

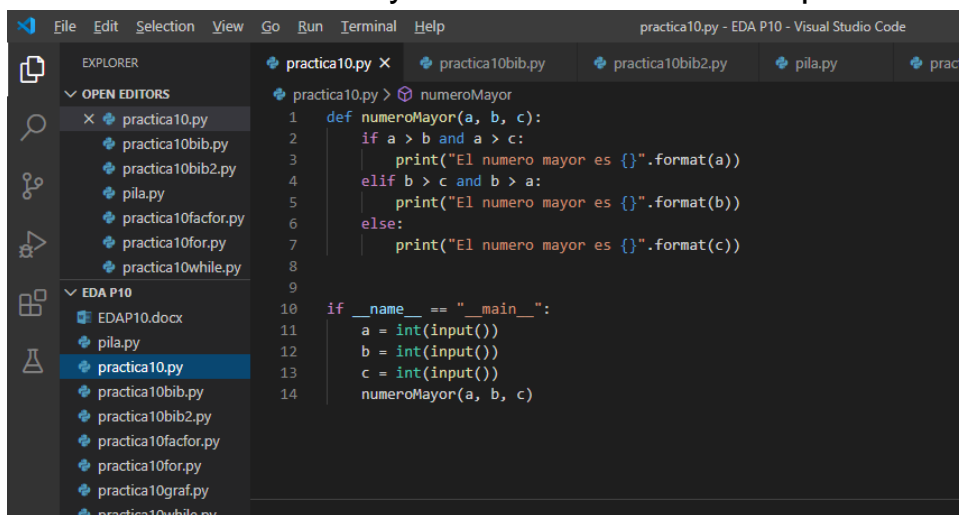
Práctica 10

Introducción

En esta práctica realizamos varios programas en los que conocemos estructuras selectivas, iteración pero esta vez en el lenguaje python tales como; if, elif, else, while, for

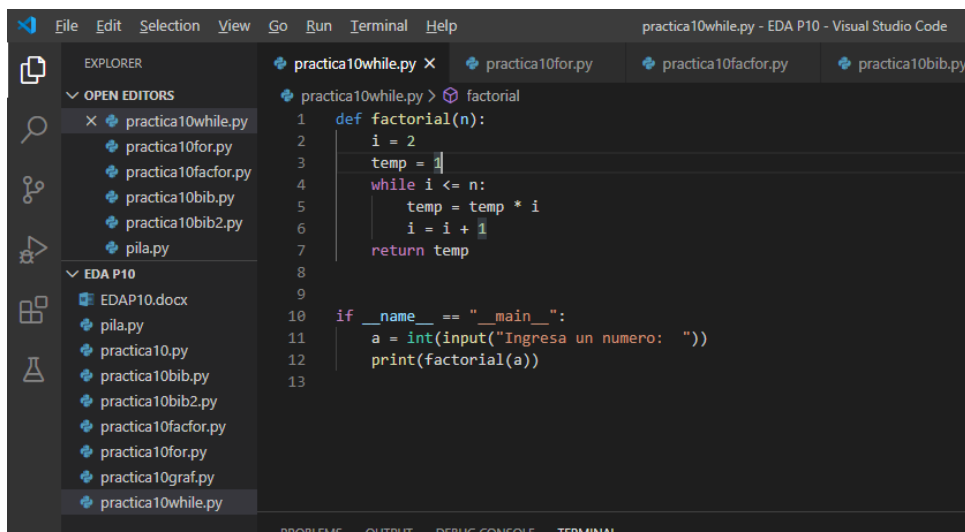
➤ Desarrollo (con ejercicios)

- En el primer ejercicio utilizamos if, elif y else para obtener el numero mayor de 3 números dados por el usuario



