

practica clase

jorge esteves – 10/07/2025 – 20:19



## Prácticas de clase 4

### TRABAJOS EN CLASE EVIDENCIAS

```
Shell  contarElemento (1).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > contarElemento (1).py > ...

1  # Definición de la función
2  def contar_elemento(lista, elemento_buscado):
3      contador = 0
4      for elemento in lista:
5          if elemento == elemento_buscado:
6              contador += 1
7      return contador
8
9  # -----
10 # 🖍 Casos de prueba con assert
11 # -----
12
13 print("\nProbando contar_elemento...")
14
15 assert contar_elemento([1, 2, 3, 2, 4, 2], 2) == 3
16 assert contar_elemento(["a", "b", "a", "c", "a"], "a") == 3
17 assert contar_elemento(["sol", "luna", "estrella"], "martes") == 0
18 assert contar_elemento([], 5) == 0
19 assert contar_elemento([True, False, True, True], True) == 3
20
21 print("¡Pruebas para contar_elemento pasaron! ✅")
22
```

```
Shell  appendVsInsert (1).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > appendVsInsert (1).py > ...

1  # Creamos una lista vacía
2  numeros = []
3
4  # append agrega al final
5  numeros.append(10)
6  numeros.append(20)
7  print("Usando append:", numeros)  # [10, 20]
8
9  # insert agrega en una posición específica
10 numeros.insert(1, 15)  # insertar en índice 1 el valor 15
11 print("Usando insert:", numeros)  # [10, 15, 20]
12
13
14 # Queremos duplicar cada número
15 numeros = [1, 2, 3, 4]
16 duplicados = list(map(lambda x: x * 5.5, numeros))
17 print("Usando map:", duplicados)  # [2, 4, 6, 8]
18
```

```
Shell  contarElementoConCount (1)... +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > contarElementoConCount (1).py > f con

1 def contar_elemento(lista, elemento_buscado):
2     return lista.count(elemento_buscado)
3 print("\nProbando contar_elemento con .count()...")
4
5 assert contar_elemento([1, 2, 3, 2, 4, 2], 2) == 3
6 assert contar_elemento([1, 2, 3, 2, 4, 2], 1) == 1
7 assert contar_elemento(["a", "b", "a", "c", "a"], "a") == 3
8 assert contar_elemento(["sol", "luna", "estrella"], "marte") == 0
9 assert contar_elemento([], 5) == 0
10 assert contar_elemento([True, False, True, True], True) == 3
11
12 print("¡Pruebas con .count() pasaron! ✅")
13
```

```
Shell  encontrarMayor (1).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > encontrarMayor (1).py > ...

1 # Definición de la función
2 def encontrar_mayor(lista_numeros):
3     # Caso especial: lista vacía
4     if not lista_numeros:
5         return None
6
7     # Paso 1: El primer "campeón"
8     mayor_temporal = lista_numeros[0]
9
10    # Paso 2-4: Recorrer la lista para buscar al más grande
11    for elemento in lista_numeros:
12        if elemento > mayor_temporal:
13            mayor_temporal = elemento
14
15    # Paso 5: Devolver el campeón
16    return mayor_temporal
17
18 # -----
19 # 🖋️ Casos de prueba con assert
20 # -----
21
22 print("Probando encontrar_mayor...")
23
24 assert encontrar_mayor([1, 5, 3, 9, 2]) == 9
25 assert encontrar_mayor([-10, -5, -3, -20]) == -3
26 assert encontrar_mayor([7, 7, 7, 7]) == 7
27 assert encontrar_mayor([]) == None # lista vacía
28 assert encontrar_mayor([42]) == 42 # un solo elemento
29
30 print("¡Pruebas para encontrar_mayor pasaron! ✅")
31
```

```
Shell factorial (1).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > factorial (1).py > ...

1 # Definición de la función recursiva
2 def factorial(n):
3     if n < 0:
4         raise ValueError("❌ El factorial no está definido para números negativos.")
5     elif n == 0 or n == 1:
6         return 1
7     else:
8         return n * factorial(n - 1)
9
10 assert factorial(0) == 1
11 assert factorial(1) == 1
12 assert factorial(5) == 120
13 assert factorial(6) == 720
14 print("¡Pruebas de factoriales pasaron! ✅")
15
16 # Solicita un número al usuario
17 try:
