

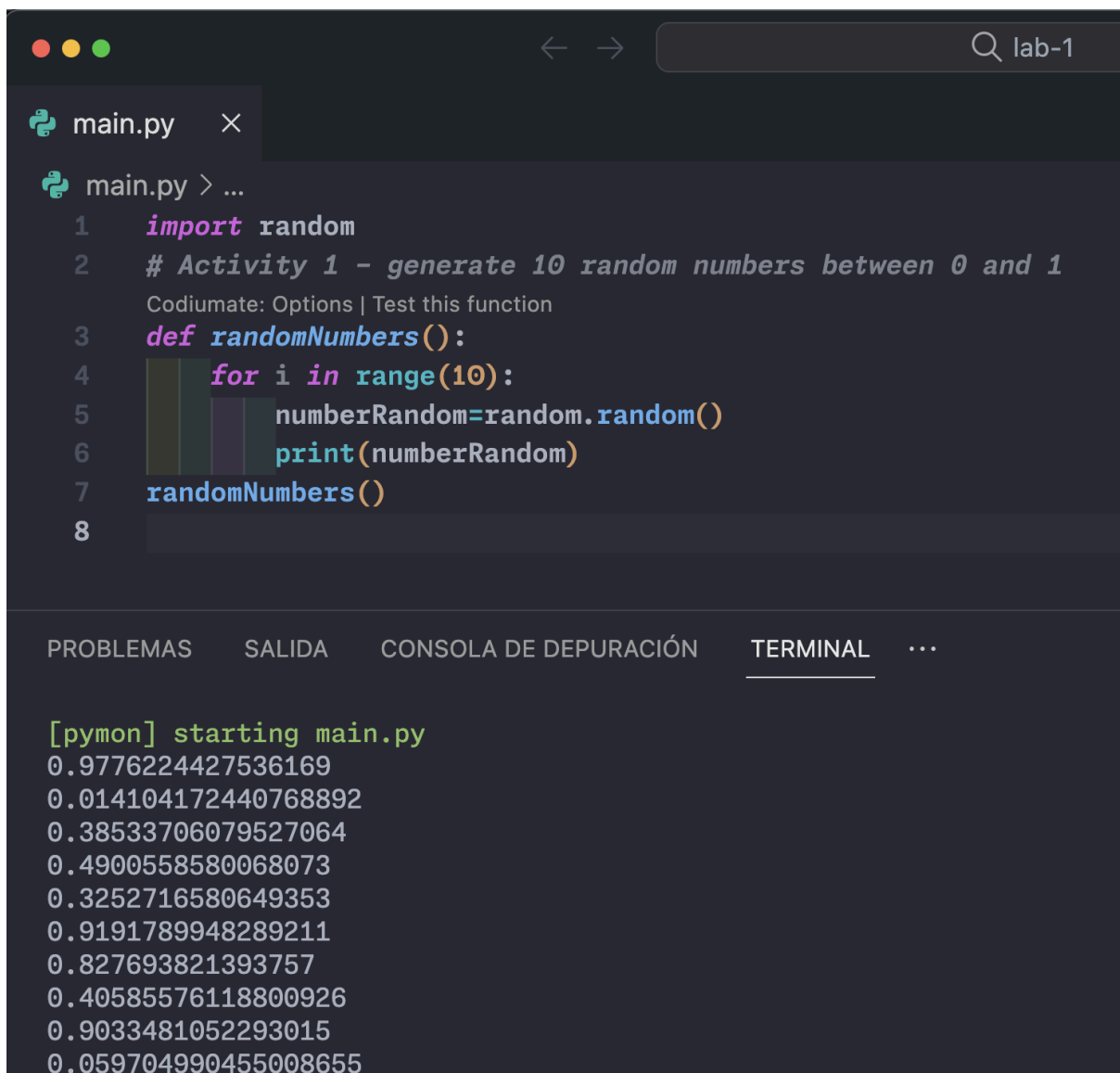
## Modelos probabilísticos

### Práctica 01

#### Generación de números pseudo-aleatorios

Nombre: Hector Paolo Barazorda cuellar  
codigo:145003

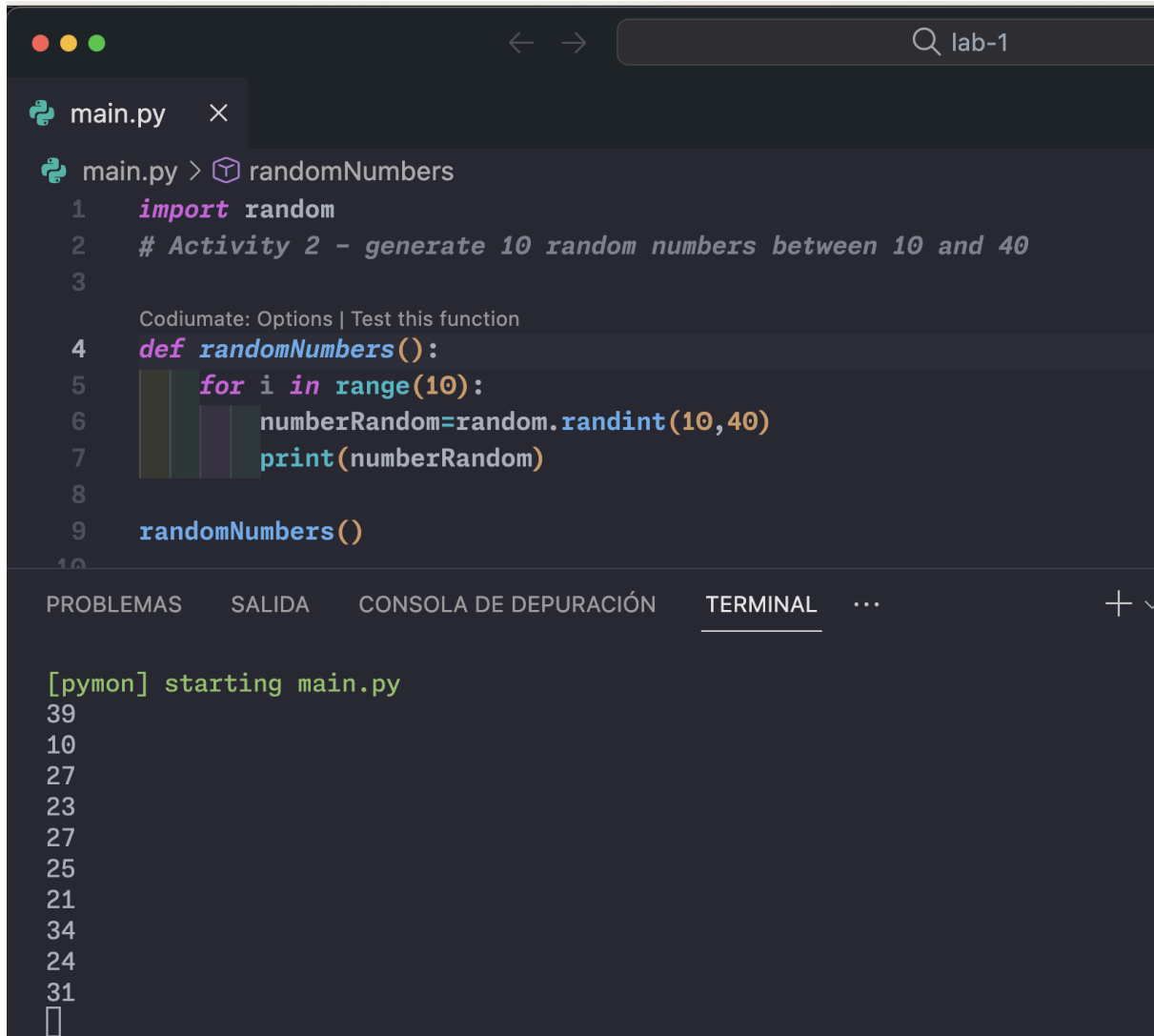
1. Generar 10 números pseudo-aleatorios entre 0 y 1 e imprimirlos. Vuelva a ejecutar y compruebe si se obtienen o no los mismos valores. Escriba sus conclusiones.



```
main.py ×  
main.py > ...  
1  import random  
2  # Activity 1 - generate 10 random numbers between 0 and 1  
   Codiumate: Options | Test this function  
3  def randomNumbers():  
4      for i in range(10):  
5          numberRandom=random.random()  
6          print(numberRandom)  
7  randomNumbers()  
8  
  
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  ...  
  
[pymon] starting main.py  
0.9776224427536169  
0.014104172440768892  
0.38533706079527064  
0.4900558580068073  
0.3252716580649353  
0.9191789948289211  
0.827693821393757  
0.40585576118800926  
0.9033481052293015  
0.059704990455008655
```

conclusiones: al volver a ejecutar dos veces seguidas se nota que; los 10 números generados son nuevos en cada ejecución.

2. Generar 10 números enteros entre 10 y 40 e imprimirlos. Vuelva a ejecutar y compruebe si se obtienen o no los mismos valores. Escriba sus conclusiones.



The screenshot shows a code editor window with a file named `main.py`. The code defines a function `randomNumbers()` that generates 10 random integers between 10 and 40 and prints them. The terminal output shows the execution of the script, displaying 10 random numbers: 39, 10, 27, 23, 27, 25, 21, 34, 24, and 31.

