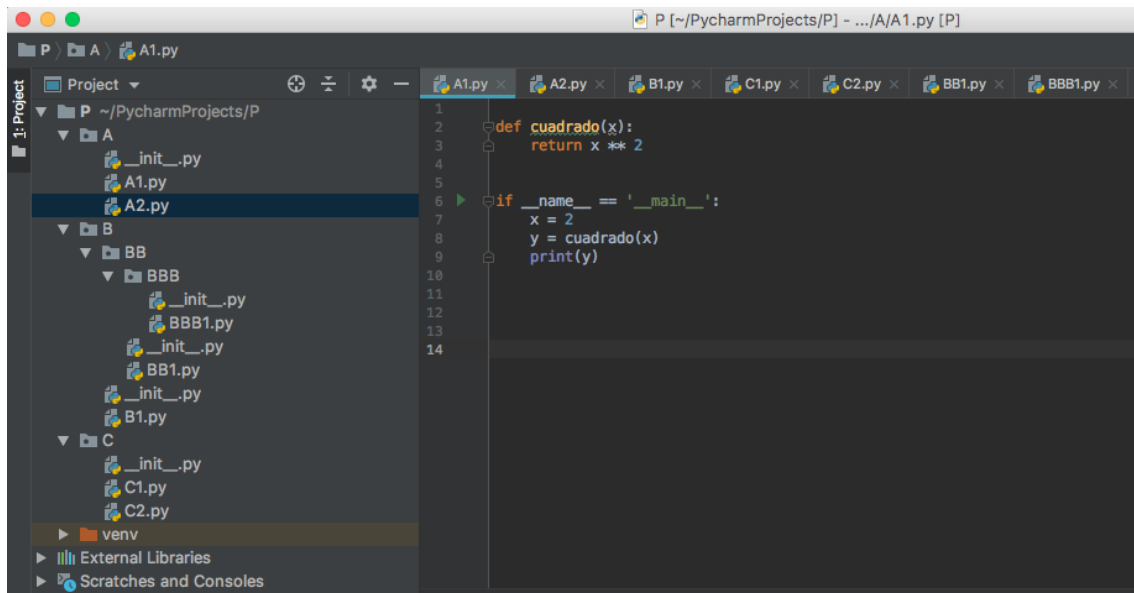


## Ejercicio guiado. Unidad 5.

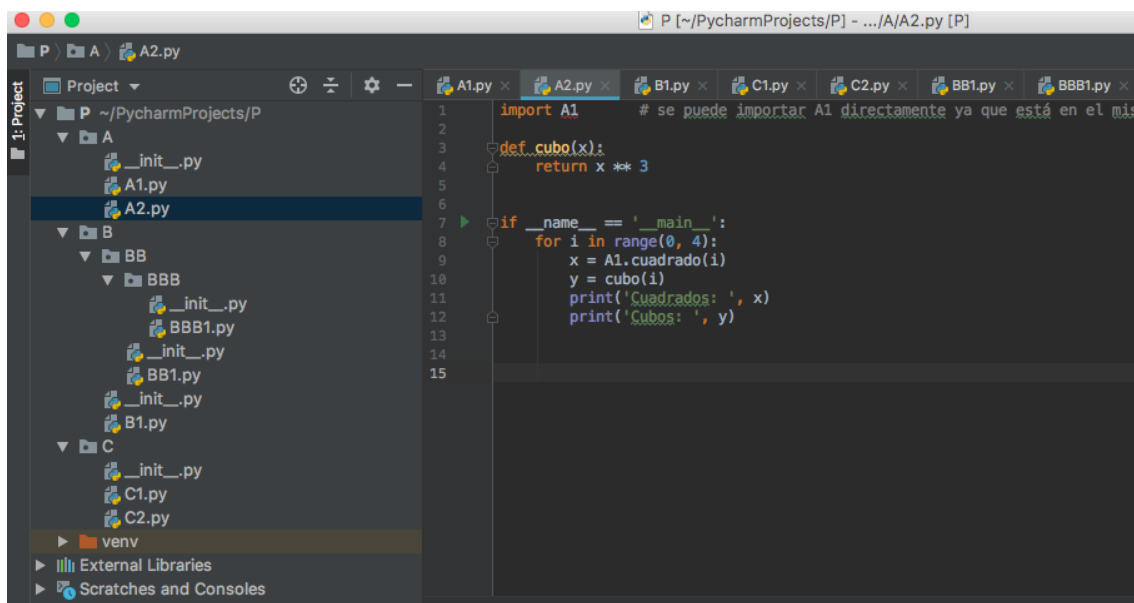
Vamos a considerar la estructura de paquetes, sub-paquetes y módulos del ejercicio anterior, que se vuelve a mostrar en la figura 1 para trabajar las importaciones desde módulos que están y que no están en la mismo paquete. En primer lugar, se ha declarado la función `cuadrado` en el módulo `A1`.



