Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2024-2

Examen

05 de Diciembre 2024

Instrucciones

- La evaluación consta de 31 preguntas de alternativas (30 de contenidos y 1 bonus sobre Tópicos Avanzados 2). Para obtener el 7.0 en la evaluación, se deben tener 29 preguntas correctas.
- Recibirás una hoja de respuestas que deberás rellenar con tus datos y las respuestas correspondientes a cada alternativa Sólo se corregirá la hoja de respuestas. **Ten mucha precaución de anotar correctamente tus datos**.
- Cada pregunta de selección múltiple tiene únicamente 1 alternativa correcta. Responder con 2 o más alternativas implicará dejar inválida esa pregunta y se considerará incorrecta. Además, cada pregunta presenta el mismo puntaje y no se descontará por respuesta incorrecta.
- Solo podrás retirarte de la sala una vez hayan transcurrido 30 minutos desde que inició la evaluación.
- Durante la evaluación se realizarán 2 rondas de preguntas. Estas preguntas únicamente serán respondidas por los profesores del ramo.
- En caso de que sea necesario, podrás solicitar hojas blancas para apoyar al desarrollo de la prueba. Para esto levanta la mano y espera que un ayudante se acerque a tu puesto.
- Para quienes justificaron su ausencia en el *midterm* con su Unidad Académica y fue aceptada su justificación, deben tener las primeras 30 preguntas correctas de esta evaluación para lograr el 7.0 equivalente solo al apartado de alternativas del *midterm* (50 % de la nota final).

Selección múltiple

1. Dado el siguiente código en Python, ¿cuál es el correcto orden de salida resultante de los llamados a `print` una vez finalizada la ejecución del programa? Para esta pregunta, considera que en vez de saltos de línea usaremos una coma por temas de espacio.

```python
import threading, time

def f_thread_1(lock):
    lock.acquire()
    print("1")

def f_thread_2(evento, lock):
    evento.wait()
    lock.acquire()
    print("2")

def f_thread_3(evento, lock):
    evento.set()
    time.sleep(111369)  # Dormir 111.369 segundos (casi 31 horas)
    print("3")
    lock.release()

evento = threading.Event()
lock = threading.Lock()

thread_1 = threading.Thread(target=f_thread_1, args=(lock, ), daemon=True)
thread_2 = threading.Thread(target=f_thread_2, args=(evento, lock), daemon=True)
thread_3 = threading.Thread(target=f_thread_3, args=(evento, lock), daemon=True)

thread_2.start()
thread_1.start()
thread_1.join()
thread_3.start()
thread_3.join(timeout=1)
time.sleep(2)
print("4")
```

A) `1, 4`

B) `1, 4, 3, 2`

C) `1, 3, 2, 4`

D) `2, 1, 3, 4`

E) `1, 2, 4`

2. Dada una clase que hereda exclusivamente de `Thread`, ¿cuál es la **diferencia** entre el método `run` y el método `start`?

   A) Al método `run` se le debe hacer *overriding*. En cambio, el método `start` es para iniciar, de forma concurrente, el *thread*.

   B) Al método `start` se le debe hacer *overriding* y dentro de este mismo método se debe llamar al método `run` para inicializar, de forma concurrente, el *thread*.

   C) El método `start` inicializa el `thread` como *daemon* mientras que el método `run` lo inicializa como no *daemon*.

   D) El método `run` no existe, solo se utiliza `start`.

   E) No tienen diferencia, ambos hacen que se ejecute el `Thread` de la misma forma.

3. Un cliente te solicita diseñar una simulación de las olimpiadas, en particular, la carrera de 100 metros planos, donde cada atleta se debe ejecutar en paralelo y empezar a correr al momento de escuchar el sonido de partida. Para esto, te solicita confeccionar un programa principal que representará a la carrera y utilizar *threads* para representar a cada atleta. Bajo esta solicitud, la primera idea que surgió fue diseñar la simulación con Eventos, pero el cliente no está convencido de dicha decisión.

   Dado este problema, ¿cuál de las siguientes afirmaciones es **correcta** para justificar el uso de Eventos en este contexto?

   A) Garantiza que todos los atletas simulen la carrera de forma concurrente.

   B) Permite que todos los atletas empiecen al mismo tiempo en función de lo determinado por otra entidad.