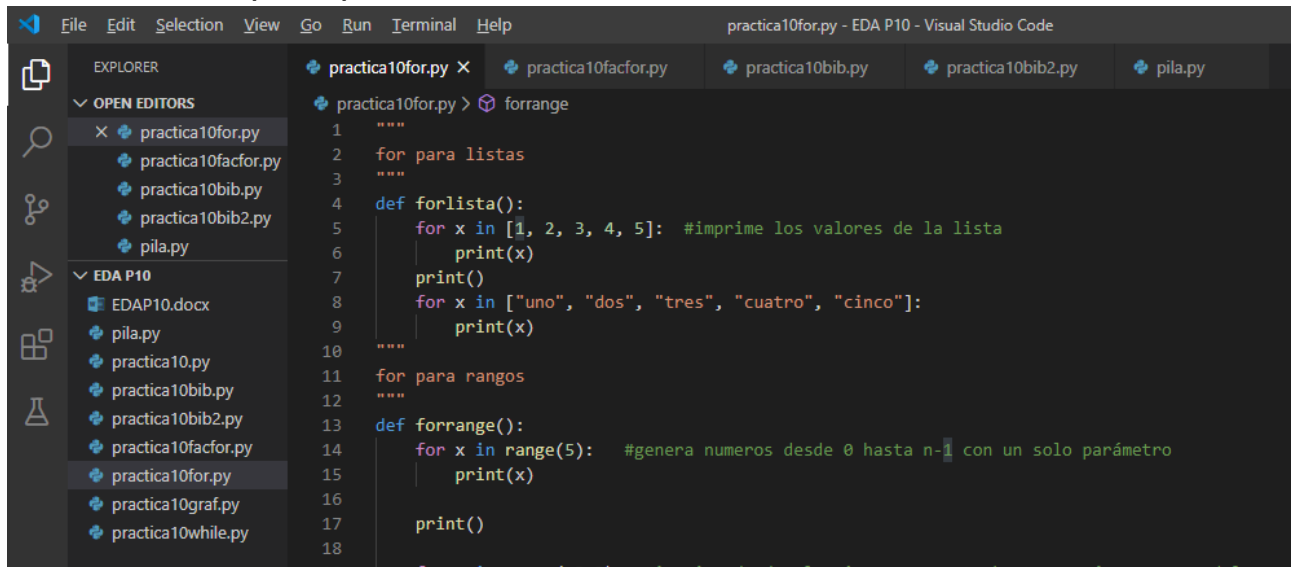
```
practica10.py X | practica10bib.py | practica10bib2.py | pila.py | practi
practica10.py > numeroMayor
1 def numeroMayor(a, b, c):
2     if a > b and a > c:
3         print("El numero mayor es {}".format(a))
4     elif b > c and b > a:
5         print("El numero mayor es {}".format(b))
6     else:
7         print("El numero mayor es {}".format(c))
8
9
10 if __name__ == "__main__":
11     a = int(input())
12     b = int(input())
13     c = int(input())
14     numeroMayor(a, b, c)
```

- En el segundo caso vimos un modo de iteración; el ciclo while en este programa realizamos la función de obtener el factorial de un numero n auxiliándonos de una variable temp que será la que almacene el valor de los cálculos realizados durante el ciclo



```
practica10while.py X | practica10for.py | practica10facfor.py | practica10bib.py
practica10while.py > factorial
1 def factorial(n):
2     i = 2
3     temp = 1
4     while i <= n:
5         temp = temp * i
6         i = i + 1
7     return temp
8
9
10 if __name__ == "__main__":
11     a = int(input("Ingresa un numero: "))
12     print(factorial(a))
13
```

- Después de eso continuamos con otro ciclo de iteración. Esta vez con el ciclo for que resulta ser un tanto curioso ya que la sintaxis a comparación del lenguaje C cambia, así como los valores que toma, en este primer caso vimos el for para listas En la cual va a tomar cada uno de los valores recorriéndola de principio a fin