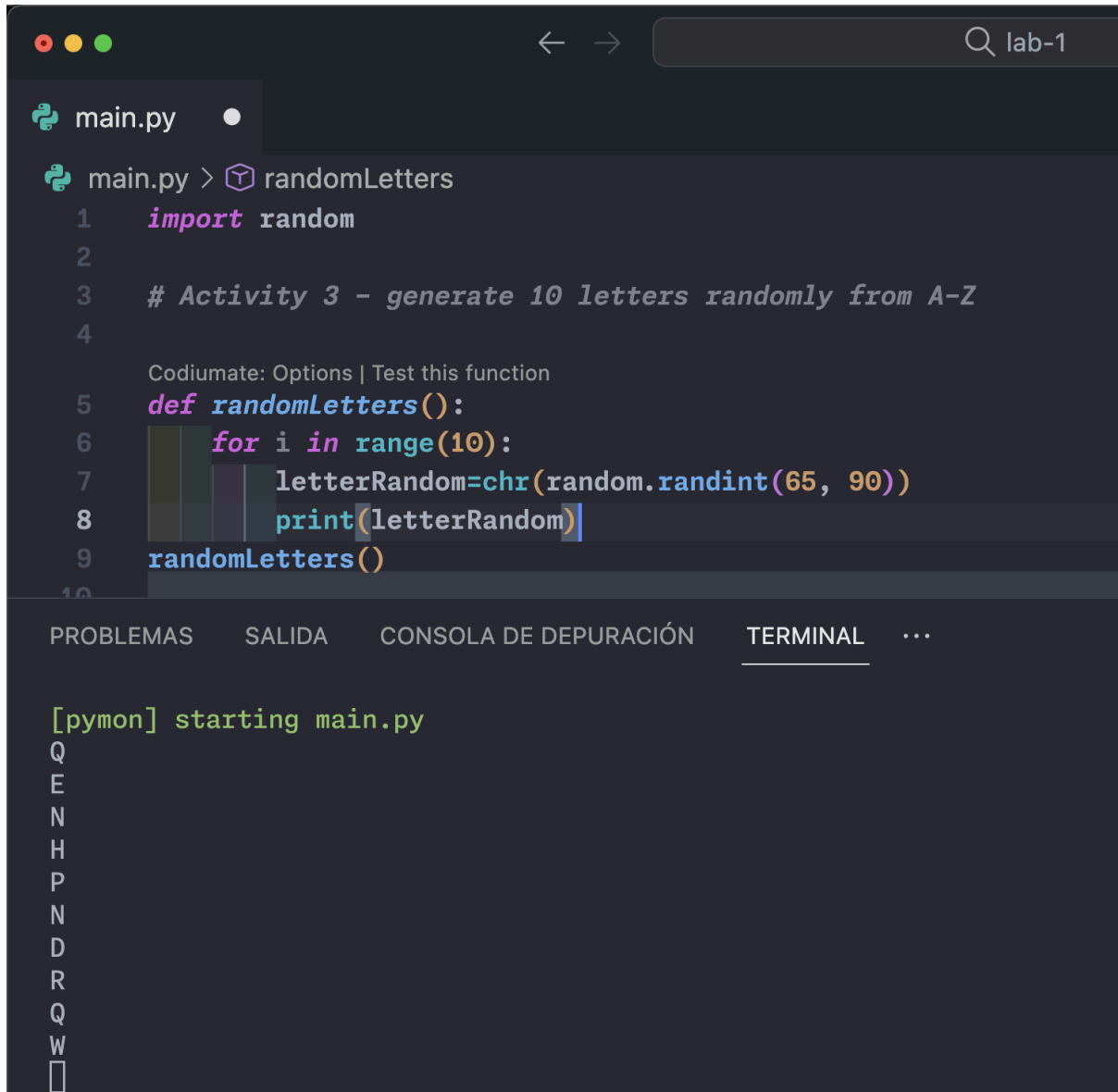
```
main.py > randomNumbers
1  import random
2  # Activity 2 - generate 10 random numbers between 10 and 40
3
4  def randomNumbers():
5      for i in range(10):
6          numberRandom=random.randint(10,40)
7          print(numberRandom)
8
9  randomNumbers()
10
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL ...

```
[pymon] starting main.py
39
10
27
23
27
25
21
34
24
31
█
```

conclusiones: Se observa igual que en la anterior pregunta que los nuevos numeros generados son diferentes en cada ejecucion.

3. Obtener 10 letras de forma aleatoria de la cadena 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' e imprimirlos. Vuelva a ejecutar y compruebe si se obtienen o no los mismos valores. Escriba sus conclusiones.



The screenshot shows a code editor with a file named `main.py`. The code defines a function `randomLetters` that generates 10 random letters from the alphabet. The terminal output shows the result of running the function, displaying 10 random letters: Q, E, N, H, P, N, D, R, Q, W.

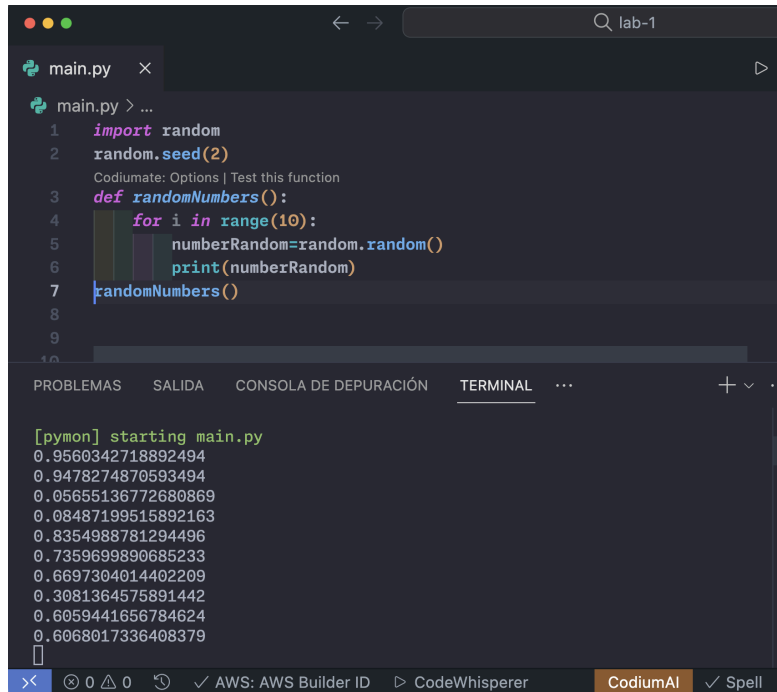
```
main.py > randomLetters
1  import random
2
3  # Activity 3 - generate 10 letters randomly from A-Z
4
5  def randomLetters():
6      for i in range(10):
7          letterRandom=chr(random.randint(65, 90))
8          print(letterRandom)
9  randomLetters()
10
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL ...