   C) Permite que todos los atletas acaben su ejecución abruptamente si es que el programa principal finaliza su ejecución.

   D) Es un mecanismo que garantiza, en todo momento, que no ocurra un *deadlock* entre los atletas.

   E) Es un mecanismo para garantizar que solo un atleta pueda avisar que llegó a la meta.

4. ¿Cuál de las siguientes afirmaciones sobre la clase `Lock` de `threading` es **correcta**?

   A) Es utilizado para limitar el acceso a recursos compartidos a un solo *thread* a la vez.

   B) Permite que varios *threads* accedan a una sección crítica simultáneamente.

   C) Siempre bloquea un *thread* deteniendo permanentemente su ejecución.

   D) Se utiliza para detener un *thread* una vez que el *main thread* haya terminado.

   E) Es utilizado principalmente para finalizar *threads* de forma segura.

5. En el siguiente bloque de código, el programa se detiene debido a un *Deadlock*. ¿Cuál sería la forma más adecuada de solucionar este problema?

```
1  import threading
2  from time import sleep
3
4  lock_a = threading.Lock()
5  lock_b = threading.Lock()
6
7  def thread1():
8      with lock_a:
9          print("Thread 1: Adquiere lock_a")
10         # Simula alguna operación
11         time.sleep(100)
12         print("Thread 1: Intentando adquirir lock_b")
13         with lock_b:
14             print("Thread 1: Adquiere lock_b")
15
16 def thread2():
17     with lock_b:
18         print("Thread 2: Adquiere lock_b")
19         # Simula alguna operación
20         time.sleep(100)
21         print("Thread 2: Intentando adquirir lock_a")
22         with lock_a:
23             print("Thread 2: Adquiere lock_a")
24
25 t1 = threading.Thread(target=thread_1)
26 t2 = threading.Thread(target=thread_2)
27 t1.start()
28 t2.start()
29 t1.join()
30 t2.join()
```

A) Modificar la sangría/indentación de las líneas 13-14 y 22-23 para que esos bloques de código no estén dentro del bloque `with` anterior.

B) Implementar un objeto `Event` para sincronizar el momento en que comienzan ambos *threads*.

C) Cambiar las líneas 13 y 17 por `with lock_a` para usar siempre el mismo *lock*.

D) Intercambiar las líneas 28 con 27, con el objetivo de comenzar primero la ejecución de `t2` y luego `t1`.

E) Ninguna de las alternativas anteriores es una solución adecuada para resolver el *deadlock*.

6. En el siguiente *script* de Python, un cliente busca conectarse a un servidor y obtener información de este. El servidor, por su parte, maneja múltiples clientes de manera concurrente utilizando *threads* desde que acepta la conexión hasta que la cierra.

```
1  import socket
2
3  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4  sock.connect(("localhost", 8080))
5
6  data1 = sock.recv(256)
7  data2 = sock.recv(256)
8
9  data = data1 + data2
10 print(data.decode("utf-8"))
```

Si el servidor envía datos al `socket` de cada nuevo cliente que se conecta, ¿cuál de las siguientes afirmaciones es **correcta** respecto al manejo de múltiples clientes en este contexto?

A) Si dos clientes se conectan simultáneamente, uno de ellos no recibirá datos y el programa lanzará una excepción en la línea 7 del cliente.

B) El servidor siempre debe garantizar que cada cliente reciba 512 *bytes* de datos antes de continuar con otro cliente.

C) Si uno de los clientes se desconecta abruptamente, el servidor deberá manejar la excepción para evitar un fallo global del sistema.

D) Si el servidor envía un mensaje de menos de 256 *bytes*, el cliente esperará en el primer `recv` a que lleguen los *bytes* restantes.

E) El uso de *threads* por parte del servidor asegura que cada cliente reciba los datos del servidor simultáneamente, sin retrasos ni datos corruptos.

7. Respecto a un cliente TCP, ¿cuál o cuáles de las siguientes afirmaciones son **correctas**?

I. El cliente no necesita usar el método `bind`, ya que no necesita abrir un puerto para comunicarse.

II. Es posible conectarse a un servidor sin necesariamente conocer la dirección IP del servidor.

III. Un cliente solo recibirá mensajes si es que envió una solicitud con `send` o `sendall` previamente.

A) Solo I

B) Solo II

C) Solo III

D) Solo I, II, III

E) Ninguna de las afirmaciones son correctas.