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' view displays a hierarchical structure: a root folder 'P' containing a subfolder 'A'. Inside 'A' are files `__init__.py`, `A1.py`, and `A2.py`. Below 'A' is a subfolder 'B', which contains a subfolder 'BB'. Inside 'BB' are `__init__.py`, `BBB1.py`, `__init__.py`, `BB1.py`, `__init__.py`, and `B1.py`. Below 'B' is a subfolder 'C' containing `__init__.py`, `C1.py`, and `C2.py`. At the bottom of the project view are 'venv', 'External Libraries', and 'Scratches and Consoles'. The main editor window shows the code for `A1.py`:

```
1 def cuadrado(x):
2     return x ** 2
3
4
5
6 if __name__ == '__main__':
7     x = 2
8     y = cuadrado(x)
9     print(y)
10
11
12
13
14
```

A su vez, el código del módulo `A2` se muestra en la figura. En él se ha importado el módulo `A1` y se ha declarado la función `cubo`, que devuelve el cubo del valor que se pasa como argumento. Si se ejecuta este módulo se imprimen los cuadrados y los cubos de los números de 0 a 3.



The screenshot shows the PyCharm IDE interface with the same project structure as the previous figure. The main editor window now shows the code for `A2.py`:

```
1 import A1 # se puede importar A1 directamente ya que está en el mismo
2
3 def cubo(x):
4     return x ** 3
5
6
7 if __name__ == '__main__':
8     for i in range(0, 4):
9         x = A1.cuadrado(i)
10        y = cubo(i)
11        print('Cuadrados: ', x)
12        print('Cubos: ', y)
13
14
15
```