[pymon] starting main.py  
Q  
E  
N  
H  
P  
N  
D  
R  
Q  
W  
□

conclusiones : observamos que obtenemos las diez letras aleatorias, y que cada vez que volvemos a ejecutar el resultado es diferente; cada ejecucion otras 10 letras en orden aleatorio

4. Defina una semilla aleatoria y resuelva los problemas 3.1, 3.2 y 3.3. Compruebe si se obtienen o no los mismos valores en 3 ejecuciones seguidas. Además, compare los valores obtenidos con los obtenidos por su colega de a lado. Escriba sus conclusiones.



The screenshot shows a code editor with a file named `main.py`. The code defines a function `randomNumbers()` that generates 10 random numbers using `random.random()`. The terminal output shows the results of running the script, which are 10 decimal values.

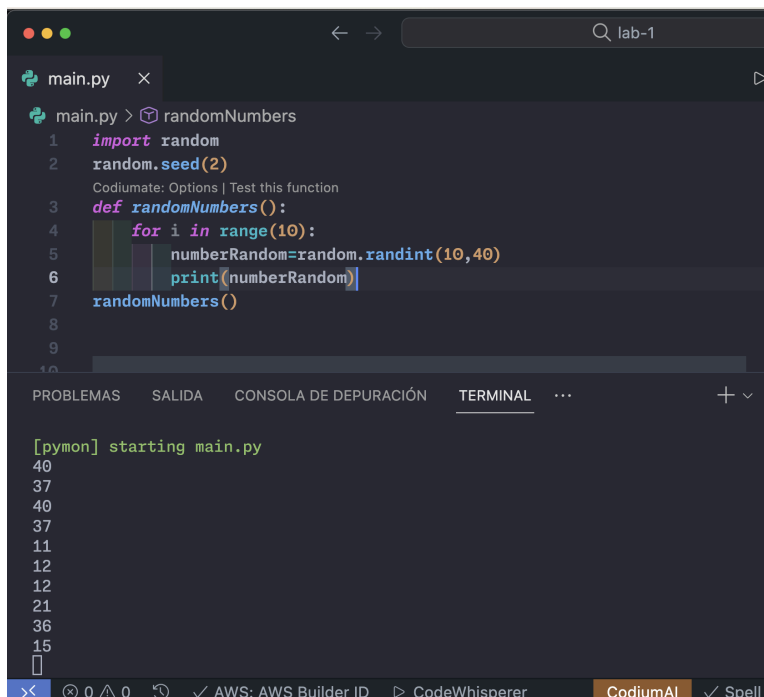
```
main.py > ...
1 import random
2 random.seed(2)
3
4 def randomNumbers():
5     for i in range(10):
6         numberRandom=random.random()
7         print(numberRandom)
8 randomNumbers()
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL ...

```
[pymon] starting main.py
0.9560342718892494
0.9478274870593494
0.05655136772680869
0.08487199515892163
0.8354988781294496
0.7359699890685233
0.6697304014402209
0.3081364575891442
0.6059441656784624
0.6068017336408379
```

>< 0 0 0 0 ✓ AWS: AWS Builder ID ▷ CodeWhisperer CodiumAI ✓ Spell

3.1



The screenshot shows a code editor with a file named `main.py`. The code defines a function `randomNumbers()` that generates 10 random numbers using `random.randint(10,40)`. The terminal output shows the results of running the script, which are 10 integer values.