8. Según lo visto en el curso, ¿cuál de las siguientes afirmaciones es **incorrecta** respecto a la serialización de datos para enviarlos a través de *sockets* en Python?

   A) El módulo `pickle` puede serializar objetos de Python a *bytes*, los cuales pueden ser enviados a través de *sockets*.

   B) Es necesario codificar a *bytes* las cadenas de texto que fueron serializadas con `json`, antes de enviarlas.

   C) Los datos serializados usando `pickle` y enviados a través de *sockets* deben ser deserializados en el receptor usando `pickle`.

   D) El módulo `json` puede serializar cualquier objeto de Python, incluyendo instancias de clases y funciones, permitiendo su reconstrucción exacta en el receptor.

   E) Para enviar datos de gran tamaño, es práctica común dividirlos en *chunks* más pequeños y enviarlos secuencialmente utilizando `send` y `recv`.

9. Se te encarga disponibilizar cierta información de forma serializada, para ser usada entre plataformas que utilizan diferentes lenguajes de programación. Debes decidir, entre las siguientes alternativas, respecto a **cuál o cuáles módulos utilizar** considerando su justificación.

   A) `json`, ya que protege los datos al no ser fácil de leer por un humano.

   B) `pickle`, ya que es fácil de leer para los humanos y lo hace más transparente.

   C) `pickle`, ya que es un formato que cualquier lenguaje puede interpretar.

   D) `json`, ya que puede ser interpretado por varios lenguajes de programación distintos.

   E) Tanto `json` como `pickle` pueden ser usados sin problemas en este caso.

10. ¿Cuál de las siguientes afirmaciones es **incorrecta** sobre el manejo de *bytes* en Python?

    A) Los objetos del tipo `bytes` son inmutables.

    B) Cada *byte* representa un valor hexadecimal, por lo que contiene 6 *bits*.

    C) La sucesión de caracteres "`\x`" indica que los dos caracteres que le siguen representarán un único *byte*.

    D) Es posible definir un objeto *bytes* válido sin usar la sucesión de caracteres "`\x`".

    E) Dos objetos *bytearray* de distinto largo pueden representar el mismo número binario.

11. A una persona se le pidió implementar una arquitectura *Peer2Peer* que permita múltiples conexiones de clientes. Respecto a la implementación, asuma que los métodos: `procesar_mensaje` , `conversar_con_servidor` están bien implementados y no generan errores. ¿Cuál de las siguientes afirmaciones es **incorrecta**?

```python
from socket import socket, AF_INET, SOCK_STREAM

class Ares:
    def __init__(self, host: int, port: dict) -> None:
        self.socket_cliente = socket(AF_INET, SOCK_STREAM)
        self.socket_servidor = socket(AF_INET, SOCK_STREAM)
        self.socket_servidor.bind((host, port))
        self.socket_servidor.listen()
        self.aceptar_conexion()
        try:
            self.conectar_a_servidor()
        except ConnectionError:
            self.socket_cliente.close()
            exit()

    def aceptar_conexion(self) -> None:
        client_socket, _ = self.socket_servidor.accept()
        self.escuchar_cliente(client_socket)

    def escuchar_cliente(self, client_socket: socket) -> None:
        while True:
            respuesta = client_socket.recv(4096)
            mensaje = respuesta.decode()
            self.procesar_mensaje(client_socket, mensaje)

    def conectar_a_servidor(self) -> None:
        self.socket_cliente.connect('192.168.0.1', ['mi puerto'])
        self.conversar_con_servidor()
```

A) Dentro del flujo normal de este código, no se cumplirían todos los roles esperados por una arquitectura *Peer2Peer*.

B) La implementación de `Ares` no acepta la conexión de múltiples clientes.

C) La configuración del protocolo TCP/IP no está bien realizada.

D) Introduciendo únicamente *threads* es posible tener un código totalmente funcional y que cumple con el rol esperado de la arquitectura *Peer2Peer*.

E) Si llega un mensaje que no se puede decodificar con `.decode()`, el código levanta un error.