Cuadrados: 0

Cubos: 0

Cuadrados: 1

Cubos: 1

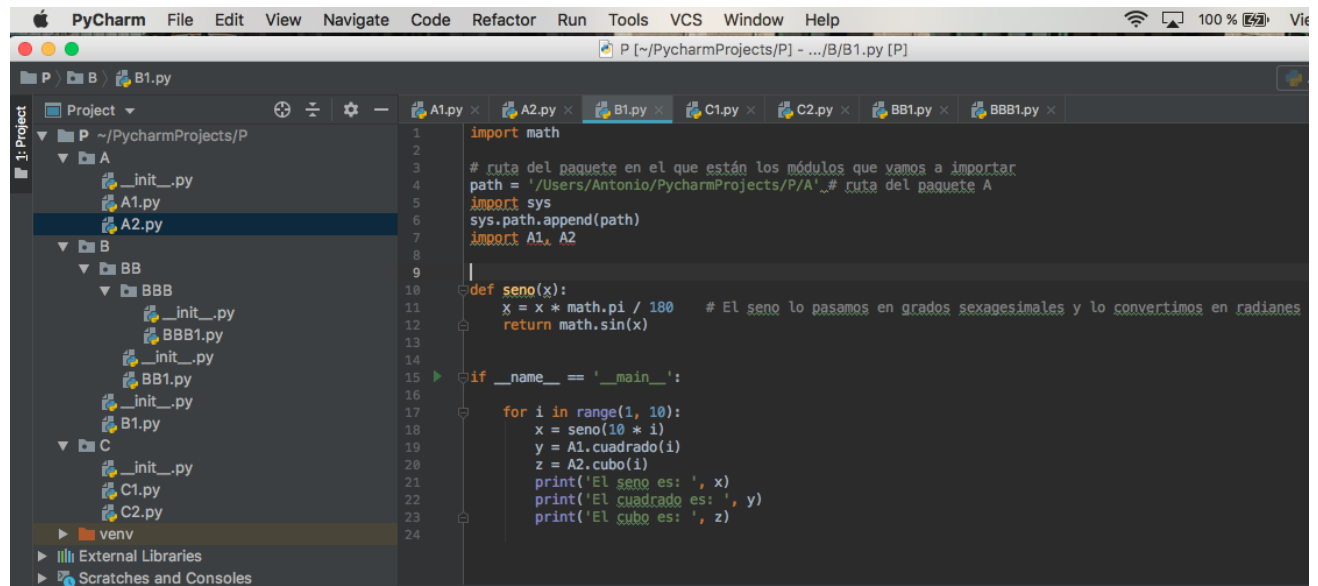
Cuadrados: 4

Cubos: 8

Cuadrados: 9

Cubos: 27

Vamos ahora al módulo B1 del paquete B. Vamos a importar los módulos A1 y A2 ya que vamos a ejecutar desde aquí las funciones que hemos declarado en ellos.



```
1 import math
2
3 # ruta del paquete en el que están los módulos que vamos a importar
4 path = '/Users/Antonio/PycharmProjects/P/A' # ruta del paquete A
5 import sys
6 sys.path.append(path)
7 import A1, A2
8
9
10 def seno(x):
11     x = x * math.pi / 180 # El seno lo pasamos en grados sexagesimales y lo convertimos en radianes
12     return math.sin(x)
13
14
15 if __name__ == '__main__':
16
17     for i in range(1, 10):
18         x = seno(10 * i)
19         y = A1.cuadrado(i)
20         z = A2.cubo(i)
21         print('El seno es: ', x)
22         print('El cuadrado es: ', y)
23         print('El cubo es: ', z)
24
```

Como los módulos A1 y A2 están en distinto paquete, no se pueden importar sin más, ya que hay que especificar la ruta del paquete A en la que Python los va a encontrar. Si se ejecuta, resulta:

El seno es: 0.17364817766693033

El cuadrado es: 1

El cubo es: 1

El seno es: 0.3420201433256687

El cuadrado es: 4

El cubo es: 8

El seno es: 0.49999999999999994

El cuadrado es: 9

El cubo es: 27

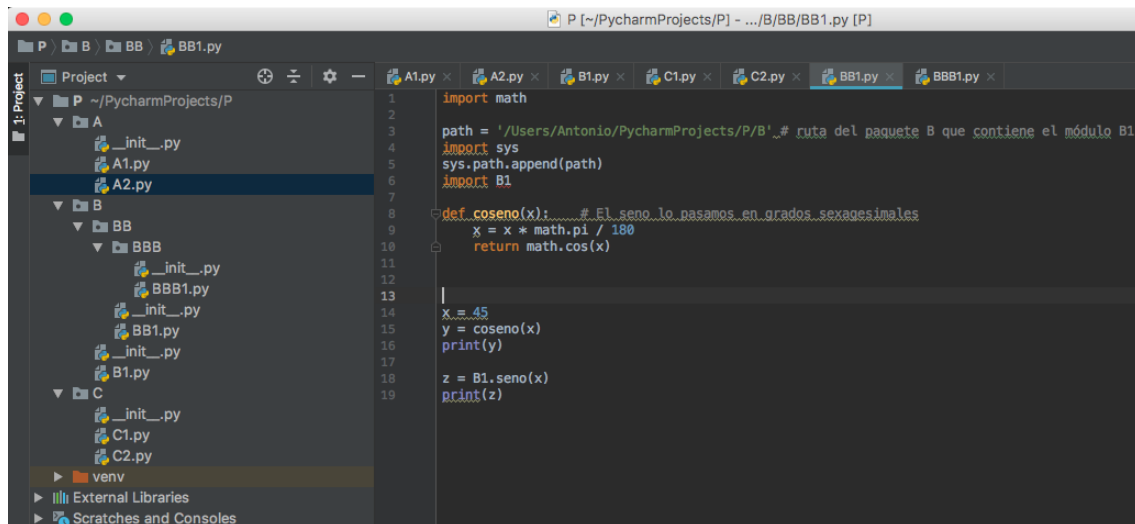
El seno es: 0.6427876096865393

El cuadrado es: 16

El cubo es: 64

...

Ahora vamos al módulo BB1 declarado en el paquete BB. En él usamos la función seno declarada en el módulo B1, por lo que hay que importarlo



```
1 import math
2
3 path = '/Users/Antonio/PycharmProjects/P/B' # ruta del paquete B que contiene el módulo B1
4 import sys
5 sys.path.append(path)
6 import B1
7
8 def coseno(x): # El seno lo pasamos en grados sexagesimales
9     x = x * math.pi / 180
10    return math.cos(x)
11
12
13
14 x = 45
15 y = coseno(x)
16 print(y)
17
18 z = B1.seno(x)
19 print(z)
```

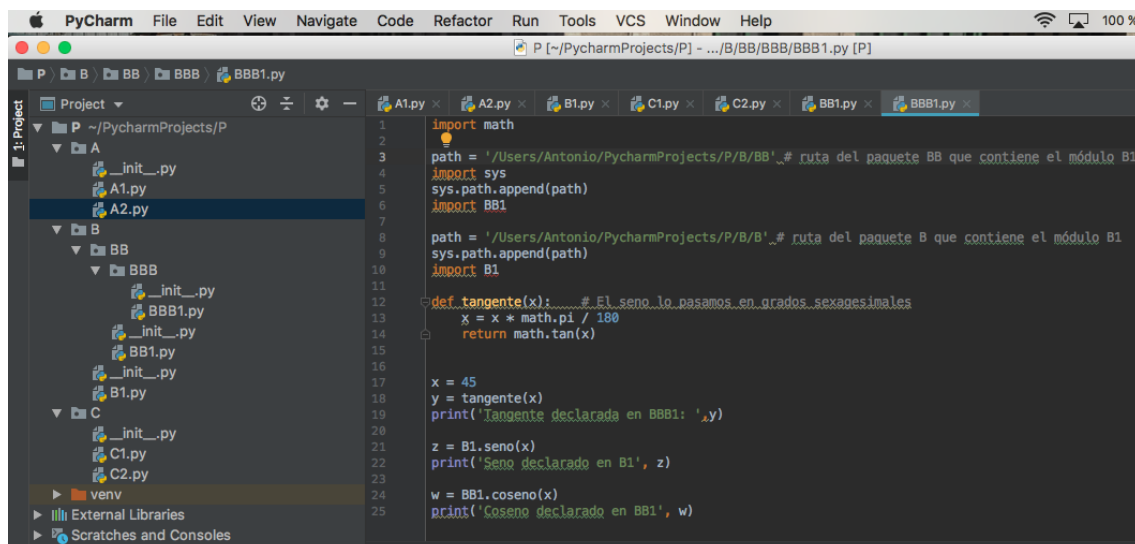
El resultado es:

0.7071067811865476

0.7071067811865475

Ya que el seno y el coseno de 45° son iguales.