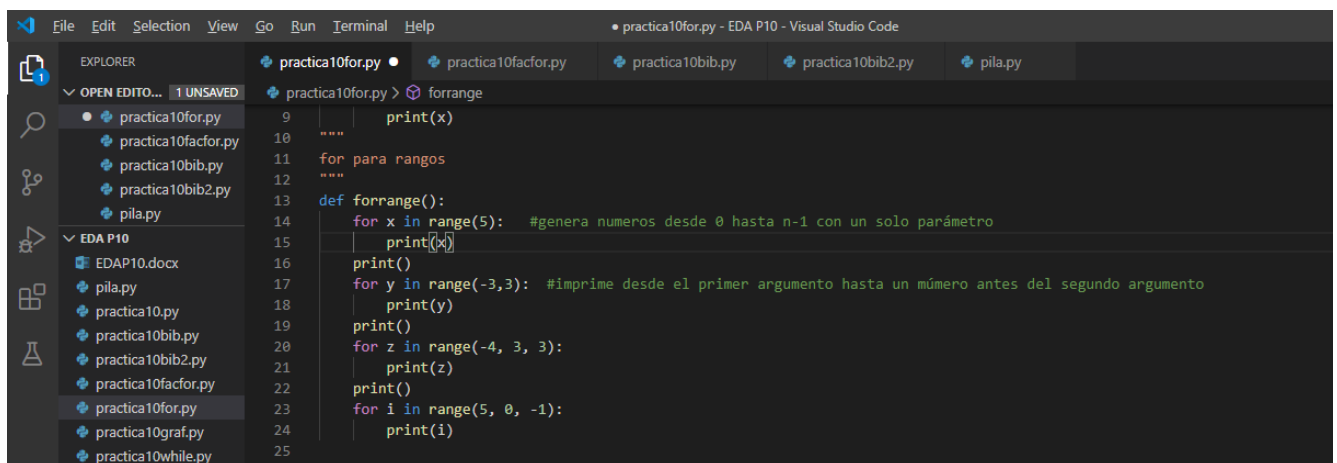


```

1  """
2  for para listas
3  """
4  def forlista():
5      for x in [1, 2, 3, 4, 5]: #imprime los valores de la lista
6          print(x)
7      print()
8      for x in ["uno", "dos", "tres", "cuatro", "cinco"]:
9          print(x)
10 """
11 for para rangos
12 """
13 def forrange():
14     for x in range(5): #genera numeros desde 0 hasta n-1 con un solo parámetro
15         print(x)
16
17     print()
18
19     for y in range(-3, 3): #imprime desde el primer argumento hasta un número antes del segundo argumento
20         print(y)
21     print()
22     for z in range(-4, 3, 3):
23         print(z)
24     print()
25     for i in range(5, 0, -1):
26         print(i)

```

- En el for también podemos colocar rangos de inicio, de final y el incremento tal como el decremento en el que queramos