18     numero = int(input("Ingrese un número entero para calcular su factorial: "))
19     resultado = factorial(numero)
20     print(f"El factorial de {numero} es: {resultado}")
21 except ValueError as e:
22     print(e)
23
```

```
Shell factorialFor (1).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > factorialFor (1).py > ...

1 # Definición de la función usando un for
2 def factorial(n):
3     if n < 0:
4         raise ValueError("❌ El factorial no está definido para números negativos.")
5
6     resultado = 1
7     for i in range(2, n + 1):
8         resultado *= i
9     return resultado
10
11
12 # Solicita un número al usuario
13 try:
14     numero = int(input("Ingrese un número entero para calcular su factorial: "))
15     resultado = factorial(numero)
16     print(f"El factorial de {numero} es: {resultado}")
17 except ValueError as e:
18     print(e)
19
```

```
Shell factorialFuncToolsReduce (1)... +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > factorialFuncToolsReduce (1).py > ...

1 from functools import reduce
2
3 def factorial(n):
4     if n < 0:
5         raise ValueError("❌ El factorial no está definido para números negativos.")
6     return 1 if n == 0 else reduce(lambda x, y: x * y, range(1, n + 1))
7
8 assert factorial(0) == 1
9 assert factorial(1) == 1
10 assert factorial(5) == 120
11 assert factorial(6) == 720
12 print("¡Pruebas de factoriales pasaron! ✅")
13
14 # Solicita un número al usuario
15 try:
16     numero = int(input("Ingrese un número entero para calcular su factorial: "))
17     resultado = factorial(numero)
18     print(f"El factorial de {numero} es: {resultado}")
19 except ValueError as e:
20     print(e)
```

```
Shell factorialMath (1).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > factorialMath (1).py > ...

1 import math
2
3 def factorial(n):
4     if n < 0:
5         raise ValueError("❌ El factorial no está definido para números negativos.")
6     return 1 if n == 0 else math.prod(range(1, n + 1))
7
8 def mostrar_factorial(n):
9     if n < 0:
10         print("❌ El factorial no está definido para números negativos.")
11         return
12
13     pasos = ' x '.join(str(i) for i in range(1, n + 1)) if n > 0 else "1"
14     resultado = factorial(n)
15     print(f"{pasos} = {resultado}")
16
17 # -----
18 # Ejecución interactiva
19 # -----
20 try:
21     numero = int(input("Ingrese un número entero para calcular su factorial: "))
22     mostrar_factorial(numero)
23 except ValueError:
24     print("❌ Entrada inválida. Por favor ingresa un número entero.")
25
```

```
Shell forTradicionalVsPythonico (1... +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > forTradicionalVsPythonico (1).
1 #CALCULO DEL CUADRADO DE LOS ELEMENTOS DE UNA LISTA DEL 0 AL 4
2 cuadrados = []
3 # TRADICIONAL
4 for x in range(5):
5     cuadrados.append(x * x)
6 print(cuadrados) # [0, 1, 4, 9, 16]
7 #PYTHONICO
8 cuadrados = [x * x for x in range(5)]
9 print(cuadrados) # [0, 1, 4, 9, 16]
10
```

```
Shell invertirLista (2).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > invertirLista (2).py > ...
1 # Definición de la función
2 def invertir_lista(lista_original):
3     lista_invertida = []
4
5     # Recorremos la lista desde el final hasta el inicio
6     for i in range(len(lista_original) - 1, -1, -1):
7         lista_invertida.append(lista_original[i])
8
9     return lista_invertida
10
11 # -----
12 # 📌 Casos de prueba con assert
13 # -----
14
15 print("\nProbando invertir_lista...")
16
17 lista_prueba = [1, 2, 3, 4, 5]
18 lista_resultante = invertir_lista(lista_prueba)
19
20 assert lista_resultante == [5, 4, 3, 2, 1], "❌ Error en inversión"
21 assert lista_prueba == [1, 2, 3, 4, 5], "❌ ¡La lista original fue modificada"
22 assert invertir_lista(["a", "b", "c"]) == ["c", "b", "a"]
23 assert invertir_lista([]) == []
24
