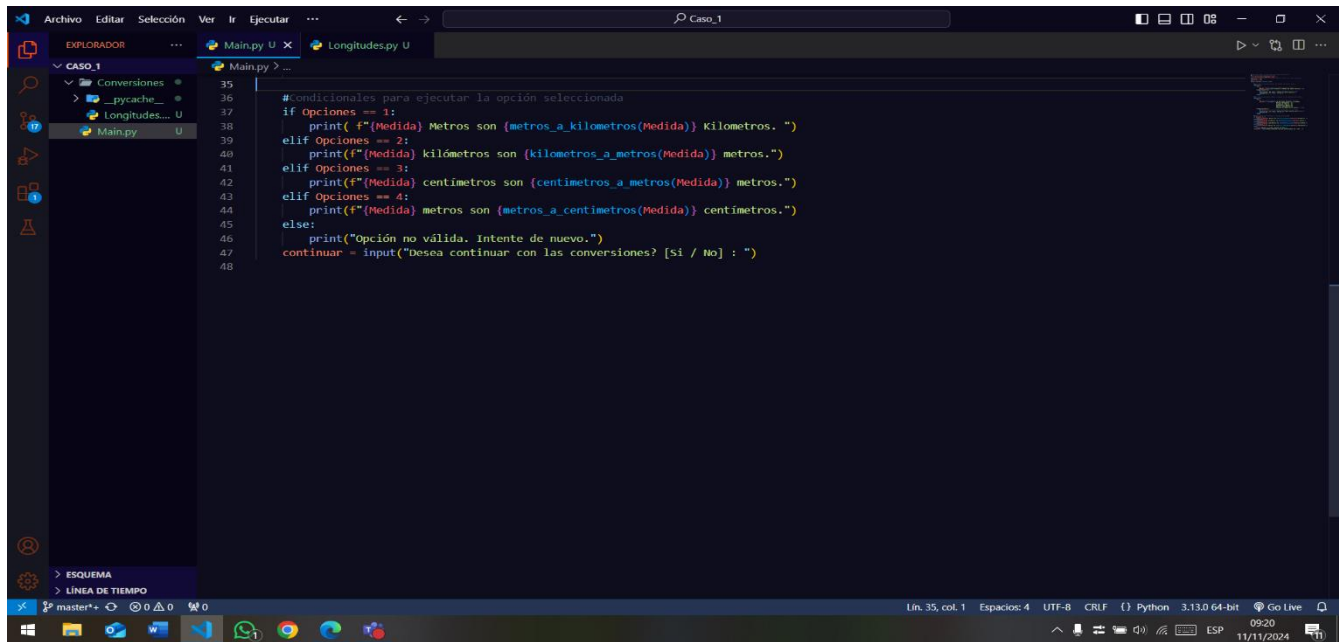


Caso 1:

```
1 #creamos funciones para las conversiones
2 def metros_a_kilometros(metros):
3     kilometros = metros * 0.001
4     return kilometros
5 def kilometros_a_metros(kilometros):
6     metros = kilometros * 1000
7     return metros
8 def centimetros_a_metros(centimetros):
9     metros = centimetros * 0.01
10    return metros
11 def metros_a_centimetros(metros):
12     centimetros = metros * 100
13     return centimetros
14 #Retornamos los resultados
```

