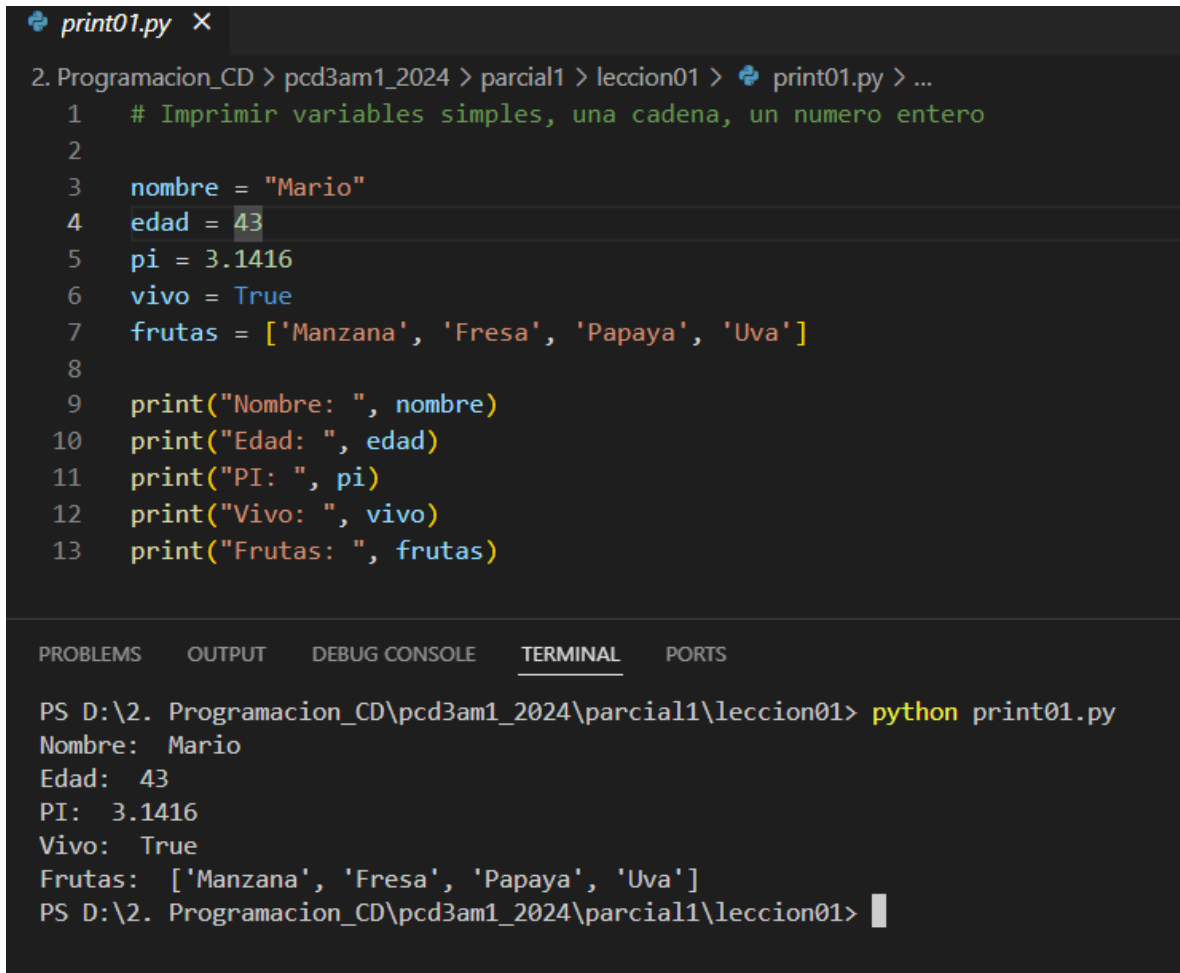


LECCIÓN 01

print01.py



The image shows a code editor window with a file named `print01.py`. The code defines several variables and a list, then prints their values. Below the code, the terminal output shows the results of running the script.

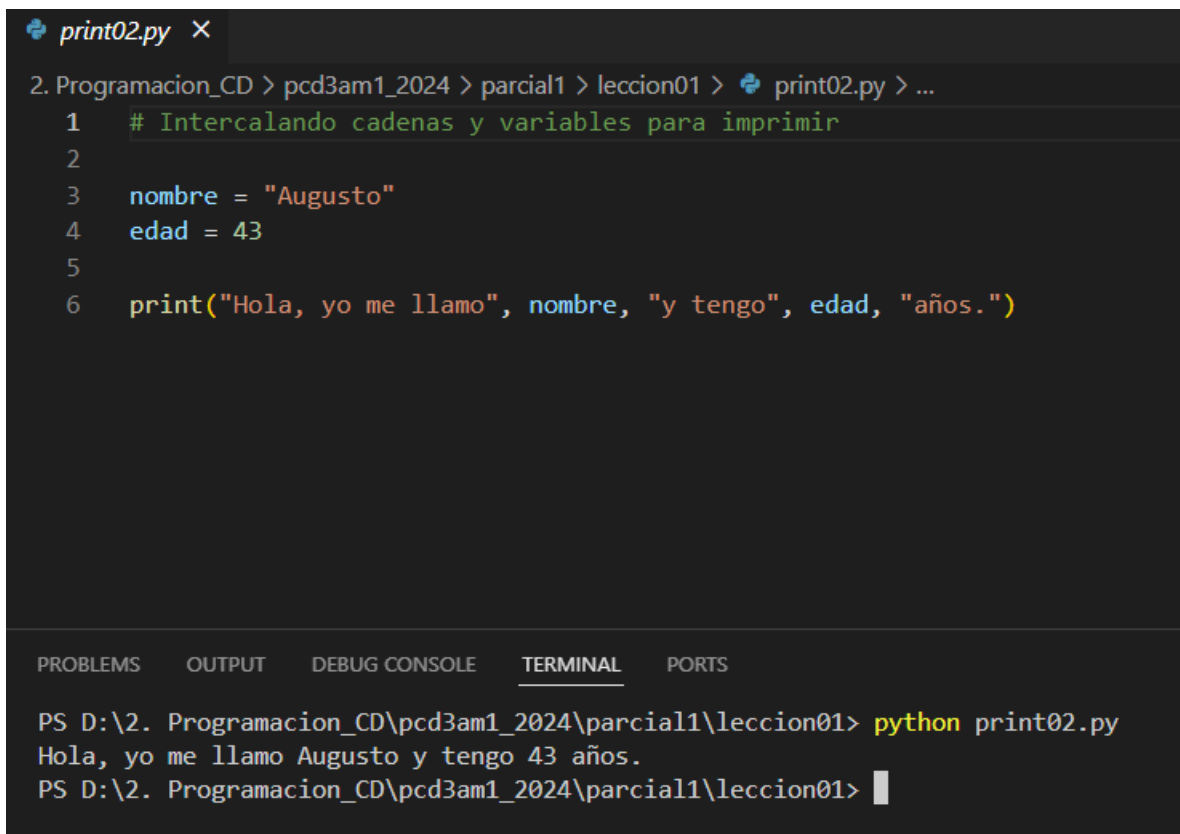
```
2. Programacion_CD > pcd3am1_2024 > parcial1 > leccion01 > print01.py > ...
1  # Imprimir variables simples, una cadena, un numero entero
2
3  nombre = "Mario"
4  edad = 43
5  pi = 3.1416
6  vivo = True
7  frutas = ['Manzana', 'Fresa', 'Papaya', 'Uva']
8
9  print("Nombre: ", nombre)
10 print("Edad: ", edad)
11 print("PI: ", pi)
12 print("Vivo: ", vivo)
13 print("Frutas: ", frutas)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print01.py
Nombre: Mario
Edad: 43
PI: 3.1416
Vivo: True
Frutas: ['Manzana', 'Fresa', 'Papaya', 'Uva']
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

Este programa declara 4 variables y una lista, después los imprime

print02.py



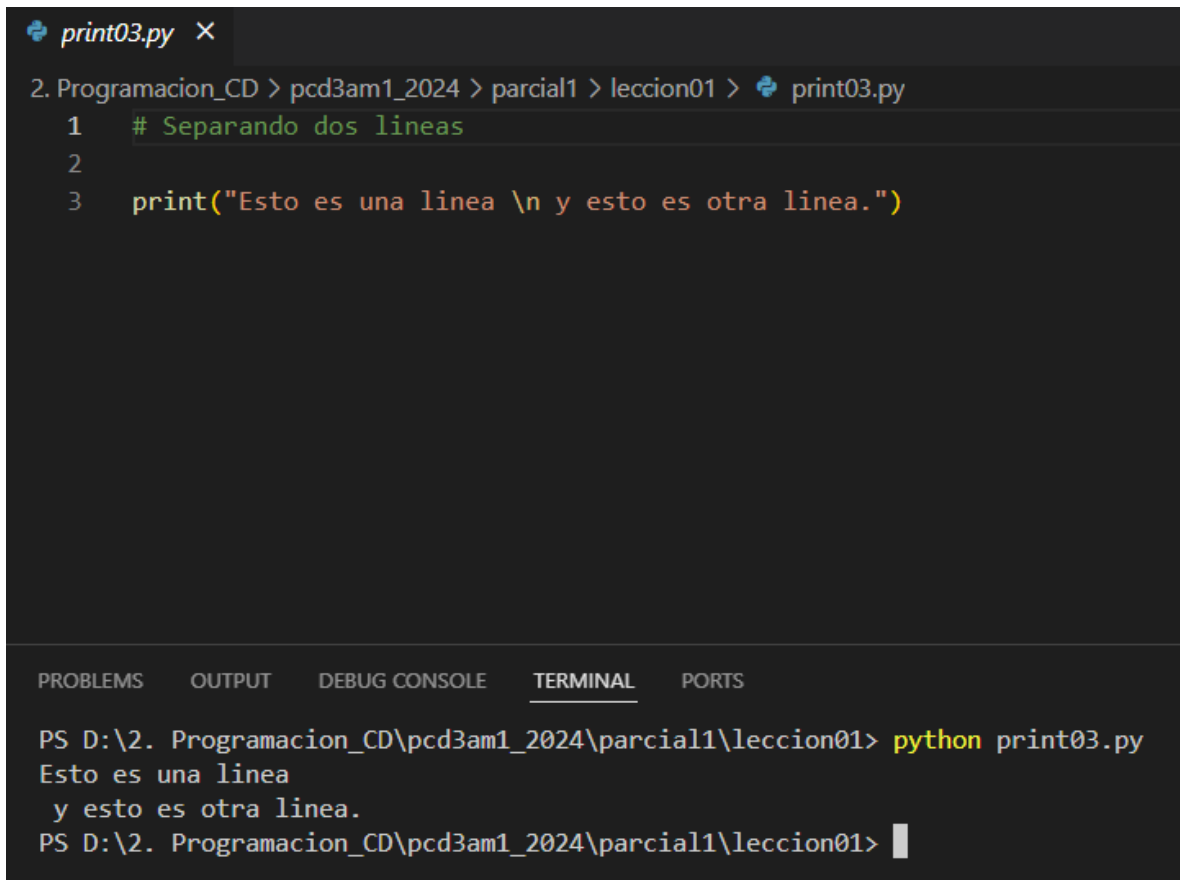
The image shows a code editor window with a dark theme. The top part displays a Python script named `print02.py` with the following content:

```
1  # Intercalando cadenas y variables para imprimir
2
3  nombre = "Augusto"
4  edad = 43
5
6  print("Hola, yo me llamo", nombre, "y tengo", edad, "años.")
```

Below the code editor, there is a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The terminal shows the command `python print02.py` being executed, resulting in the output: `Hola, yo me llamo Augusto y tengo 43 años.`

En este código, se declaran dos variables y después las imprime intercalando el texto que va a salir en pantalla y las variables

print03.py



The image shows a code editor window with a file named `print03.py`. The code contains three lines: a comment, a blank line, and a `print` statement using a newline character. Below the editor is a terminal window showing the command to run the script and the resulting output, which is split across two lines.

```
print03.py X
2. Programacion_CD > pcd3am1_2024 > parcial1 > leccion01 > print03.py
1  # Separando dos lineas
2
3  print("Esto es una linea \n y esto es otra linea.")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print03.py
Esto es una linea
y esto es otra linea.
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

En este código se aprecia la forma en la que se puede hacer un salto de línea cuando se imprime texto en pantalla, se usa `\n`.

print04.py

```
print04.py X
2. Programacion_CD > pcd3am1_2024 > parcial1 > leccion01 > print04.py
1  # Aqui vamos a imprimir una linea
2  print ("El mejor ron del mundo es Zacapa XO")
3
4  # Aqui vamos a substituir el caracter de fin de linea por ::
5  print ("Sin embargo este ron también es muy bueno", end= "::")
6  print("Diplomatico")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print04.py
El mejor ron del mundo es Zacapa XO
Sin embargo este ron también es muy bueno::Diplomatico
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

En este código hace que inmediatamente

print05.py



The image shows a code editor window with a file named 'print05.py' open. The code in the editor is as follows:

```
2. Programacion_CD > pcd3am1_2024 > parcial1 > leccion01 > print05.py
1  #Concatenando cadenas para imprimir
2
3  print('Amor' + ' & ' + 'Paz')
4  |
```

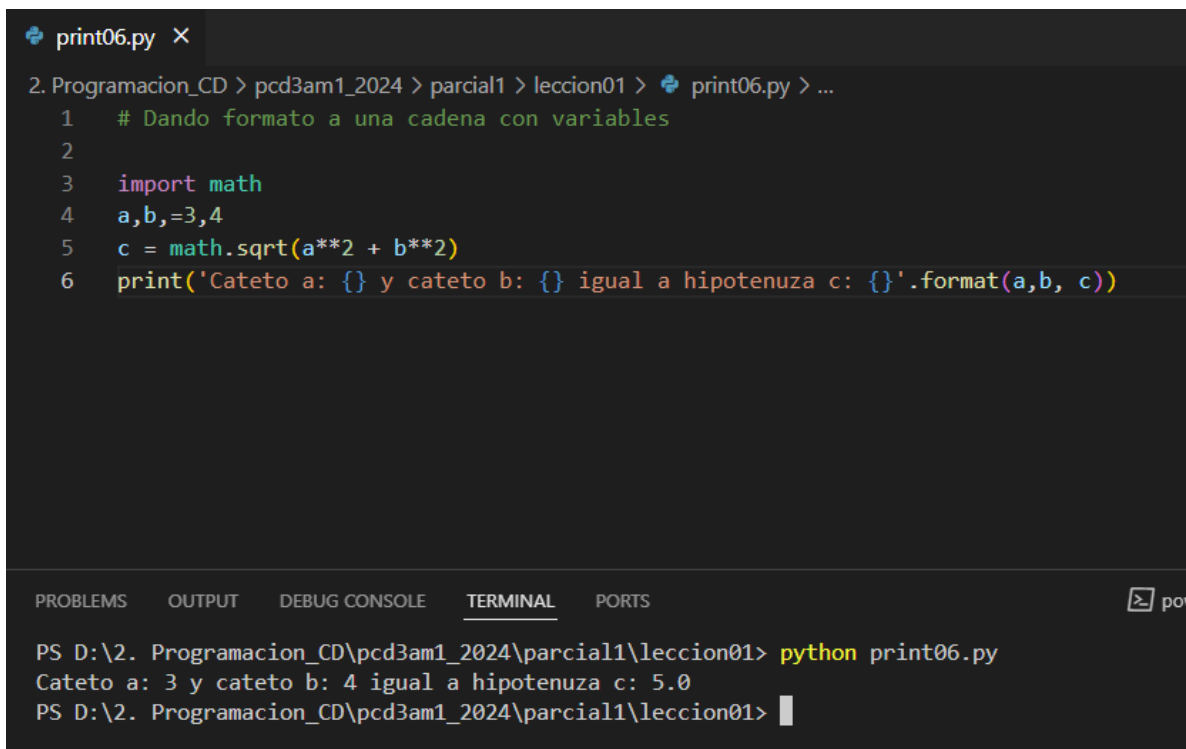
Below the code editor is a terminal window with the following output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print05.py
Amor & Paz
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> |
```

En esta línea de código se aplica la concatenación que sirve para cuando se tiene una línea muy larga y la queremos continuar en otra línea