12. En base al siguiente código en Python que representa un servidor y un cliente, indique cuál de las presentes afirmaciones es **correcta**:

`servidor.py`

```python
import socket

sock_a = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock_b = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock_a.bind(('localhost', 11111))
sock_b.bind(('localhost', 22222))

sock_a.listen()
sock_b.listen()

sock_cliente_a, (host_cliente_a, puerto_cliente_a) = sock_a.accept()
sock_cliente_b, (host_cliente_b, puerto_cliente_b) = sock_b.accept()

data_a = sock_cliente_a.recv(4096).decode("utf-8")
print(f"Recibí (A): {data_a}")

data_b = sock_cliente_b.recv(4096).decode("utf-8")
print(f"Recibí (B): {data_b}")
```

`cliente.py`

```python
import socket

sock_b = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock_b.connect(('localhost', 22222))

sock_a = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock_a.connect(('localhost', 11111))

sock_b.sendall('Guau'.encode('utf-8'))
sock_a.sendall('Miau'.encode('utf-8'))
```

A) Ocurre un error en la línea 7 del servidor, ya que tratamos de abrir dos *sockets* en la misma dirección IP.

B) El código se quedará estancado en la línea 12 del servidor y la línea 4 del cliente, ya que el código se bloquea al esperar que el `sock_a` acepte conexión, pero se trató de conectar al `sock_b` primero.

C) Las variables `puerto_cliente_a` y `puerto_cliente_b` tendrán el mismo valor, ya que provienen del mismo cliente.

D) El código se quedará estancado en la línea 15 del servidor y la línea 10 del cliente, ya que el código se bloquea al esperar que un mensaje llegue al `sock_a`, pero el primer mensaje se envía al `sock_b`.

E) Tanto el código del servidor como del cliente se ejecutan sin levantar alguna excepción.

13. ¿Cuál de las siguientes siguientes afirmaciones sobre el uso de APIs es **correcta**?

    A) Los códigos de respuesta siempre se envían en el *body* de la respuesta de la API al cliente.

    B) En cualquier *endpoint* que reciba parámetros, estos siempre deben enviarse en el *body* de la solicitud.

    C) En un *endpoint* de una API cualquiera, el método `POST` siempre se usará para crear un recurso nuevo en el servidor.

    D) Todas las APIs usan el formato JSON en sus respuestas a solicitudes de clientes.

    E) Ninguna de las afirmaciones anteriores es correcta.

14. Según lo visto en el curso sobre redes, ¿cuál de las siguientes distribuciones servidor-cliente **NO** es posible de realizar?

    A) Un cliente conectado a dos servidores distintos.

    B) Un cliente conectado al servidor de otro cliente.

    C) Un cliente conectado al servidor de otro cliente, y ambos conectados a un mismo servidor.

    D) Un cliente conectado a los respectivos servidores de múltiples clientes.

    E) Todas las distribuciones anteriores son posibles de realizar.

15. Según las siguientes definiciones asociadas a los principios de diseño de *software* de calidad:

    Concepto 1 Cada uno de los componentes del *software* debe realizar solo las tareas para las que fue creado.

    Concepto 2 La modificación de un componente, implica que es necesario modificar otro componente para que la implementación del cambio sea correcta y completa.

    Indique qué debemos buscar al momento de diseñar un *software* de calidad:

    A) Evitar el **Concepto 1** y asegurar el **Concepto 2**.

    B) Asegurar el **Concepto 1** y evitar el **Concepto 2**.

    C) Asegurar el **Concepto 1** y el **Concepto 2**.

    D) Solo es necesario asegurar el **Concepto 1**.

    E) Solo es necesario asegurar el **Concepto 2**.

16. ¿Cuál de las siguientes afirmaciones sobre señales es **correcta**?

    A) Siempre deben enviar uno o más valores a través de ellas.

    B) Solo se pueden conectar a una única función objetivo.

    C) Solo pueden ser declaradas dentro de una clase de PyQt.

    D) Una señal se puede ejecutar directamente de la misma forma que una función o método.

    E) Todas las alternativas anteriores son incorrectas.

17. En el contexto de Interfaces Gráficas, ¿cuál de las siguientes alternativas **NO** corresponde a uno de los elementos fundamentales del modelo basado en eventos?

    A) El objeto evento.

    B) El momento de emisión del evento.

    C) El emisor del evento.

    D) El receptor del evento.

    E) Todos los elementos anteriores son fundamentales.