```
main.py > randomNumbers
1 import random
2 random.seed(2)
3
4 def randomNumbers():
5     for i in range(10):
6         numberRandom=random.randint(10,40)
7         print(numberRandom)
8 randomNumbers()
```

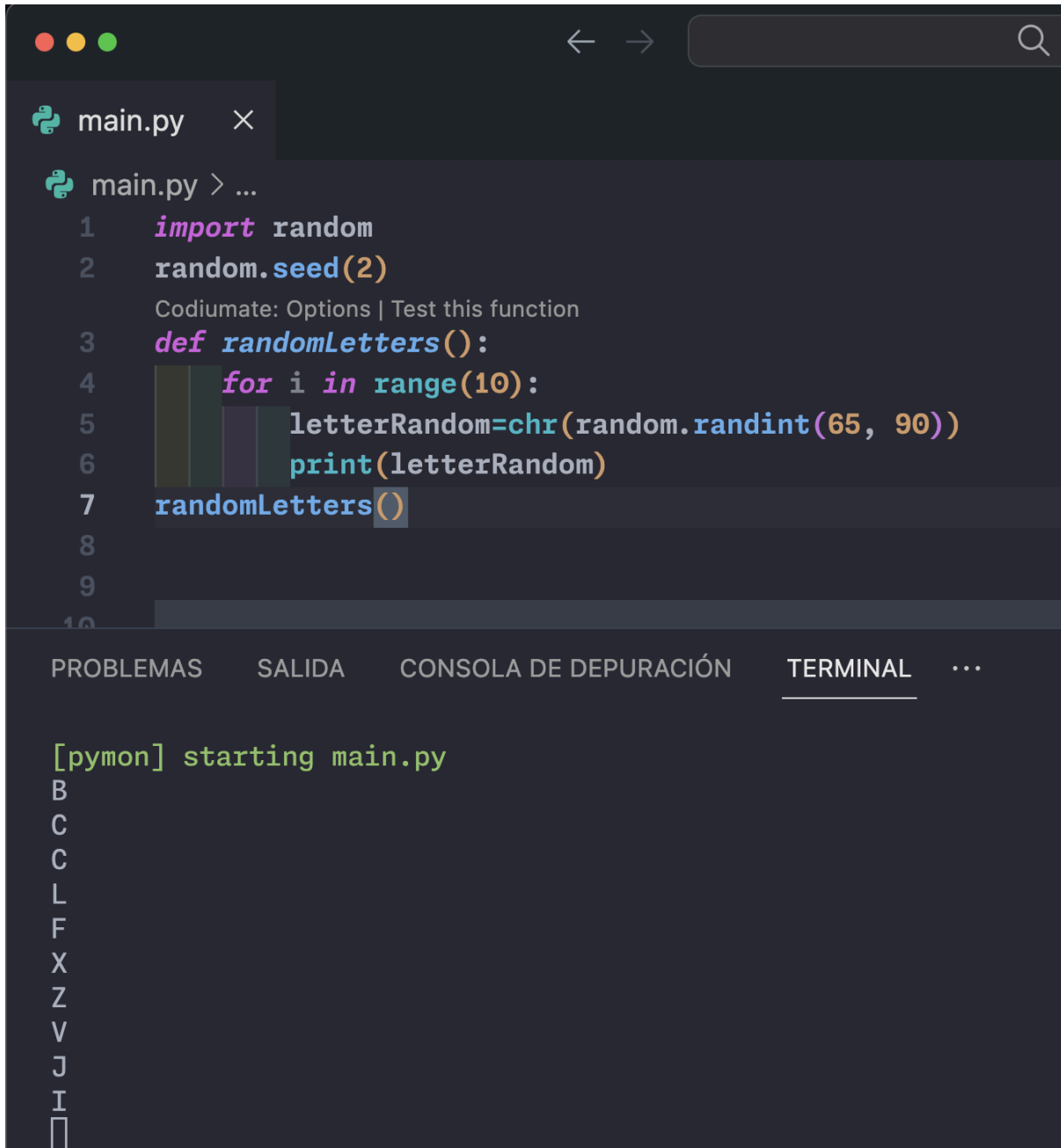
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL ...