printf06.py



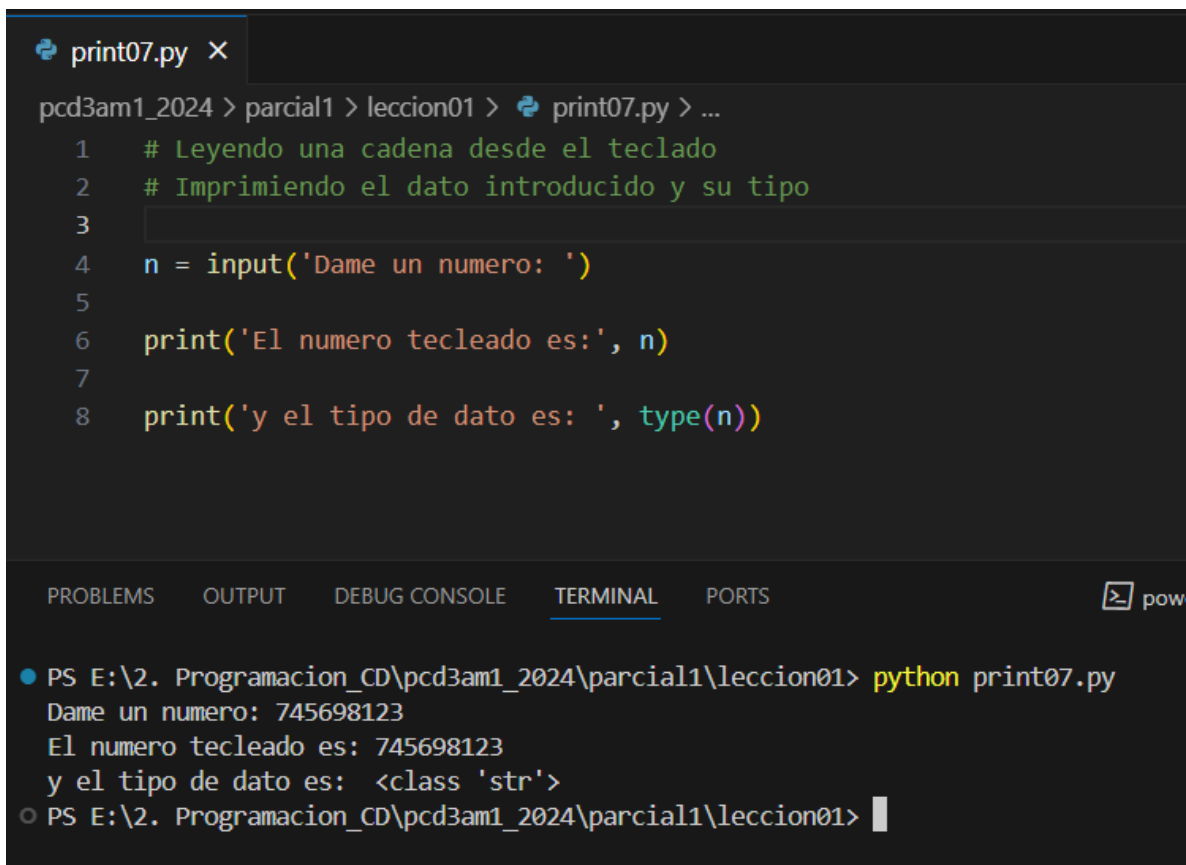
```
print06.py X
2. Programacion_CD > pcd3am1_2024 > parcial1 > leccion01 > print06.py > ...
1  # Dando formato a una cadena con variables
2
3  import math
4  a,b,=3,4
5  c = math.sqrt(a**2 + b**2)
6  print('Cateto a: {} y cateto b: {} igual a hipotenuza c: {}'.format(a,b, c))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print06.py
Cateto a: 3 y cateto b: 4 igual a hipotenuza c: 5.0
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

En este codigo se hace uso de la librería math que después de usa para implementar la herramienta de potencia.

La palabra .format ayuda a darle mejor formato a la hora de imprimir texto y sus variables correspondientes

printf07.py



The image shows a code editor window with a tab titled 'printf07.py'. The editor contains a Python script with the following code:

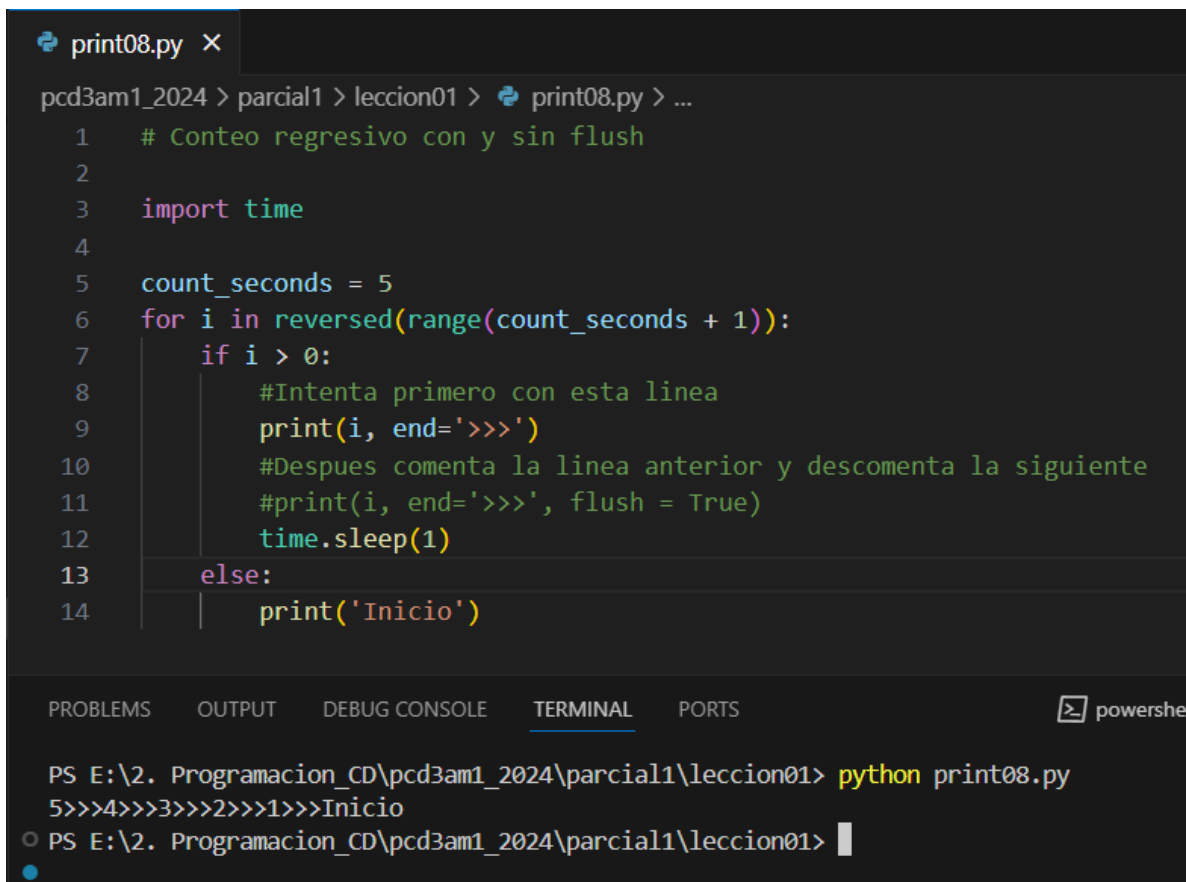
```
1 # Leyendo una cadena desde el teclado
2 # Imprimiendo el dato introducido y su tipo
3
4 n = input('Dame un numero: ')
5
6 print('El numero tecleado es:', n)
7
8 print('y el tipo de dato es: ', type(n))
```

Below the code editor, there is a terminal window with the following output:

```
PS E:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python printf07.py
Dame un numero: 745698123
El numero tecleado es: 745698123
y el tipo de dato es: <class 'str'>
PS E:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01>
```

En este programa se le solicita al usuario que ingrese un numero y posteriormente se imprime el numero ingresado por el usuario y después el tipo de dato que se ingresó, esto es gracias a la palabra `type` que sirve para poder identificar el tipo de dato de la variable que se le indica.

python08.py



The image shows a code editor window with a file named 'print08.py'. The code is a Python script that counts down from 5 to 1 and then prints 'Inicio'. The script uses the 'time' module for a 1-second delay between each number. The terminal output shows the execution of the script, resulting in the sequence '5>>>4>>>3>>>2>>>1>>>Inicio'.

```
print08.py X
pcd3am1_2024 > parcial1 > leccion01 > print08.py > ...
1  # Conteo regresivo con y sin flush
2
3  import time
4
5  count_seconds = 5
6  for i in reversed(range(count_seconds + 1)):
7      if i > 0:
8          #Intenta primero con esta linea
9          print(i, end='>>>')
10         #Despues comenta la linea anterior y descomenta la siguiente
11         #print(i, end='>>>', flush = True)
12         time.sleep(1)
13     else:
14         print('Inicio')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS E:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print08.py
5>>>4>>>3>>>2>>>1>>>Inicio
PS E:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01>
```

Aquí se implementa la librería `import time` que la aplicamos al momento de declarar `count_seconds = 5`, lo que hace que se espere unos segundos y después imprima toda la lista completa y finalmente la palabra inicio

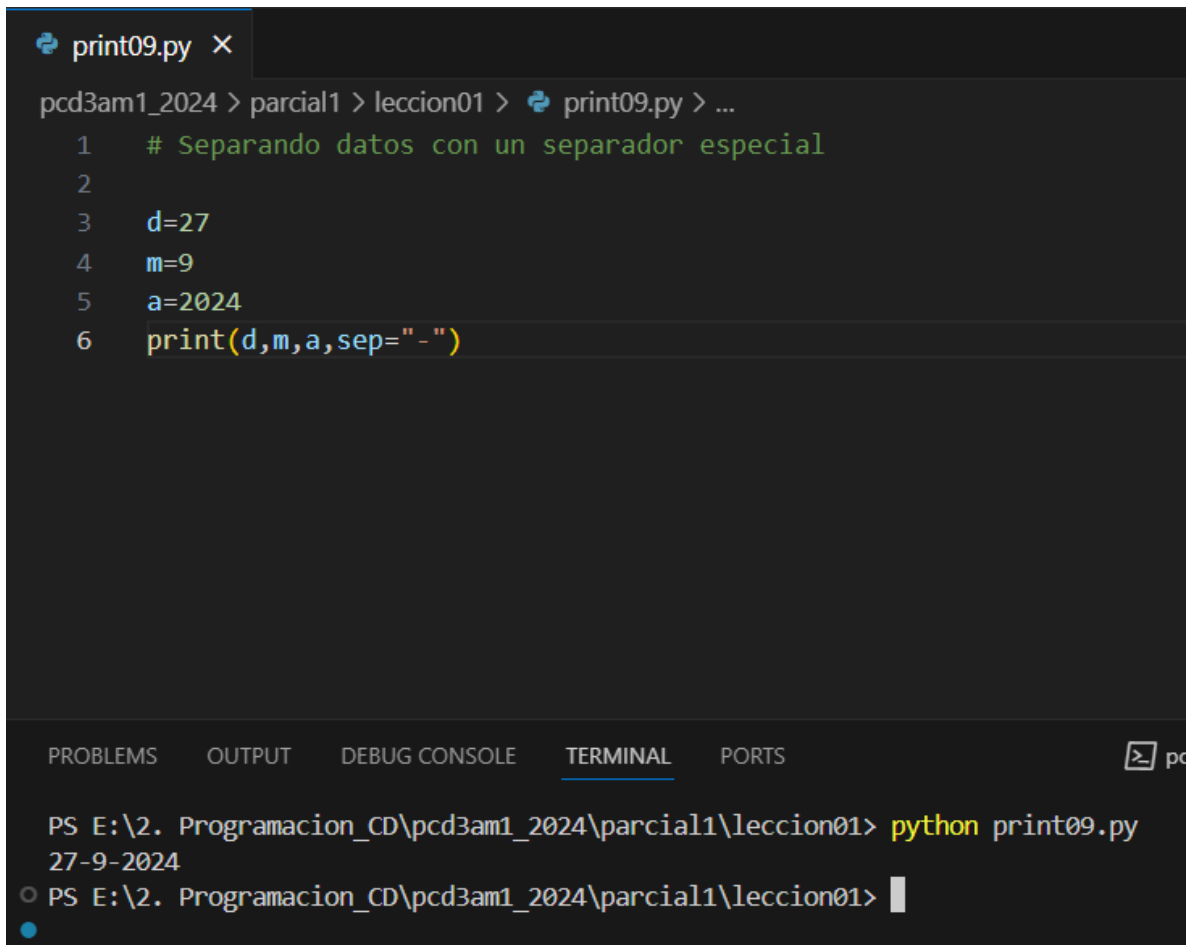