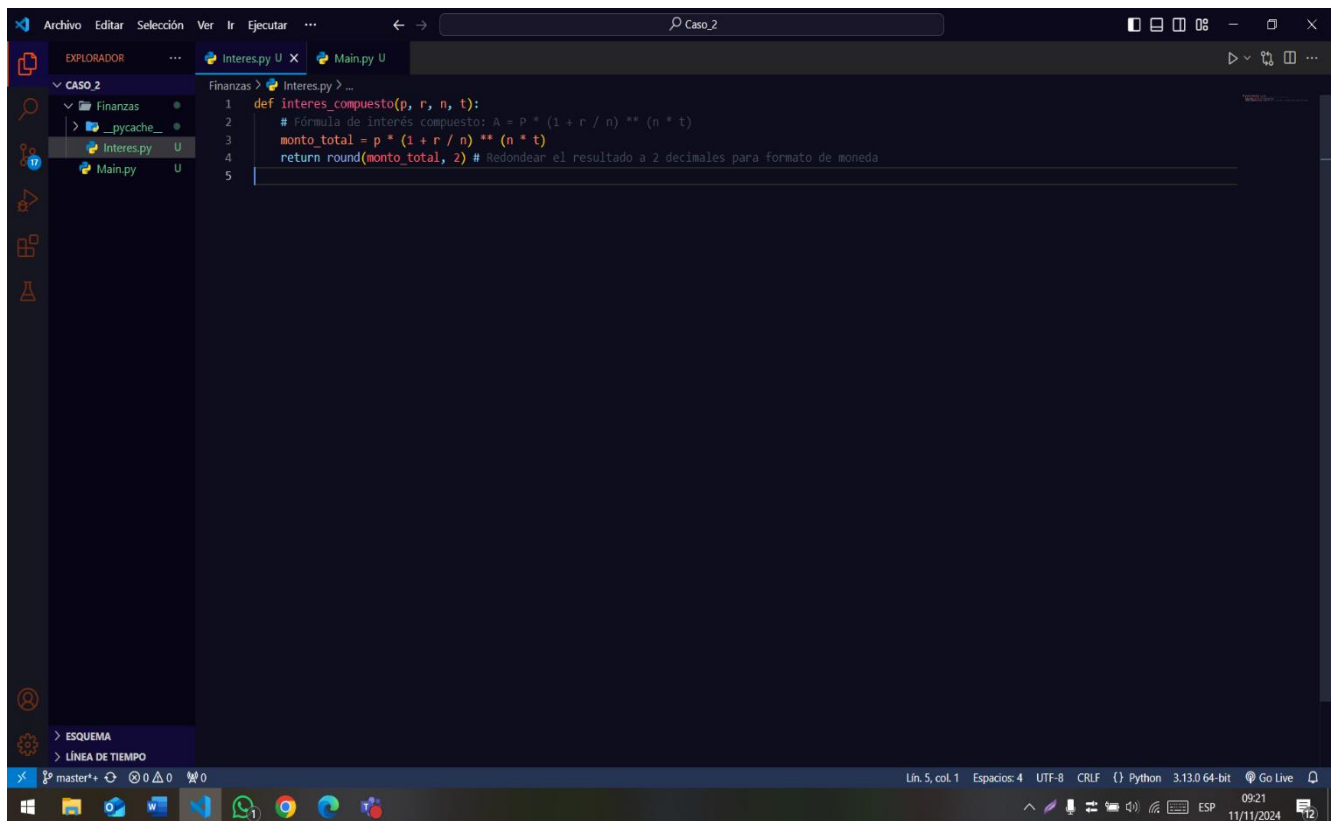
```
1 #Importamos todos los elementos del modulo para poderlo
2 #usar en nuestro programa principal
3 from Conversiones.Longitudes import *
4 #Variable que controla si el usuario desea realizar una nueva conversion
5 continuar = "si"
6 #Bucle principal
7 while continuar.lower() == "si":
8
9     #Bucle que nos permite obtener una medida valida del usuario
10    while True:
11        try:
12            #se solicita al usuario la medida que desea convertir
13            Medida = float(input("Ingrese la medida que desea convertir: "))
14        except ValueError:
15            #Mensaje de error si la medida ingresada no es un número
16            print("Error: Por favor, ingrese un valor numérico.")
17            continue #vuelve a pedir la medida
18        break
19
20    #Bucle que nos permite obtener la opcion de la conversion del usuario
21    while True:
22        try:
23            #Se solicita al usuario la conversion que desea realizar
24            Opciones = int(input("""En que desea convertir la medida?
25                                Metros a Kilometros [1]
26                                Kilometros a Metros [2]
27                                Centimetros a Metros [3]
28                                Metros a Centimetros [4]
29                                ¡Ingrese el numero de su opcion! : """))
30        except ValueError:
31            #Mensaje de error si la opción ingresada no es un número
32            print("Error: Por favor, ingrese un número entero entre 1 y 4.")
33            continue #vuelve a pedir la opcion
34        break
35
36    #Condicionales para ejecutar la opción seleccionada
37    if Opciones == 1:
38        print(f"({Medida}) metros son {metros_a_kilometros(Medida)} kilometros")
```