18. Acabas de encontrar un juego programado en Python que se juega en solitario. Está programado con un patrón *backend-frontend*, correctamente modelado para asegurar la cohesión y acoplamiento esperado. Ahora, se desea trasladar toda la lógica del juego a un Servidor y agregar un nuevo chat para que los jugadores puedan conversar entre ellos mientras juegan en solitario.

    ¿Cuál de las siguientes alternativas es **correcta** frente a este cambio?

    A) Se deberá utilizar algún módulo de concurrencia para garantizar un correcto funcionamiento del chat.

    B) Solo es necesario conocer la dirección del Servidor para que el juego funcione correctamente.

    C) El Servidor no puede ser una API. Debe ser programado con *socket* y el protocolo TCP-IP.

    D) Dado que el juego está programado en Python, el servidor también debe estar programado en Python.

    E) El *frontend* del juego no debería sufrir modificaciones.

19. ¿Cuál de los siguientes casos debe ser programado exclusivamente con `Thread` o con `QThread`, pero no con ambos?

    A) Un servidor que permite la conexión de múltiples clientes simultáneamente.

    B) Una interfaz para ver el progreso de descarga de múltiples archivos.

    C) Un juego donde hay que disparar meteoritos que caen del cielo.

    D) Una ventana que se mueve de forma aleatoria por la pantalla.

    E) Ninguna de los anteriores.

20. Suponiendo que todos los módulos necesarios están correctamente importados y no existe error de sintaxis, ¿qué **ocurre** al ejecutar el siguiente código en la consola?

```
class Backend()
    def __init__(self):
        self.componente = "ALU"

    def obtener_componentes(self):
        print(self.componente)

class PlacaFPGA(QWidget):
    def __init__(self):
        super().__init__()
        self.backend = Backend()
        self.t = Thread(target=self.conectar_cables)
        self.t.start()

    def conectar_cables(self):
        print('XX')
        self.backend.obtener_componentes()

app = QApplication([])
placa = PlacaFPGA()
placa.show()
sys.exit(app.exec_())
```

A) Saldrá un error de ejecución porque no se puede usar un `Thread` para conectarse a un método de una clase que hereda de un objeto de `PyQt`.

B) Solo se imprime `"XX"` y luego saldrá un error de ejecución por intentar ejecutar un método de otra clase sin usar señales.

C) Se va a imprimir `"XX"` y luego `"ALU"`.

D) Solo se imprime `"XX"` y luego no ocurre nada más porque estamos llamando al *backend* sin usar señales.

E) No imprime nada porque como no es un `QThread`, nunca se ejecuta `conectar_cables`.

21. ¿Cuál alternativa es **incorrecta** respecto a los algoritmos de búsqueda BFS y DFS, usados en grafos en donde todas las aristas poseen el mismo peso?

A) Es importante considerar qué vértices del grafo ya han sido visitados.

B) BFS es útil para encontrar la cantidad mínima de aristas a recorrer entre nodos.

C) DFS encontrará siempre el camino más corto.

D) BFS recorrerá el grafo por amplitud, y DFS recorrerá el grafo por profundidad.

E) DFS explora un camino completo, desde un nodo, hasta llegar a un final o a un nodo ya visitado antes de retroceder y explorar otros caminos.

22. Dado el siguiente `DataFrame` de pandas, con nombre `df`, el cual representa la matriz de adyacencia de un grafo de cuatro nodos, ¿cuál de las siguientes opciones de código devuelve el valor que indica si existe una conexión del nodo B al nodo D?

	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	1	0	0	0
D	0	1	0	0

A) `df.loc['B', 'D']`

B) `df.head(2).iloc[1, 3]`

C) `df.iloc[2, 3]`

D) `df.iloc['D', 'B']`

E) `df.loc[:, ['B', 'D']]`

23. ¿Cuáles de las siguientes filas de esta tabla demarcan **comparaciones válidas** entre listas de Python y *arrays* creados con NumPy?

		Listas de Python	**Arrays de NumPy**
I.	Son más eficientes en memoria y velocidad.	✗	✓
II.	Puede modificarse su contenido.	✓	✗
III.	Se puede acceder a sus elementos mediante índices.	✓	✓
IV.	Alta compatibilidad con operaciones de álgebra lineal y transformación de matrices.	✗	✓

A) II y III

B) I, II y III

C) I, II y IV

D) I, III y IV

E) II, III, IV

24. A partir de lo visto en el curso sobre Diagramas de Clases, ¿cuál de los siguientes elementos **NO** es necesario representar en el diagrama?