```
print08.py ×
pcd3am1_2024 > parcial1 > leccion01 > print08.py > ...
1  # Conteo regresivo con y sin flush
2
3  import time
4
5  count_seconds = 5
6  for i in reversed(range(count_seconds + 1)):
7      if i > 0:
8          #Intenta primero con esta linea
9          #print(i, end='>>>')
10         #Despues comenta la linea anterior y descomenta la siguiente
11         print(i, end='>>>', flush = True)
12         time.sleep(3)
13     else:
14         print('Inicio')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print08.py
5>>>4>>>3>>>2>>>1>>>Inicio
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> █
```

Aquí va imprimiendo la lista del 5 al 1 pero elemento por elemento, pero con una separación de tres segundos y así hasta que llegue a imprimir Inicio

print09.py



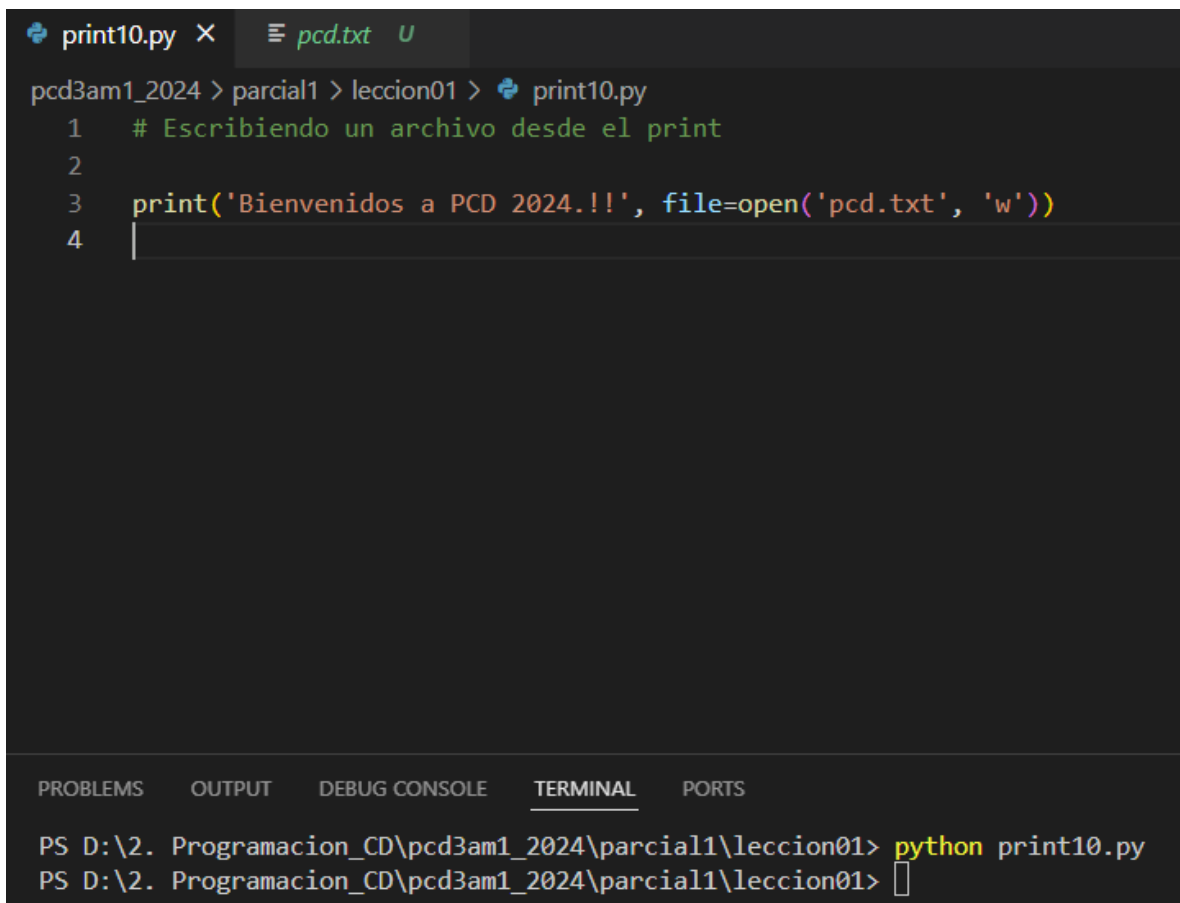
The image shows a code editor window with a file named `print09.py`. The code in the editor is as follows:

```
1 # Separando datos con un separador especial
2
3 d=27
4 m=9
5 a=2024
6 print(d,m,a,sep="-")
```

Below the editor is a terminal window. The first command executed is `python print09.py`, which outputs `27-9-2024`. The second command is a prompt `PS E:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01>` with a cursor.

Aquí se usa un parámetro llamado `sep` que hace que las variables declaradas anteriormente estén separadas por un –

print10.py

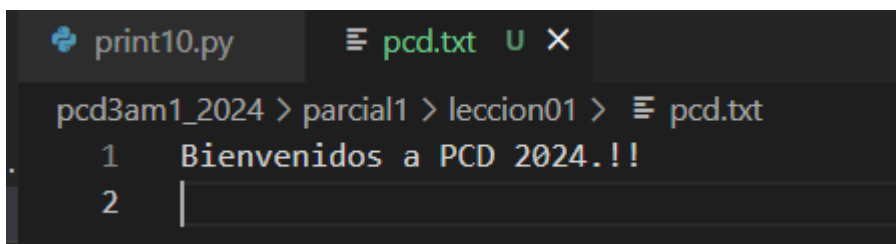


The image shows a VS Code editor with two tabs: 'print10.py' and 'pcd.txt'. The 'print10.py' tab is active, displaying the following code:

```
1 # Escribiendo un archivo desde el print
2
3 print('Bienvenidos a PCD 2024.!!', file=open('pcd.txt', 'w'))
4
```

Below the editor, the 'TERMINAL' panel is open, showing the command prompt output:

```
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print10.py
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01>
```

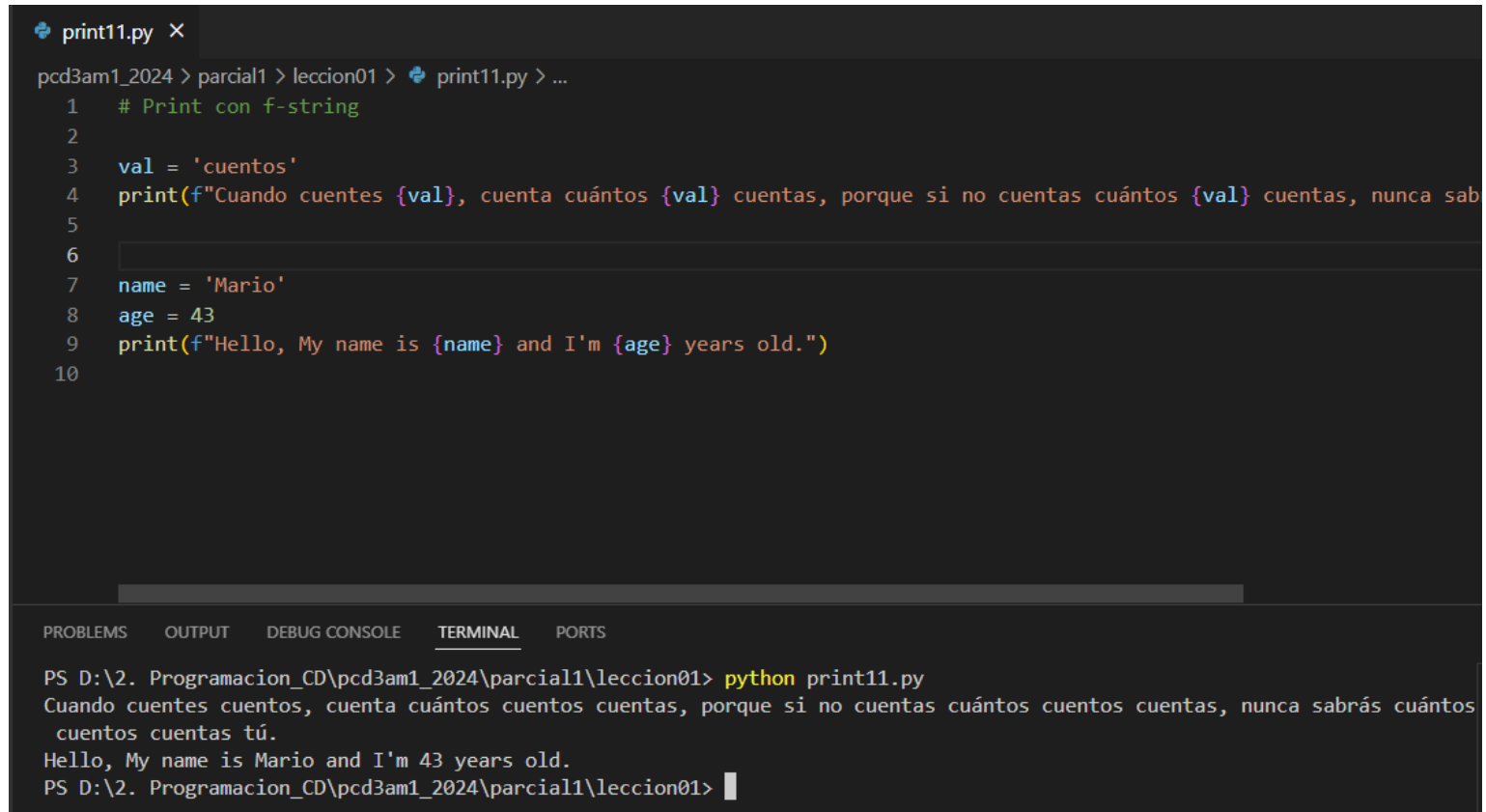


The image shows the 'pcd.txt' tab in the VS Code editor, which contains the output of the Python script:

```
1 Bienvenidos a PCD 2024.!!
2
```

Este programa crea un archivo txt desde el print con la implementación de file = open, sino existe el archivo se crea y después permite escribir gracias al parámetro w

print11.py



The image shows a code editor window with a file named 'print11.py'. The code is written in Python and uses f-strings for formatting. It defines a variable 'val' with the value 'cuentos' and a variable 'name' with the value 'Mario'. It then prints a sentence using 'val' and another sentence using 'name' and 'age' (43). Below the code, the terminal output shows the command 'python print11.py' being executed, resulting in the printed text.

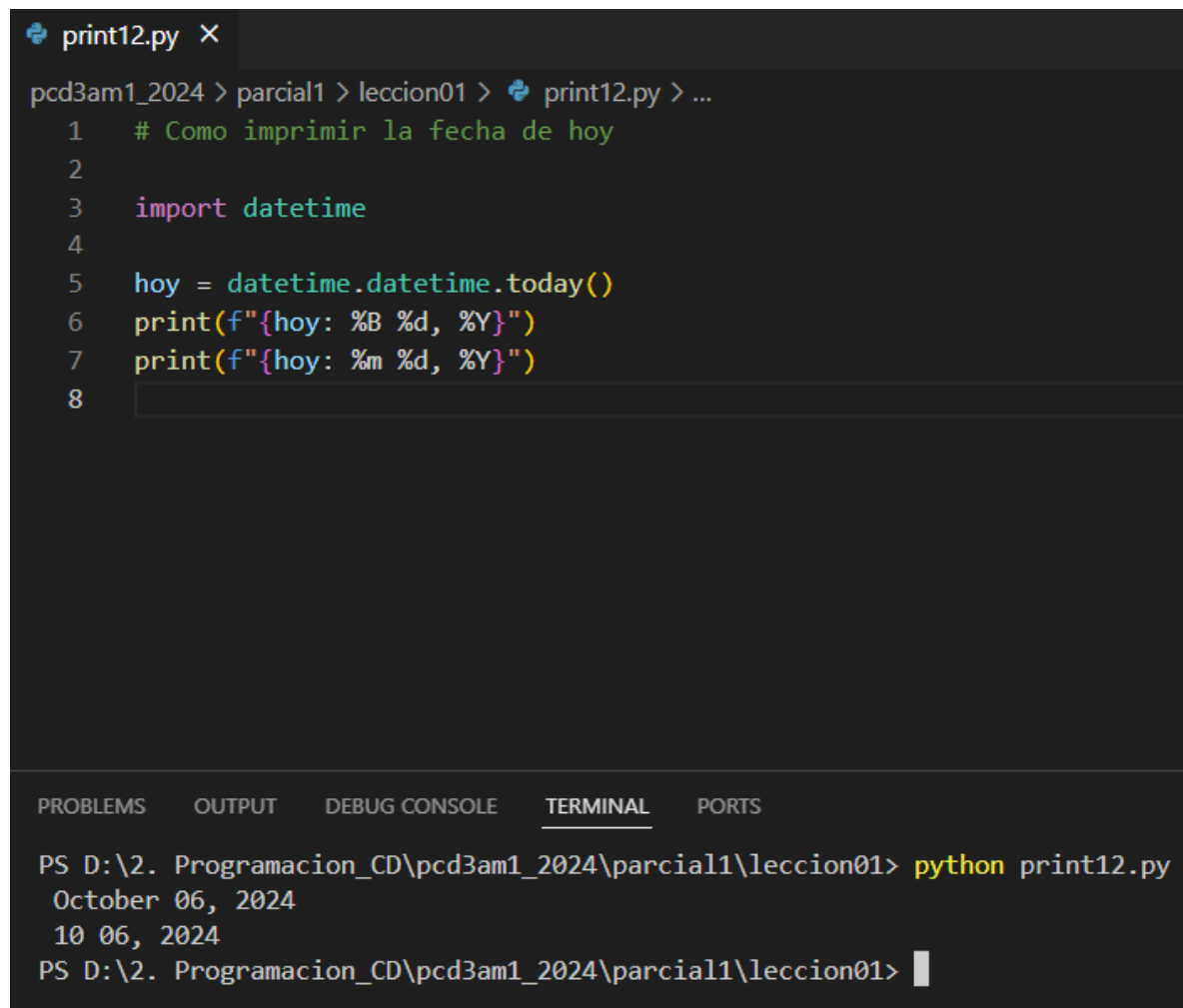
```
print11.py X
pcd3am1_2024 > parcial1 > leccion01 > print11.py > ...
1  # Print con f-string
2
3  val = 'cuentos'
4  print(f"Cuando cuentas {val}, cuenta cuántos {val} cuentas, porque si no cuentas cuántos {val} cuentas, nunca sab
5
6
7  name = 'Mario'
8  age = 43
9  print(f"Hello, My name is {name} and I'm {age} years old.")
10

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print11.py
Cuando cuentas cuentos, cuenta cuántos cuentos cuentas, porque si no cuentas cuántos cuentos cuentas, nunca sabrás cuántos
cuentos cuentas tú.
Hello, My name is Mario and I'm 43 years old.
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

Antes del primer printf se declara una variable que después cuando queremos imprimir el texto, antes de abrir comillas se usa una f y cada que se quiera usar la palabra declarada en la variable, solo abrimos {} y adentro ponemos el nombre de la variable a usar.

print12.py



The image shows a code editor window with a file named 'print12.py' and a terminal window below it. The code in the editor uses the 'datetime' module to get the current date and print it in two different formats. The terminal shows the command 'python print12.py' being executed, resulting in two lines of output: 'October 06, 2024' and '10 06, 2024'.

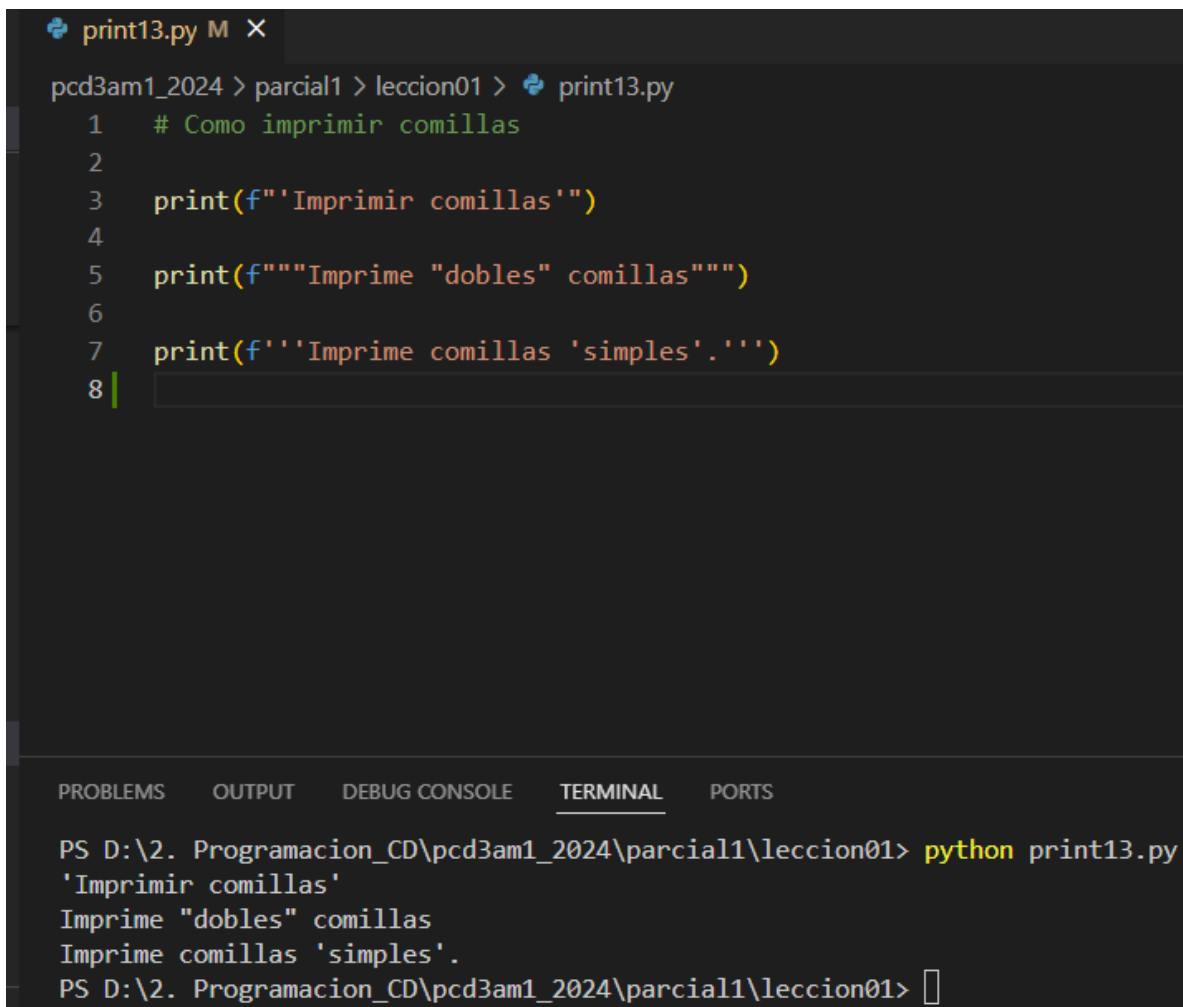
```
print12.py X
pcd3am1_2024 > parcial1 > leccion01 > print12.py > ...
1  # Como imprimir la fecha de hoy
2
3  import datetime
4
5  hoy = datetime.datetime.today()
6  print(f"{hoy: %B %d, %Y}")
7  print(f"{hoy: %m %d, %Y}")
8

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print12.py
October 06, 2024
10 06, 2024
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> |
```

Este código es para cuando queremos imprimir la fecha del día en el que se está ejecutando el programa, se usa la librería datetime y después nos muestran dos formas de imprimir la fecha.

%B es cuando se quiere imprimir el mes completo y %m cuando solo se quiere imprimir el numero del mes.

print13.py



The image shows a code editor window with a file named `print13.py`. The code contains three lines of Python code demonstrating different ways to print text with quotes. Below the code editor is a terminal window showing the command `python print13.py` being executed, followed by the output of the program.