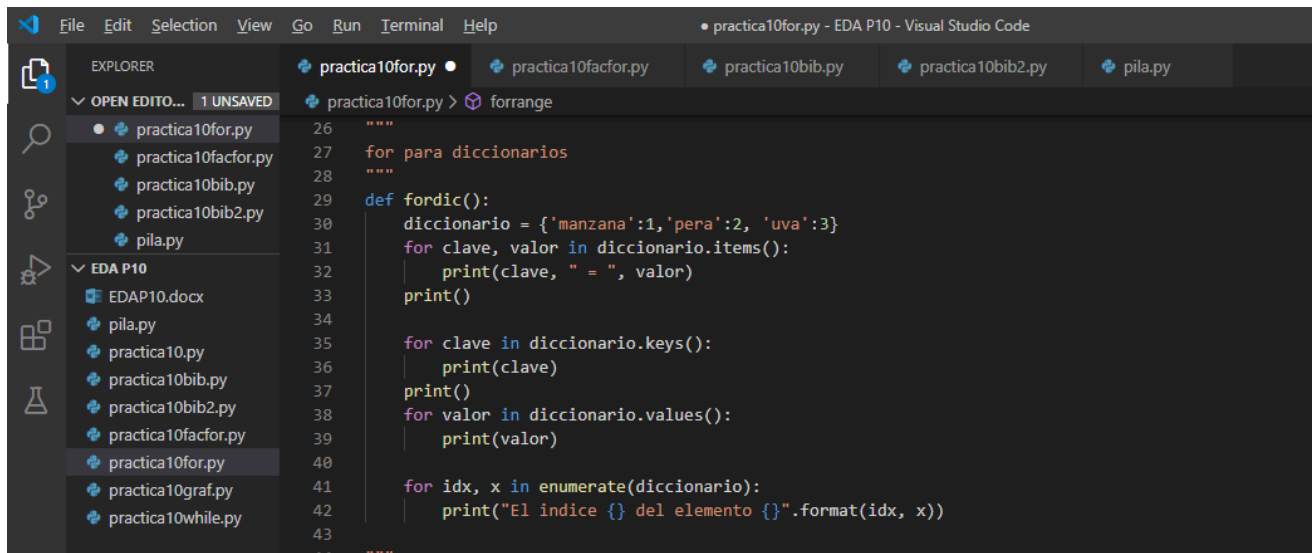


```

9      print(x)
10 """
11 for para rangos
12 """
13 def forrange():
14     for x in range(5): #genera numeros desde 0 hasta n-1 con un solo parámetro
15         print(x)
16     print()
17     for y in range(-3, 3): #imprime desde el primer argumento hasta un número antes del segundo argumento
18         print(y)
19     print()
20     for z in range(-4, 3, 3):
21         print(z)
22     print()
23     for i in range(5, 0, -1):
24         print(i)
25
26 """

```

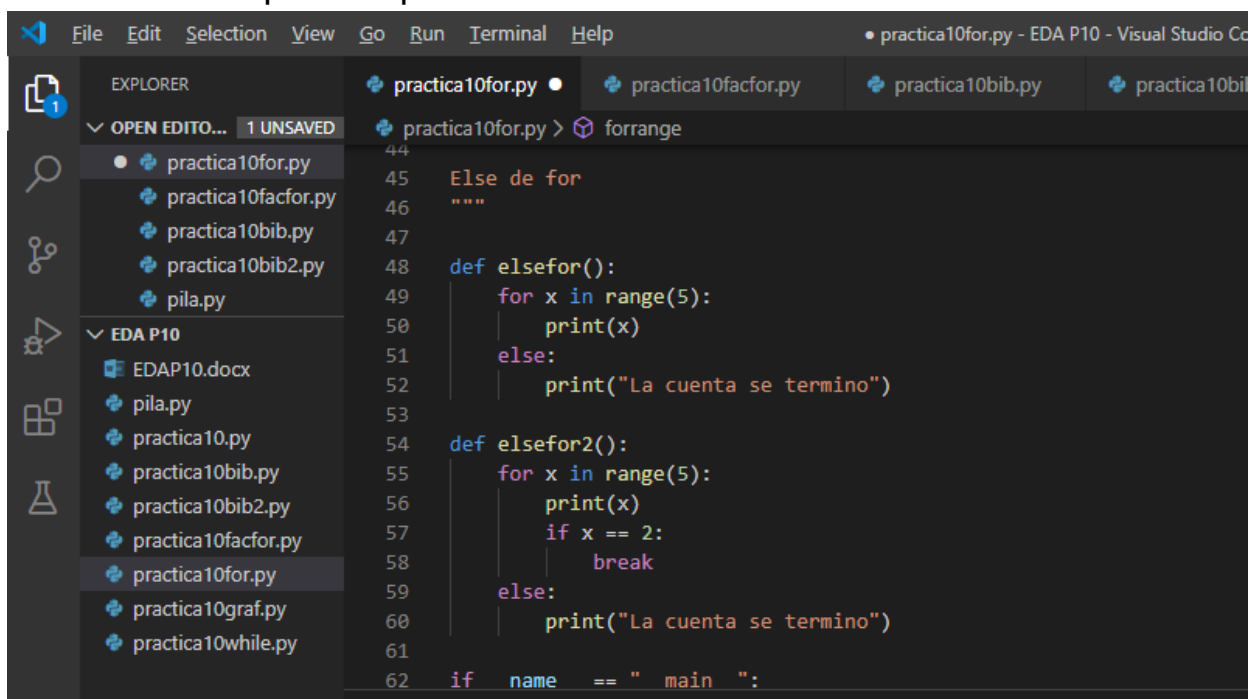
- En python existe algo llamado diccionarios que se definen con valor y clave, podemos recuperar ya sea el valor o la clave de cada diccionario