```
[pymon] starting main.py
40
37
40
37
11
12
12
21
36
15
```

>< 0 0 0 0 ✓ AWS: AWS Builder ID ▷ CodeWhisperer CodiumAI ✓ Spell

3.2

### 3.3



The image shows a code editor window with a dark theme. The top bar has three colored window control buttons (red, yellow, green) on the left, navigation arrows in the center, and a search icon on the right. Below the top bar, there's a tab labeled 'main.py' with a close button. The main editor area shows the following Python code:

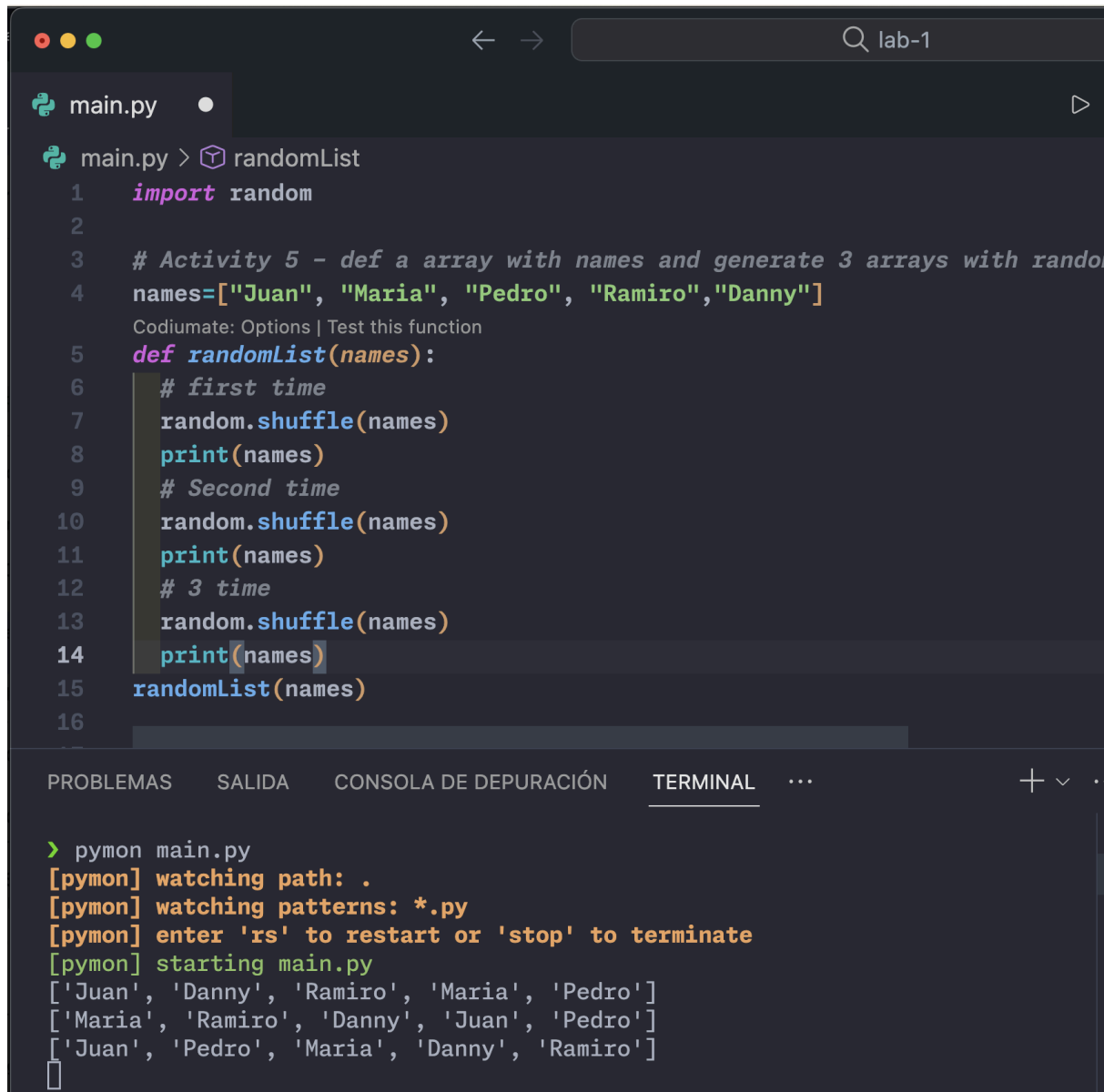
```
1 import random
2 random.seed(2)
3 def randomLetters():
4     for i in range(10):
5         letterRandom=chr(random.randint(65, 90))
6         print(letterRandom)
7 randomLetters()
```

Below the code editor, there's a panel with tabs: 'PROBLEMAS', 'SALIDA', 'CONSOLA DE DEPURACIÓN', and 'TERMINAL'. The 'TERMINAL' tab is selected and shows the output of the script:

```
[pymon] starting main.py
B
C
C
L
F
X
Z
V
J
I
█
```

observación: se observa que dando una semilla específica los resultados no cambian, osea la semilla controla los resultados que obtendremos; con la misma semilla obtenemos el mismo resultado en varias ejecuciones o maquinas.

5. Defina una lista de nombres y genere 3 listas diferentes, con los mismos nombres ubicados en diferentes posiciones.



The screenshot shows a code editor window with a file named `main.py`. The code defines a list of names and a function to shuffle and print them three times. The terminal output shows the execution of the script, displaying three different shuffled lists of the names.

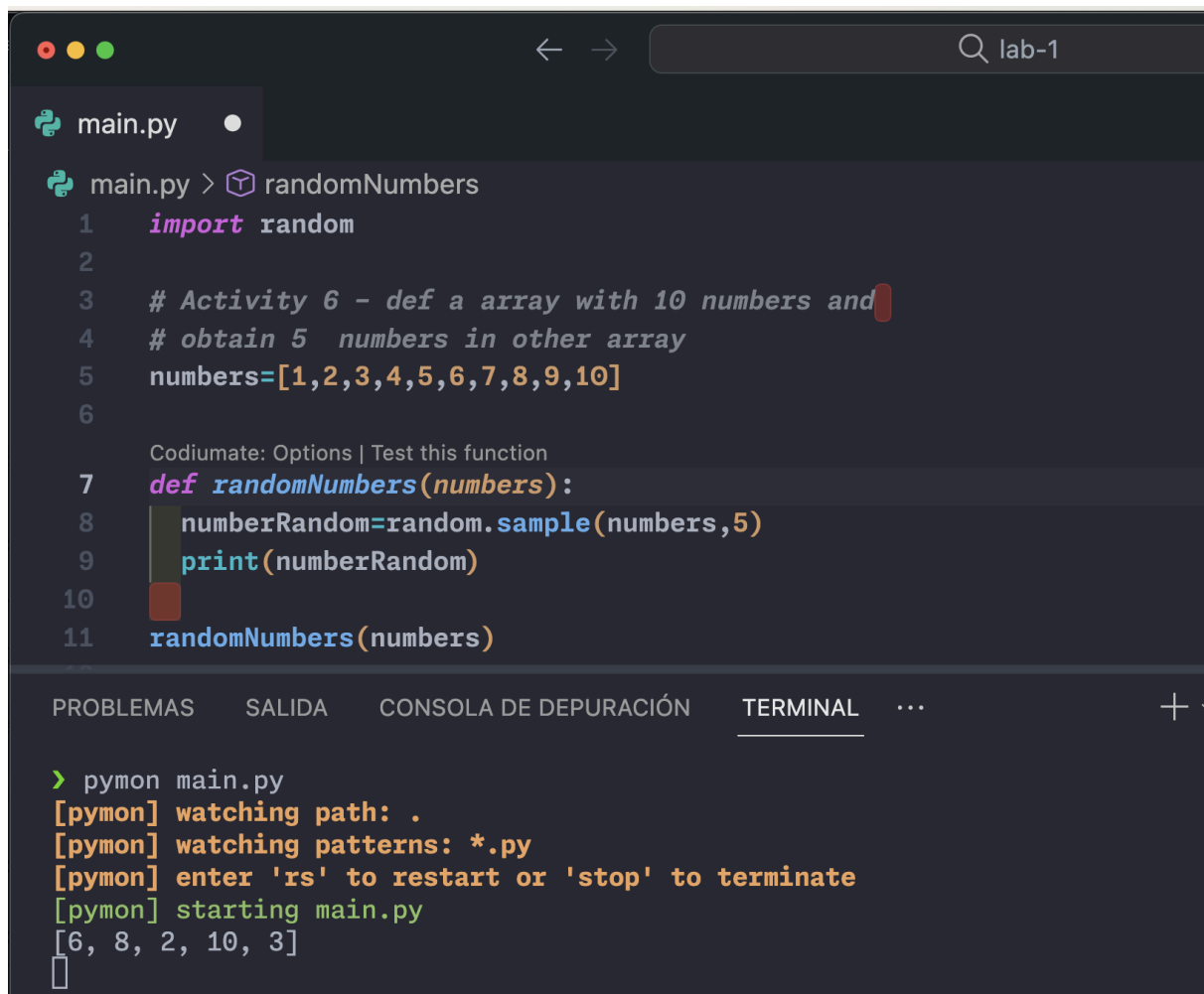
```
main.py > randomList
1  import random
2
3  # Activity 5 - def a array with names and generate 3 arrays with random
4  names=["Juan", "Maria", "Pedro", "Ramiro","Danny"]
   Codiumate: Options | Test this function
5  def randomList(names):
6      # first time
7      random.shuffle(names)
8      print(names)
9      # Second time
10     random.shuffle(names)
11     print(names)
12     # 3 time
13     random.shuffle(names)
14     print(names)
15     randomList(names)
16
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL ...

```
> pymon main.py
[pymon] watching path: .
[pymon] watching patterns: *.py
[pymon] enter 'rs' to restart or 'stop' to terminate
[pymon] starting main.py
['Juan', 'Danny', 'Ramiro', 'Maria', 'Pedro']
['Maria', 'Ramiro', 'Danny', 'Juan', 'Pedro']
['Juan', 'Pedro', 'Maria', 'Danny', 'Ramiro']
```

observaciones: se observa que obtenemos listas desordenadas, y podría suponer que si le definimos la semilla este no cambiaría en cada lista

6. Defina una lista con 10 números y genera otra lista con 5 números obtenidos de forma aleatoria de la primera lista.



The screenshot shows a code editor with a file named `main.py`. The code defines a list `numbers` with 10 elements and a function `randomNumbers` that uses `random.sample` to select 5 random elements from the list. The terminal output shows the script being executed with `pymon`, displaying the watched path, patterns, and the resulting list of 5 random numbers: `[6, 8, 2, 10, 3]`.