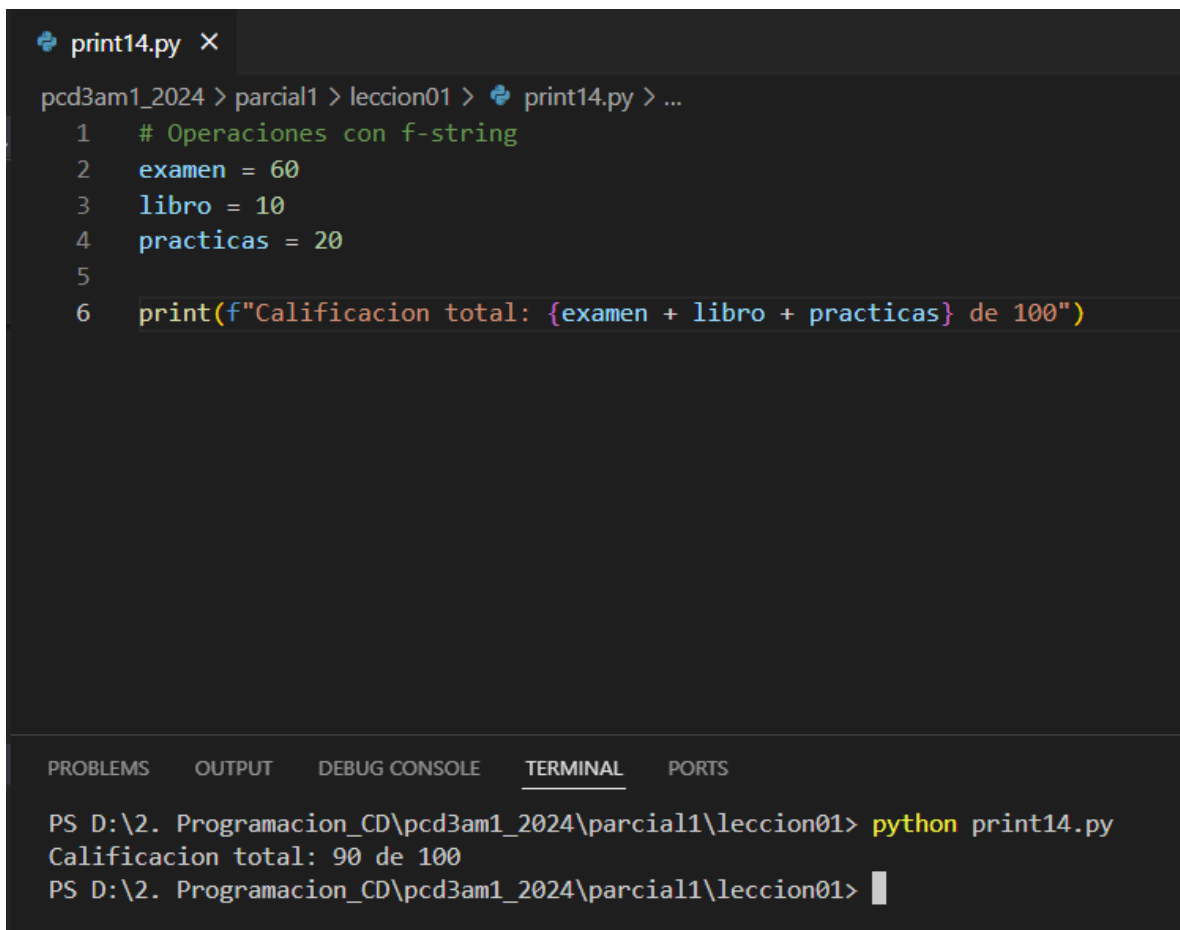
```
print13.py M X
pcd3am1_2024 > parcial1 > leccion01 > print13.py
1  # Como imprimir comillas
2
3  print(f'Imprimir comillas')
4
5  print(f"""Imprime "dobles" comillas""")
6
7  print(f'''Imprime comillas 'simples'.''')
8

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print13.py
'Imprimir comillas'
Imprime "dobles" comillas
Imprime comillas 'simples'.
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

Aquí se nos muestran tres diferentes formas de imprimir comillas

print14.py



The image shows a code editor window with a file named 'print14.py' open. The code in the editor is as follows:

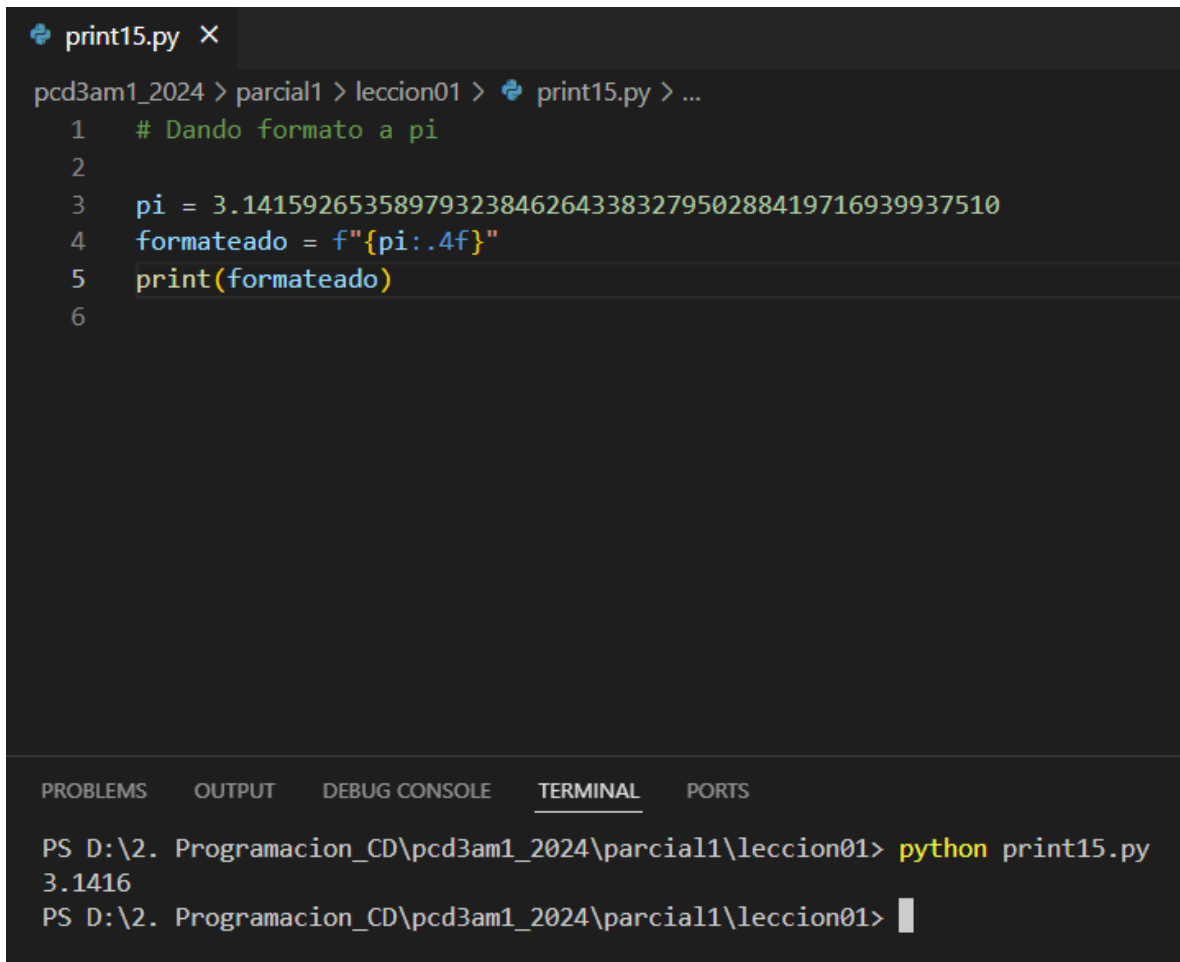
```
1 # Operaciones con f-string
2 examen = 60
3 libro = 10
4 practicas = 20
5
6 print(f"Calificacion total: {examen + libro + practicas} de 100")
```

Below the code editor is a terminal window. The terminal shows the command to run the script and its output:

```
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print14.py
Calificacion total: 90 de 100
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01>
```

Aquí se usa la operación básica de la suma para sumar las tres variables declaradas implementando el f – string

print15.py



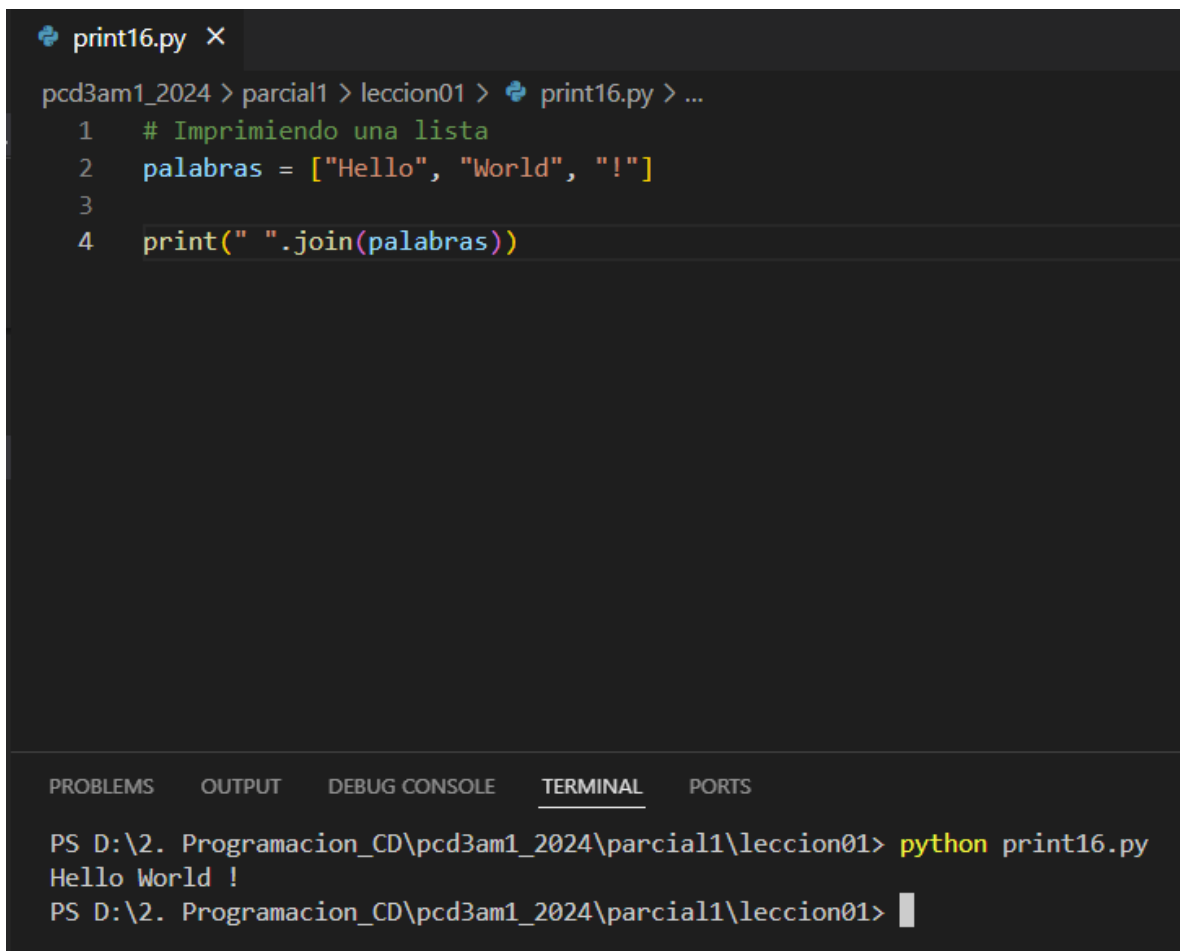
The image shows a code editor window with a tab labeled 'print15.py'. The editor contains a Python script with six lines of code. The first line is a comment. The second line is empty. The third line assigns a long decimal value to the variable 'pi'. The fourth line uses an f-string to format 'pi' to four decimal places. The fifth line prints the formatted string. The sixth line is empty. Below the editor, there is a terminal window with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active, showing the command 'python print15.py' being executed in a PowerShell prompt, followed by the output '3.1416'.

