

**Índice**

Ejecución y resultados de la solución



**Introducción**

* Ejecutar la solución.
* Reflexión sobre los resultados.



# Ejecución y resultados de la solución



## Diseño de la solución

|  |
| --- |
| **Persona** |
| nombre |
|  |

##### tiene

cocinar(solicitud)

energia habilidad

**Cocinero**

nombre platos cocineros repartidores calificacion

**Restaurant**

***tiene***

**Cliente**

**Repartidor**

recibir\_pedido

##### tiene

recibir\_pedido(pedido, demora)

exigencia distancia platos\_preferidos

repartir(pedido, distancia)

energia velocidad

nombre

calidad dificultad

**Plato**

***prepara***

**Comestible**

**Bebestible**

cubiertos

tamaño



Archivo platos.py

class Plato:

...

class Bebestible(Plato):

...

class Comestible(Plato):

...

## Módulos

Archivo personas.py

class Persona:

...

class Cliente(Persona):

...

class Cocinero(Persona):

...

... # Toma decisiones en base a tipos de plato

Archivo restaurante.py

class Restaurante:

...



**Archivo personas.py**

def crear\_cocineros(): cocineros = [] for \_ in range(5):

cocineros.append(Cocinero(choice(NOMBRES), randint(1, 10)))

return cocineros

def crear\_repartidores(): repartidores = [] for \_ in range(2):

repartidores.append(Repartidor(choice(NOMBRES), randint(1, 10)))

return repartidores

def crear\_clientes(): clientes = []

for \_ in range(20):

platos = sample(PLATOS.keys(), randint(1, 5))

cliente = Cliente( choice(NOMBRES), randint(1, 10), randint(20, 80),platos) clientes.append(cliente)

return clientes

def crear\_restaurante(): cocineros = crear\_cocineros()

repartidores = crear\_repartidores()

return Restaurante("OverEats", PLATOS, cocineros, repartidores)

restaurante = crear\_restaurante() clientes = crear\_clientes() restaurante.recibir\_pedidos(clientes)

print( f"La calificación final del restaurante {restaurante.nombre} " f"es

{restaurante.calificacion}" )

#### Podemos:

* El archivo main.py nos permite probar nuestra implementación.
* Contiene código que crea clientes, crea cocineros y repartidores, y los usa para crear un objeto de clase Restaurant.
* El código entregado utiliza una parámetro inicial (semilla, seed) común para las funciones aleatorias. Esto permite que el resultado sea siempre el mismo.
* Para efectos de prueba podemos eliminar esa línea.



## Resultado

* El archivo main.py nos permite probar nuestra implementación.

La calificación final del restaurante OverEats es 5.9875

* Contiene código que crea clientes, crea cocineros y repartidores, y los usa para crear un objeto de clase Restaurant.
* El código entregado utiliza una parámetro inicial (semilla, seed) común para las funciones aleatorias. Esto permite que el resultado sea siempre el mismo.
* Para efectos de prueba podemos eliminar esa línea.



## Síntesis

### Revisar el diseño de la solución.

* Analizar el código que permite probar la solución completa.
* Hemos construido una aplicación compleja utilizando clases y estructuras de datos, y permitiendo una interacción apropiada entre ellas.

## Referencias bibliográficas

### Documentación Python 3.8. https://docs.python.org/3/reference