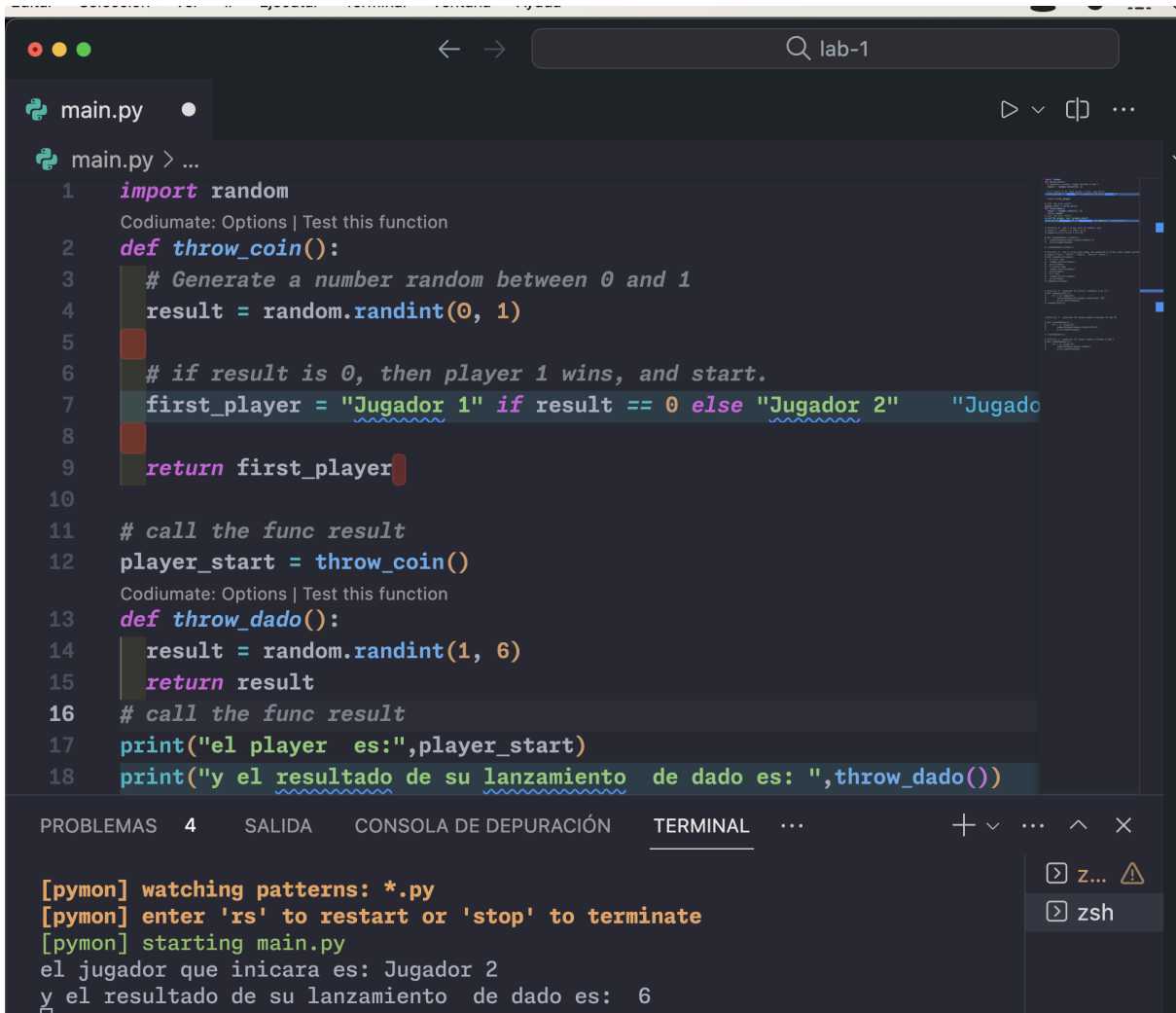
```
main.py > randomNumbers
1  import random
2
3  # Activity 6 - def a array with 10 numbers and
4  # obtain 5 numbers in other array
5  numbers=[1,2,3,4,5,6,7,8,9,10]
6
7  def randomNumbers(numbers):
8      numberRandom=random.sample(numbers,5)
9      print(numberRandom)
10
11  randomNumbers(numbers)
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL ...

```
> pymon main.py
[pymon] watching path: .
[pymon] watching patterns: *.py
[pymon] enter 'rs' to restart or 'stop' to terminate
[pymon] starting main.py
[6, 8, 2, 10, 3]
```

observación: se observa que al usar el método **simple** para obtener números aleatorios de la lista, pero si definimos una semilla este no varía.

7. Codifique una función para simular el lanzamiento de una moneda, lo cual, determinará quien iniciará el lanzamiento de dados entre 2. Muestre el jugador que inició el juego y el resultado de lanzar el dado.



```
main.py
main.py > ...
1  import random
   Codiumate: Options | Test this function
2  def throw_coin():
3      # Generate a number random between 0 and 1
4      result = random.randint(0, 1)
5
6      # if result is 0, then player 1 wins, and start.
7      first_player = "Jugador 1" if result == 0 else "Jugador 2"
8
9      return first_player
10
11 # call the func result
12 player_start = throw_coin()
   Codiumate: Options | Test this function
13 def throw_dado():
14     result = random.randint(1, 6)
15     return result
16 # call the func result
17 print("el player es:",player_start)
18 print("y el resultado de su lanzamiento de dado es: ",throw_dado())

PROBLEMAS 4 SALIDA CONSOLA DE DEPURACIÓN TERMINAL ...
[pymon] watching patterns: *.py
[pymon] enter 'rs' to restart or 'stop' to terminate
[pymon] starting main.py
el jugador que inicara es: Jugador 2
y el resultado de su lanzamiento de dado es: 6
```

Definimos primeramente un función para obtener quien iniciara el juego, luego ejecutamos otra función donde obtenemos un número aleatorio del 1-6 y así obtenemos quien jugará primero y cuál será el resultado de su primer lanzamiento.