The screenshot shows the Visual Studio Code interface with a file explorer on the left displaying a project named 'CASO_1' containing folders like 'Conversiones', '._pycache_', and 'Longitudes...'. The main editor window shows a Python script for unit conversions. The script uses a series of if-elif-else statements to handle different input options (1 for Kilometros, 2 for metros, 3 for centímetros, 4 for metros). It includes functions like `metros_a_kilometros`, `kilometros_a_metros`, and `centimetros_a_metros`. The script ends with a loop that asks the user if they want to continue with conversions.

```
35
36
37
38 #condicionales para ejecutar la opción seleccionada
39 if Opciones == 1:
40     print(f"{Medida} Metros son {metros_a_kilometros(Medida)} Kilometros. ")
41 elif Opciones == 2:
42     print(f"{Medida} kilómetros son {kilometros_a_metros(Medida)} metros.")
43 elif Opciones == 3:
44     print(f"{Medida} centímetros son {centimetros_a_metros(Medida)} metros.")
45 elif Opciones == 4:
46     print(f"{Medida} metros son {metros_a_centimetros(Medida)} centímetros.")
47 else:
48     print("Opción no válida. Intente de nuevo.")
49 continuar = input("Desea continuar con las conversiones? [Si / No] : ")
```

CASO 2



The screenshot shows the Visual Studio Code interface with a file explorer on the left displaying a project named 'CASO_2' containing folders like 'Finanzas', '._pycache_', and 'Interes.py'. The main editor window shows a Python script for calculating compound interest. The script defines a function `interes_compuesto` that takes parameters `p`, `r`, `n`, and `t`. It uses the formula $A = P * (1 + r / n) ** (n * t)$ to calculate the total amount. The result is rounded to 2 decimal places using `round(monto_total, 2)`.

```
1 def interes_compuesto(p, r, n, t):
2     # Fórmula de Interés compuesto: A = P * (1 + r / n) ** (n * t)
3     monto_total = p * (1 + r / n) ** (n * t)
4     return round(monto_total, 2) # Redondear el resultado a 2 decimales para formato de moneda
5
```

```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  ...  Caso_2
EXPLORADOR  Interes.py U  Main.py U X
CASO_2
  Finanzas
  _pycache_
  Interes.py U
  Main.py U
Main.py > ...
1  # Importamos todos los elementos del módulo para usarlos en nuestro programa principal
2  from Finanzas.Interes import *
3  # Flujo principal del programa
4  print("Calculadora de Interés Compuesto")
5
6  # Solicitar al usuario los datos necesarios para el cálculo, validando entradas
7  while True:
8      try:
9          # Solicitar el monto principal
10         p = float(input("Ingrese el monto principal (P): "))
11         if p <= 0:
12             print("Error: El monto principal debe ser mayor que cero.")
13             continue
14
15         # Solicitar la tasa de interés anual en decimal
16         r = float(input("Ingrese la tasa de interés anual (r): "))
17         if not (0 < r < 1):
18             print("Error: La tasa de interés debe estar entre 0 y 1.")
19             continue
20
21         # Solicitar el número de veces que se compone el interés por año
22         n = int(input("Ingrese el número de veces que se compone el interés por año (n): "))
23         if n <= 0:
24             print("Error: El número de veces debe ser un entero positivo.")
25             continue
26
27         # Solicitar el número de años
28         t = int(input("Ingrese el número de años (t): "))
29         if t <= 0:
30             print("Error: El número de años debe ser un entero positivo.")
31             continue
32
33         # Si todas las entradas son válidas, termina el bucle
34         break
35
36 except ValueError:
37     print("Error: Por favor, ingrese un valor numérico válido.")
38
```

```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  ...  Caso_2
EXPLORADOR  Interes.py U  Main.py U X
CASO_2
  Finanzas
  _pycache_
  Interes.py U
  Main.py U
Main.py > ...
38
39 # Calcular el interés compuesto
40 resultado = interes_compuesto(p, r, n, t)
41 print(f"El monto total después de {t} años es: {resultado}")
```