```
print15.py X
pcd3am1_2024 > parcial1 > leccion01 > print15.py > ...
1  # Dando formato a pi
2
3  pi = 3.14159265358979323846264338327950288419716939937510
4  formateado = f"{pi:.4f}"
5  print(formateado)
6

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print15.py
3.1416
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

En este código se le da formato a pi, reduciendo su gran cantidad de decimales a solo 4 usando {pi: .4f}. la f es de float

print16.py



The image shows a code editor window titled "print16.py" with a dark background. The code is as follows:

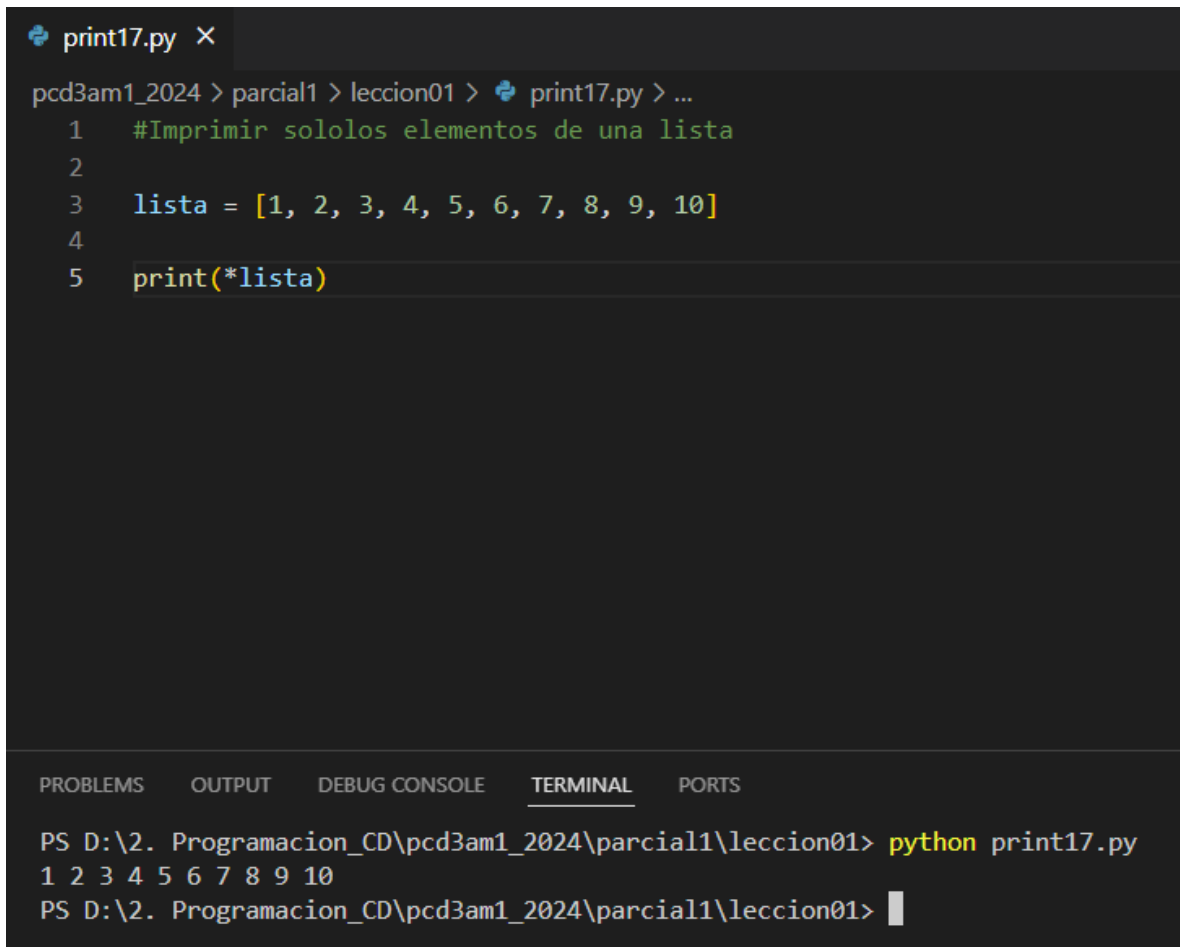
```
1  # Imprimiendo una lista
2  palabras = ["Hello", "World", "!"]
3
4  print(" ".join(palabras))
```

Below the code editor is a terminal window with tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The "TERMINAL" tab is active, showing the command prompt and the execution of the script:

```
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print16.py
Hello World !
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

Aquí se hace uso de la función join que hace que los elementos de la lista palabras se unan e imprima todo junto

print17.py



The image shows a code editor window with a file named 'print17.py'. The code in the editor is as follows:

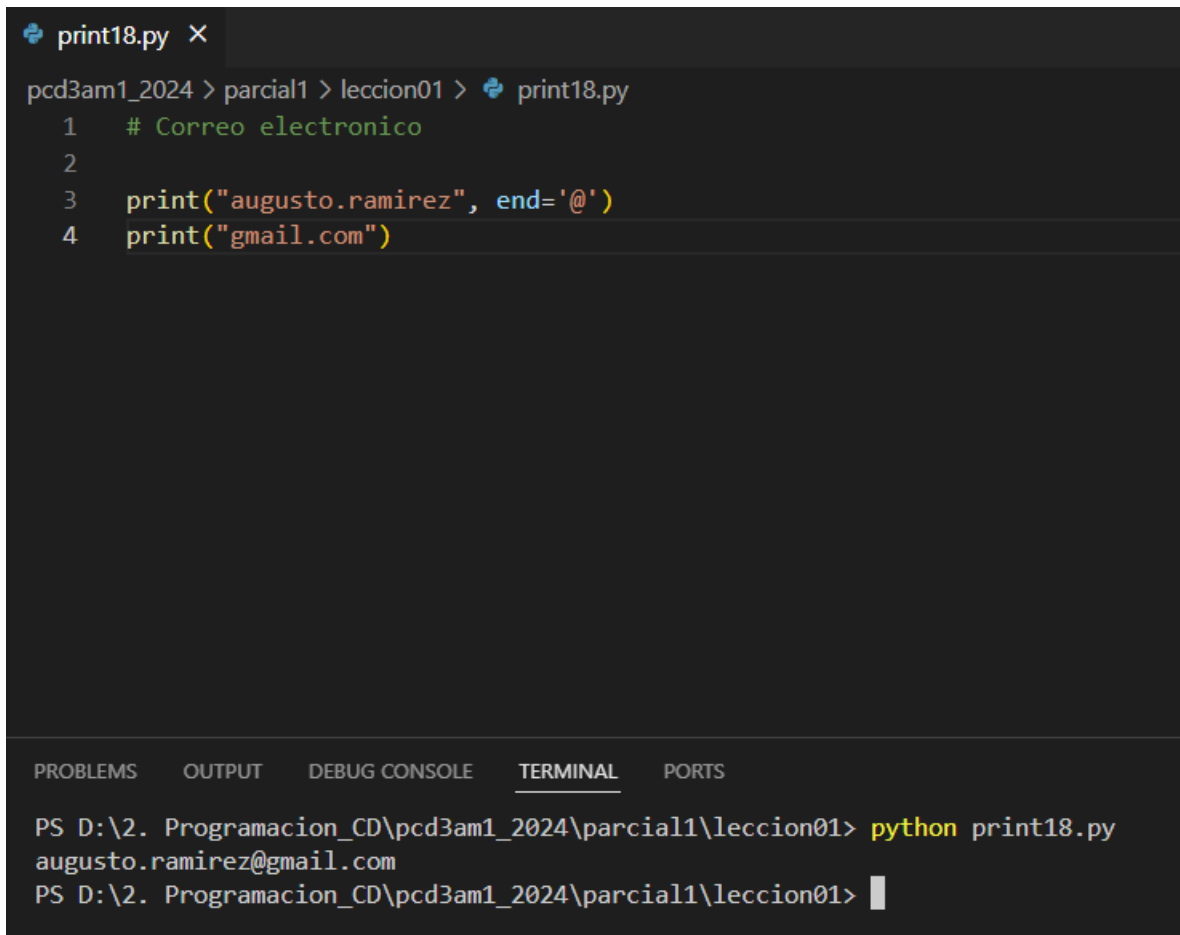
```
1  #Imprimir sololos elementos de una lista
2
3  lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
4
5  print(*lista)
```

Below the code editor is a terminal window. The terminal shows the command to run the script and its output:

```
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print17.py
1 2 3 4 5 6 7 8 9 10
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01>
```

Aquí se tiene una lista cualquiera, la diferencia es que al imprimirla se pone un * antes del nombre de la lista, lo que hace que se impriman solo los valores de la lista, es decir, sin los []

print18.py



The image shows a code editor window with a file named `print18.py`. The code contains four lines: a comment, a blank line, and two `print` statements. The first `print` statement uses the `end` parameter to append '@' to the output. Below the code editor is a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The terminal shows the command `python print18.py` being executed, resulting in the output `augusto.ramirez@gmail.com`.