The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor on the right. The file explorer shows a project named 'EDA P10' with several files, including 'practica10for.py'. The code editor displays the content of 'practica10for.py', which contains a for loop over a dictionary. The code is as follows:

```
26 """
27 for para diccionarios
28 """
29 def fordic():
30     diccionario = {'manzana':1, 'pera':2, 'uva':3}
31     for clave, valor in diccionario.items():
32         print(clave, " = ", valor)
33     print()
34
35     for clave in diccionario.keys():
36         print(clave)
37     print()
38     for valor in diccionario.values():
39         print(valor)
40
41     for idx, x in enumerate(diccionario):
42         print("El indice {} del elemento {}".format(idx, x))
43 """
```

- Y por último encontramos una situación curiosa sobre un else de for en el que una vez terminada la iteración ejecutará las instrucciones dadas en el for, si y solo si no hay algo que rompa de repente el ciclo



The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor on the right. The file explorer shows a project named 'EDA P10' with several files, including 'practica10for.py'. The code editor displays the content of 'practica10for.py', which contains a for loop with an else clause. The code is as follows:

```
44
45 Else de for
46 """
47
48 def elsefor():
49     for x in range(5):
50         print(x)
51     else:
52         print("La cuenta se termino")
53
54 def elsefor2():
55     for x in range(5):
56         print(x)
57         if x == 2:
58             break
59     else:
60         print("La cuenta se termino")
61
62 if name == " main ":
```

- El profesor solicitó que realizáramos un programa que calcule el factorial de un número pero esta vez en lugar de un ciclo while nos indicó que fuese con el ciclo for para que pudiésemos poner en práctica lo estudiado, este es programa que diseñé, funciona para los casos como el factorial de 0 y 1 que podemos considerar como casos base