25 print("¡Pruebas para invertir_lista pasaron! ✅")
26
```

```
Shell  invertirListaSlicing (2).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > invertirListaSlicing (2).py > f invertir_lista >
1 def invertir_lista(lista_original):
2     # lista[inicio:fin:paso]
3     return lista_original[::-1]
4     # Esto usa slicing con paso negativo
5     # inicio: índice desde donde empezar (incluye este elemento).
6     # fin: índice hasta donde cortar (no incluye este elemento).
7     # paso: cuántos pasos avanzar (puede ser negativo).
8
9
10    # Pruebas
11    print("\nProbando invertir_lista con slicing...")
12
13    lista_prueba = [1, 2, 3, 4, 5]
14    lista_resultante = invertir_lista(lista_prueba)
15
16    assert lista_resultante == [5, 4, 3, 2, 1], "❌ Error en inversión con slicing"
17    assert lista_prueba == [1, 2, 3, 4, 5], "❌ ¡La lista original fue modificada!"
18    assert invertir_lista(["a", "b", "c"]) == ["c", "b", "a"]
19    assert invertir_lista([]) == []
20
21    print("¡Pruebas con slicing pasaron! ✅")
22
```

```
Shell  invertirTexto.py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > invertirTexto.py > x texto
1 texto = "hola"
2 invertido = ''.join(texto[i] for i in range(len(texto) - 1, -1, -1))
3 print(invertido) # Resultado: "aloh"
4
```

```
Shell sumaElementos.py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > sumaElementos.py > ...

1 # Definición de la función para sumar elementos
2 def sumar_elementos(lista_numeros):
3     acumulador_suma = 0
4     for elemento_actual in lista_numeros:
5         acumulador_suma += elemento_actual
6     return acumulador_suma
7
8 # Casos de prueba con assert
9 print("Probando sumar_elementos...")
10
11 assert sumar_elementos([1, 2, 3, 4, 5]) == 15
12 assert sumar_elementos([10, -2, 5]) == 13
13 assert sumar_elementos([]) == 0 # ¡Importante probar con una lista vacía!
14 assert sumar_elementos([100]) == 100
15
16 print("OK! ¡Pruebas para sumar_elementos pasaron!")
17
```

```
Shell sumaNotas (1).py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > sumaNotas (1).py > ...

1 # Crear una lista con notas numéricas
2 mis_notas = [85.5, 90, 78, 88.5, 95, 82]
3
4 # Inicializar variable para la suma
5 suma_total = 0
6
7 # Usar bucle for para calcular la suma total sin usar sum()
8 for nota in mis_notas:
9     suma_total += nota
10
11 # Calcular el promedio
12 promedio = suma_total / len(mis_notas)
13
14 # Imprimir resultados de forma clara
15 print(f"Suma total de las notas: {suma_total}")
16 print(f"Promedio de las notas: {promedio:.2f}")
17
18 # -----
19 # Validación con pruebas assert
20 # -----
21
22 # Cálculo esperado usando sum() como referencia
23 suma_esperada = sum(mis_notas)
24 promedio_esperado = suma_esperada / len(mis_notas)
25
26 # Pruebas
27 assert suma_total == suma_esperada, f"❌ Error: la suma debería ser {suma_esperada}"
28 assert promedio == promedio_esperado, f"❌ Error: el promedio debería ser {promedio_esperado:.2f}"
29
30 print("OK ¡Todo correcto! Las validaciones pasaron exitosamente.")
31
```

```
Shell sumaPythonica.py +
PunterosLetales > jorgeEsteves200 > ejercicios_prog2 > carpeta practicas > sumaPythonica.py > ...

1  # Definición de la función para sumar elementos (versión pythonica)
2  def sumar_elementos(lista_numeros):
3      # return sum(lista_numeros)
4      return sum([x for x in lista_numeros])
5
6  # Casos de prueba con assert
7  print("Probando sumar_elementos...")
8
9  assert sumar_elementos([1, 2, 3, 4, 5]) == 15
10 assert sumar_elementos([10, -2, 5]) == 13
11 assert sumar_elementos([]) == 0 # ¡Importante probar con una lista vacía!
12 assert sumar_elementos([100]) == 100
13
14 print("¡Pruebas para sumar_elementos pasaron! ✅")
15
```

15 ARCHIVOS DIFERENTES QUE SE VIERON EN CLASE PARA EDITAR Y PROBAR DIFERENTES VALORES

JORGE ESTEVES – 10/07/2025 – 21:47