Finalmente, en el módulo BBB1, del paquete BBB declaramos una función tangente e importamos los módulos B1 y BB1 ya que vamos a usar las funciones que hemos declarado en ellos. No olvidar que sólo se pueden usar aquellos módulos que están en el paquete en que estamos.



```
1 import math
2
3 path = '/Users/Antonio/PycharmProjects/P/B/BB' # ruta del paquete BB que contiene el módulo B1
4 import sys
5 sys.path.append(path)
6 import B1
7
8 path = '/Users/Antonio/PycharmProjects/P/B/BB' # ruta del paquete B que contiene el módulo B1
9 sys.path.append(path)
10 import B1
11
12 def tangente(x): # El seno lo pasamos en grados sexagesimales
13     x = x * math.pi / 180
14     return math.tan(x)
15
16
17 x = 45
18 y = tangente(x)
19 print('Tangente declarada en BBB1: ', y)
20
21 z = B1.seno(x)
22 print('Seno declarado en B1', z)
23
24 w = BB1.coseno(x)
25 print('Coseno declarado en BB1', w)
```

Si se ejecuta queda:

0.7071067811865476

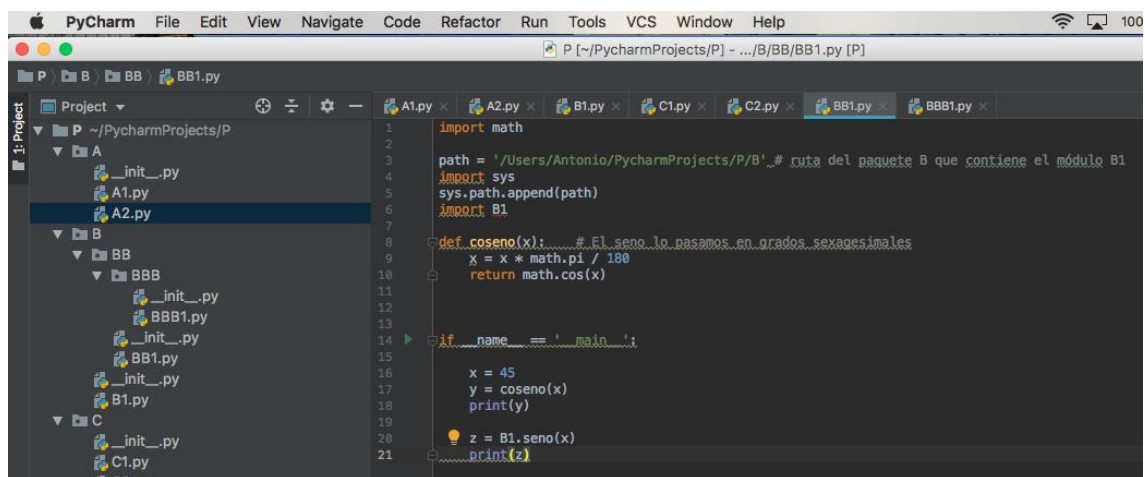
0.7071067811865475

Tangente declarada en BBB1: 0.9999999999999999

Seno declarado en B1 0.7071067811865475

Coseno declarado en BB1 0.7071067811865476

Nos puede sorprender que en primer lugar haya escrito los valores del coseno y del seno de  $45^\circ$ , antes de escribir el código que figura en el módulo BBB1. Parece como si hubiese ejecutado en primer lugar la parte ejecutable del módulo BB1 antes de ejecutar la del BBB1. Y es así. ¿Cómo se podría haber evitado y que hubiese escrito sólo lo que se le pedía en el módulo BBB1?. Muy sencillo, bastaría con añadir la línea `if __name__ == '__main__':` en el módulo BB1 como se muestra en la siguiente figura.



```
1 import math
2
3 path = '/Users/Antonio/PycharmProjects/P/B' # ruta del paquete B que contiene el módulo B1
4 import sys
5 sys.path.append(path)
6 import B1
7
8 def coseno(x): # El seno lo pasamos en grados sexagesimales
9     x = x * math.pi / 180
10    return math.cos(x)
11
12
13
14 if __name__ == '__main__':
15
16     x = 45
17     y = coseno(x)
18     print(y)
19
20     z = B1.seno(x)
21     print(z)
```

Si ahora ejecutamos el módulo BBB1, resulta:

Tangente declarada en BBB1: 0.9999999999999999

Seno declarado en B1 0.7071067811865475

Coseno declarado en BB1 0.7071067811865476