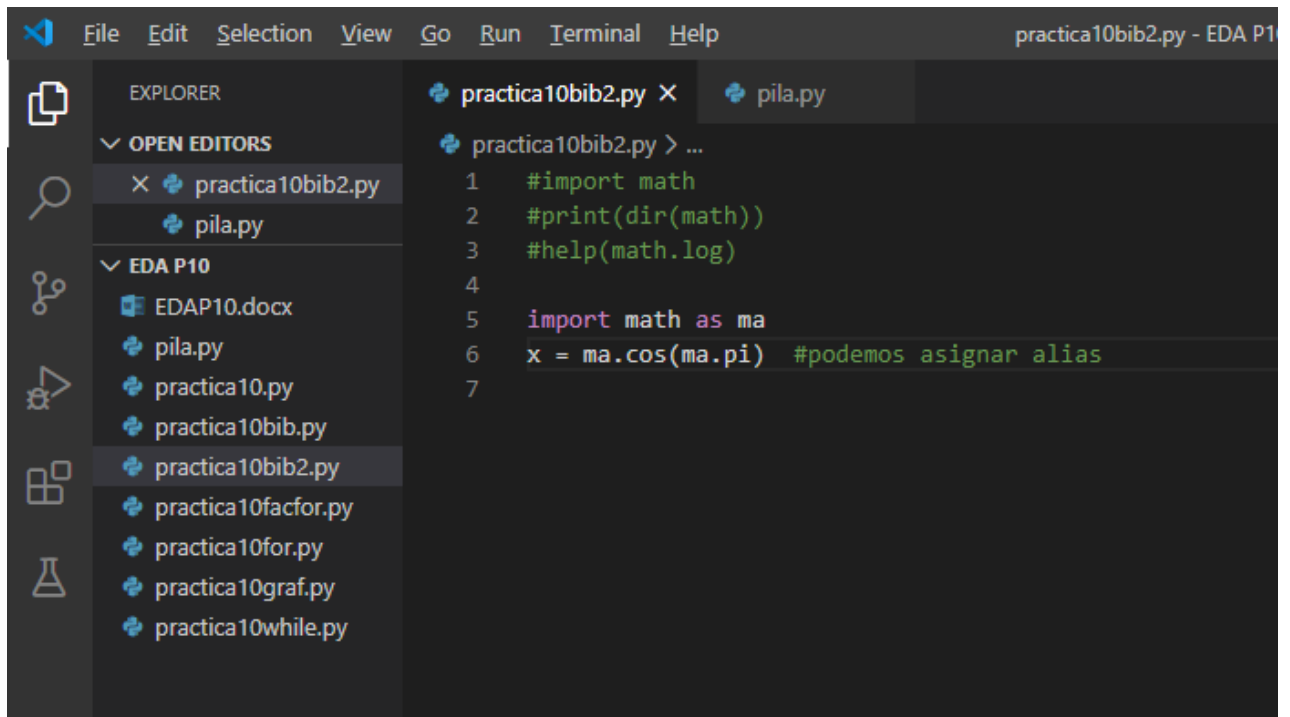
The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor on the right. The file explorer shows a project named 'EDA P10' containing several files, including 'practica10factor.py'. The code editor displays the following Python code:

```
1 #programar la funcion factorial usando un for
2 def fact(n):
3     x = 1
4     if n>1:
5         for x in range(n-1, 0, -1):
6             n = n*x
7         return n
8     return x
9
10
11 if __name__ == "__main__":
12     a = int(input("Ingresa el numero del cual quieres calcular el factorial:\n"))
13     print("El factorial es:")
14     print(fact(a))
15
16
17
```

- La práctica también muestra la acción de importar bibliotecas en python podemos incluir toda la biblioteca, solo las funciones necesitadas e incluso darle alias a la misma biblioteca