```
print18.py X
pcd3am1_2024 > parcial1 > leccion01 > print18.py
1  # Correo electronico
2
3  print("augusto.ramirez", end='@')
4  print("gmail.com")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print18.py
augusto.ramirez@gmail.com
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

Como ya se había visto en ejercicios anteriores, se usa la palabra `end` que indica que en lugar de dar un salto de línea añadirá al final un `@`

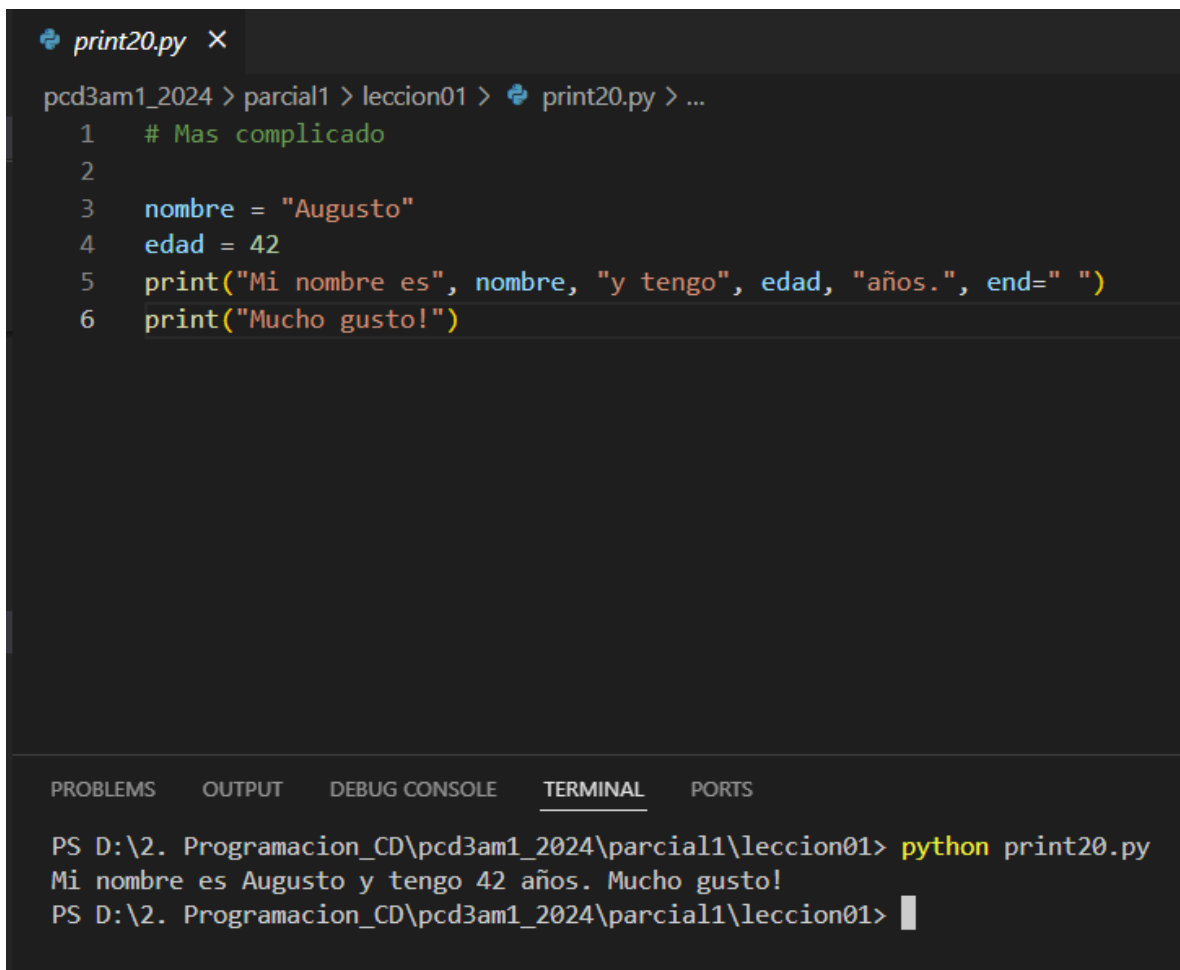
print19.py

```
print19.py X
pcd3am1_2024 > parcial1 > leccion01 > print19.py
1  # Experimentos con el fin de linea
2
3  print('C','D','M', sep='', end='')
4  print('X')
5
6  # Despues de un print sin el delimitador end, se restablece el fin de linea
7  print('27','09','2024', sep='-', end='\n')
8
9
10 # Otro fin de linea
11 print('VERDE','BLANCO','ROJO', sep='|', end='@')
12 print('mexico')
13

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print19.py
CDMX
27-09-2024
VERDE|BLANCO|ROJO@mexico
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

En este programa se implementa una combinación del sep y del end, mostrándonos de diferentes formas en las que se puede aplicar.

print20.py



The image shows a code editor window with a tab labeled `print20.py`. The editor contains the following Python code:

```
1  # Mas complicado
2
3  nombre = "Augusto"
4  edad = 42
5  print("Mi nombre es", nombre, "y tengo", edad, "años.", end=" ")
6  print("Mucho gusto!")
```

Below the editor is a terminal window with tabs for `PROBLEMS`, `OUTPUT`, `DEBUG CONSOLE`, `TERMINAL` (selected), and `PORTS`. The terminal shows the command `python print20.py` being executed, resulting in the output: `Mi nombre es Augusto y tengo 42 años. Mucho gusto!`. The prompt `PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01>` is visible before and after the command.

Aquí se nos muestra otra forma de imprimir variables y al final viene la implementación del end

print21.py

```
print21.py M X
pcd3am1_2024 > parcial1 > leccion01 > print21.py > ...
1  # Formato con el caracter (%)
2  pi = 3.14159265358979323846264338327950288419716939937510
3
4  print("Estudiantes : %2d, Edad promedio : %6.2f" % (35, 019.333))
5
6  print("Hombres : %3d, Mujeres : %2d" % (20, 15))
7
8  print("Octal: %7.3o" % (25))    # Imprimir en Octal
9
10 print("Pi: %10.4E" % (pi))    # Notacion exponencial
11

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print21.py
Estudiantes : 35, Edad promedio :  19.33
Hombres :   20, Mujeres :  15
Octal:      031
Pi: 3.1416E+00
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

print22.py

```
print22.py X
pcd3am1_2024 > parcial1 > leccion01 > print22.py
1  # Diferentes formas de imprimir con formato
2
3  print('I love {}. "{}!"'.format('this game!', 'Just do it!'))
4
5  print('{0} and {1}'.format('I love this game!', 'Just do it!'))
6
7  print('{1} and {0}'.format('I love this game!', 'Just do it!'))
8
9  print(f"I love {'this game'}! and \"{'Just do it'}!\")
10
11 print(f'{'I love this game!'} and {'Just do it!'}")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print22.py
I love this game!. "Just do it!!"
I love this game! and Just do it!
Just do it! and I love this game!
I love this game! and "Just do it!"
I love this game! and Just do it!
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

Aquí se muestran diferentes formas para imprimir con .format

print23.py

```
print23.py X
pcd3am1_2024 > parcial1 > leccion01 > print23.py
1  # argumentos por posicion y por nombre
2  print('El mejor equipo {0}, el segundo {1}, y el tercero {otro}.')
3  |   .format('CELTICS', 'NUGGETS', otro='BULLS')
4
5  # con format, posiciones y formato
6  print("Primera posicion, entero de un digito:>>{0:3d}<<, segunda posicion flotante:>>{1:8.2f}<<".
7  |   format(12, 00.546))
8
9  # Cambiando posiciones
10 print("segundo argumento flotante:>>{1:8.2f}<< primer argumento entero:>>{0:3d}<<, ".
11 |   format(12, 00.546))
12
13 # Argumentos por nombre
14 print("a: {a:5d}, Portbal: {b:8.2f}").
15 |   format(a = 1234, b = 19.123456789))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print23.py
El mejor equipo CELTICS, el segundo NUGGETS, y el tercero BULLS.
Primera posicion, entero de un digito:>> 12<<, segunda posicion flotante:>>    0.55<<
segundo argumento flotante:>>    0.55<< primer argumento entero:>> 12<<,
a: 1234, Portbal:    19.12
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> |
```

Aquí se ven las diferentes formas de imprimir y usar format, así como el uso de decimales

print24.py

```
print24.py X
pcd3am1_2024 > parcial1 > leccion01 > print24.py > ...
1  texto = "BOSTON Celtics"
2
3  # Centrado
4  print("Texto centrado y lleno con #:")
5  print(texto.center(40, '#'))
6
7  # Alineacion a la izquierda
8  print("Alineado a la izquierda : ")
9  print(texto.ljust(40, '-'))
10
11 # Alineacion a la derecha
12 print("Alineado a la derecha : ")
13 print(texto.rjust(40, '*'))
14

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> python print24.py
Texto centrado y lleno con #:
#####BOSTON Celtics#####
Alineado a la izquierda :
BOSTON Celtics-----
Alineado a la derecha :
*****BOSTON Celtics
PS D:\2. Programacion_CD\pcd3am1_2024\parcial1\leccion01> 
```

En este código se pueden apreciar las diferentes opciones que tenemos para poder imprimir texto, ya sea centrado, a la izquierda o a la derecha.