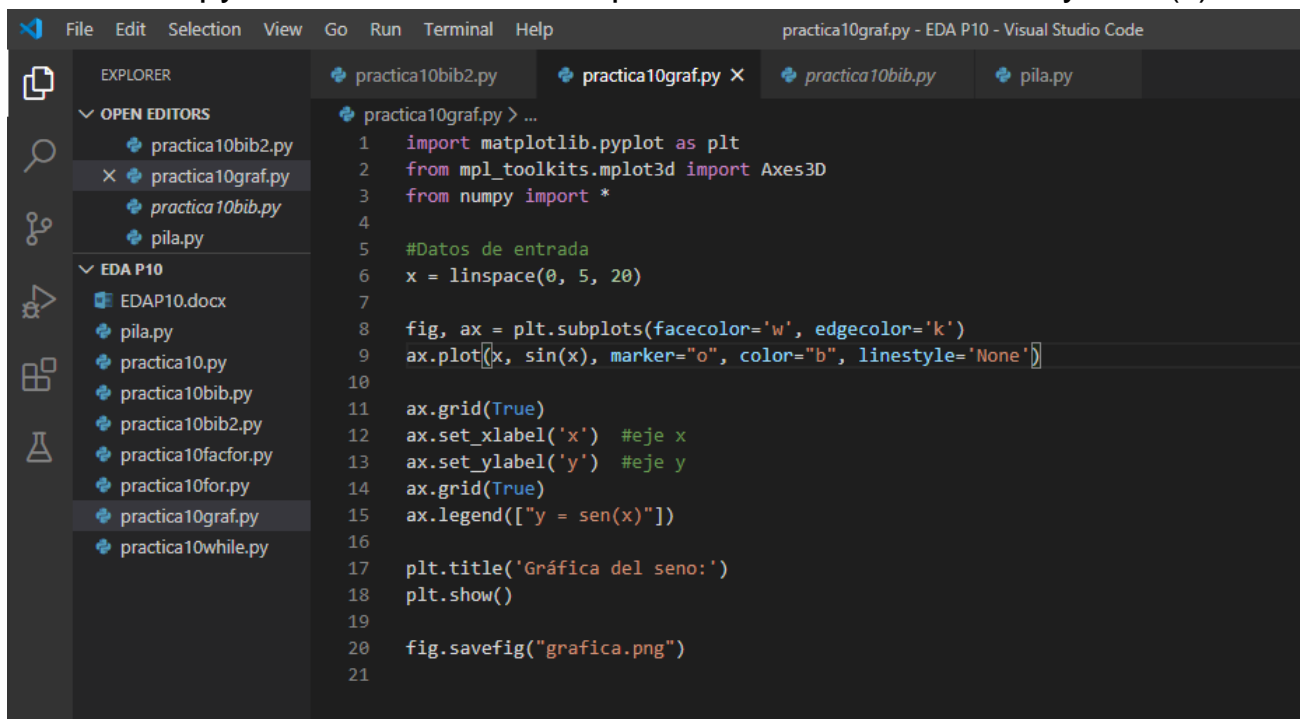
The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor on the right. The file explorer shows a project named 'EDA P10' containing several files, including 'practica10bib.py'. The code editor displays the following Python code:

```
1 #importar bibliotecas
2 """
3 import math
4 x = math.cos(math.pi)
5 print(x)
6 """
7
8 """
9 from math import * #el asterisco le dice que sean todas las funciones de math
10 x = cos(pi)
11 print(x)
12 """
13 from math import cos, pi
14 x = cos(pi)
15 print(x)
```



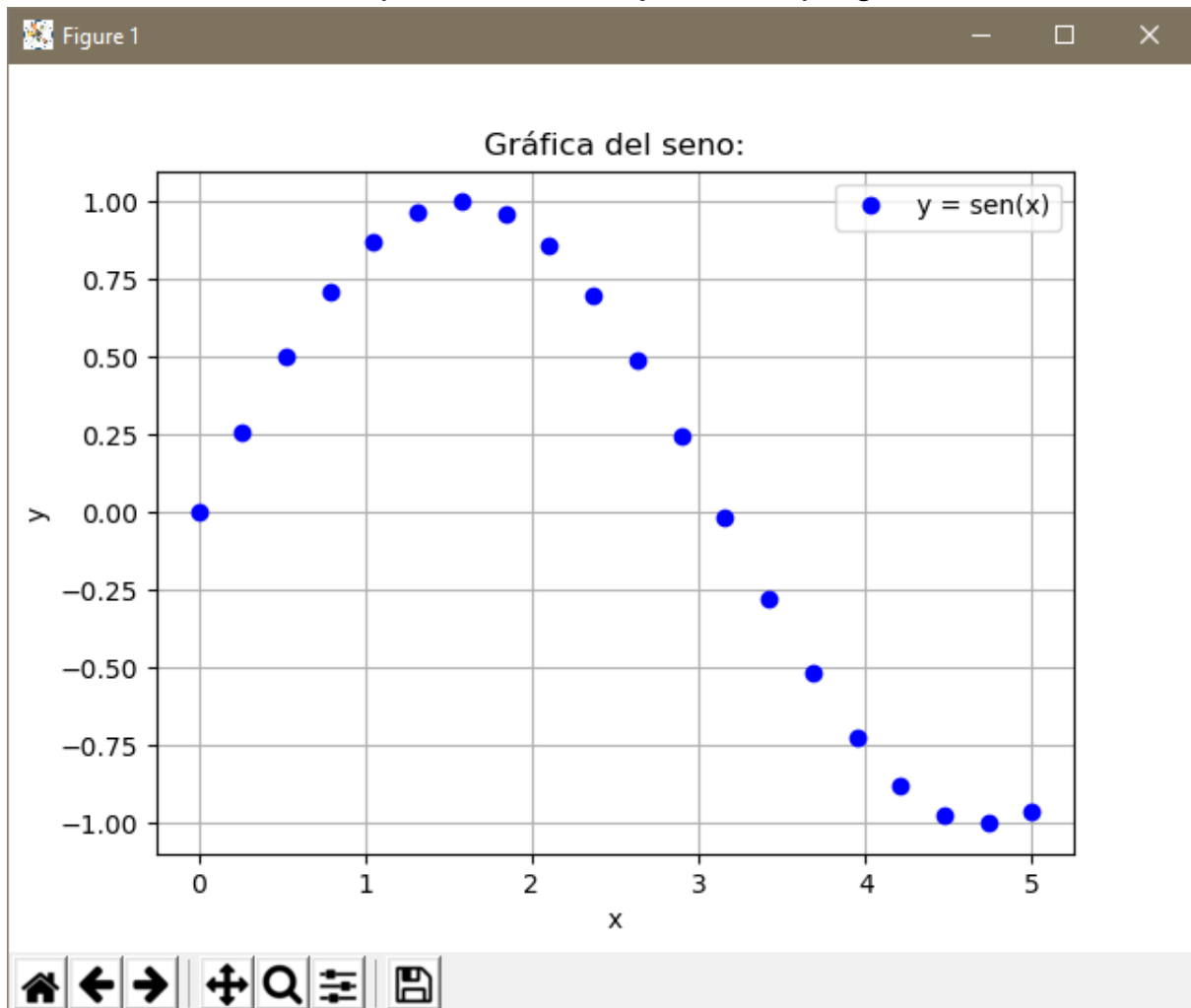
```
practica10bib2.py X pila.py
practica10bib2.py > ...
1  #import math
2  #print(dir(math))
3  #help(math.log)
4
5  import math as ma
6  x = ma.cos(ma.pi) #podemos asignar alias
7
```

- Luego dentro de la práctica hay que hacer una gráfica en python con la librería matplotlib esta es de la función $y = \sin(x)$



```
practica10bib2.py practica10graf.py X practica10bib.py pila.py
practica10graf.py > ...
1  import matplotlib.pyplot as plt
2  from mpl_toolkits.mplot3d import Axes3D
3  from numpy import *
4
5  #Datos de entrada
6  x = linspace(0, 5, 20)
7
8  fig, ax = plt.subplots(facecolor='w', edgecolor='k')
9  ax.plot(x, sin(x), marker="o", color="b", linestyle='None')
10
11  ax.grid(True)
12  ax.set_xlabel('x') #eje x
13  ax.set_ylabel('y') #eje y
14  ax.grid(True)
15  ax.legend(["y = sen(x)"])
16
17  plt.title('Gráfica del seno:')
18  plt.show()
19
20  fig.savefig("grafica.png")
21
```

- Esto es lo que muestra al ejecutar el programa



- Para concluir la práctica el profesor indicó que en python es más sencillo diseñar las estructuras de datos, y en este caso diseñamos funciones para la ED-pila

```
pila.py
1 def insertar(lista, dato):
2     lista.append(dato) #agrega al final
3
4 def borrar(lista):
5     dato = lista.pop() #elimina al final
6     return dato
7
8 def imprimir_pila(lista):
9     lista.reverse()
10    for x in lista:
11        print(x)
12    print()
13    lista.reverse()
14
15
```

➤ **Conclusión**

○ **Caballero Hernandez Juan Daniel**

El lenguaje python, resulta más sencillo de leer y es más intuitivo quizá lo que tiene es que en la sintaxis tiene una regla muy estricta en la indentación ya que si no se respeta es 100% seguro que haya un error al momento de ejecutar. Con esta práctica ya podemos diseñar algoritmos que sean transcritos a lenguaje python.