    A) Relación de herencia entre dos clases.

    B) Atributos de la clase y su tipo.

    C) Relación de contención entre dos clases.

    D) Variables definidas dentro de los métodos de la clase.

    E) Métodos de la clase, *input* y tipo de *output*.

25. Asumiendo que dispones de los siguientes dos archivos, ubicados en el mismo directorio. ¿Cuál de las siguientes afirmaciones podría justificar el uso de "`if __name__ == '__main__'`" en el archivo `suma.py`?

    `suma.py`

```python
from random import randint

def suma_mentirosa(a: int, b: int) -> int:
    return a + b + randint(1, 10)

if __name__ == '__main__':
    b = randint(1, 2)
    a = 3
    print(suma_mentirosa(b, a))
```

    `main.py`

```python
from suma import suma_mentirosa

print(suma_mentirosa(5, 8))
```

    A) Evitar que se importe `random` en `main.py`.

    B) Evitar que se importe la función `suma_mentirosa` en `main.py`.

    C) Evitar que se puedan realizar importaciones circulares entre `suma.py` y `main.py`.

    D) Evitar que se ejecute el código del bloque "`if __name__ == '__main__'`" cuando se ejecute `main.py`.

    E) No tiene efecto alguno ya que solo se importa la función `suma_mentirosa`.

26. De los siguientes comandos de `Git`, ¿cuál o cuáles interactúan **directamente** con la copia **remota (ubicada en la nube)** de un repositorio?

    I. `git pull`

    II. `git add`

    III. `git commit`

    IV. `git push`

    A) Solo IV

    B) I y IV

    C) III y IV

    D) I, III y IV

    E) I, II, III y IV

27. En *Python*, las *properties* permiten encapsular el acceso a atributos en una clase, proporcionando una interfaz controlada. ¿Cuál de las siguientes afirmaciones sobre las *properties* es **incorrecta**?

    A) Una *property* puede ser definida utilizando el decorador "`@property`", y por defecto solo permite lectura del atributo.

    B) Una forma de habilitar la edición de una *property* llamada "`precio`" es con el decorador "`@precio.setter`".

    C) Usar *properties* puede ayudar a restringir modificaciones de un atributo.

    D) Las *properties* solo funcionan para atributos privados.

    E) Usar *properties* puede ayudar a implementar validaciones al momento de modificar un atributo.

28. Considerando el conjunto de reglas de PEP8 que considera el curso, ¿cuál de las siguientes afirmaciones **NO** corresponde a una de estas reglas?

    A) Aplicar formatos específicos para los nombres de clases, funciones y variables.

    B) Importar al inicio de los archivos.

    C) Nombres de variables descriptivos.

    D) Uso de "`/`" para escribir *paths*.

    E) Líneas de largo máximo de 100 caracteres.

29. Suponiendo que en el siguiente código todos los módulos necesarios están correctamente importados y no existen errores de sintaxis. ¿Cuál de las siguientes afirmaciones es **correcta**?

```
1  Canción = namedtuple('Canción', 'nombre,artista,popularidad')
2
3  class ListaReproducción:
4      def __init__(self, canciones: list, forma_reproducción: str = '') -> None:
5          self.canciones = canciones
6          self.forma_reproducción = forma_reproducción
7
8      def __iter__(self) -> IteradorListaReproducción:
9          return IteradorListaReproducción(self.canciones, self.forma_reproducción)
10
11 class IteradorListaReproducción:
12     def __init__(self, canciones: list, forma_reproducción: str) -> None:
13         self.canciones = canciones.copy()
14
15         if forma_reproducción == 'populares':
16             filter(lambda c: c.popularidad > 100, self.canciones)
17
18     def __next__(self) -> Canción:
19         if not self.canciones:
20             raise StopIteration('Lista reproducción vacía')
21         self.canciones.pop(-1)
22         return self.canciones[0]
23
24 lista_reproducción = ListaReproducción([
25     Canción('Makeba', 'Jain', 341),
26     Canción('Symphony (feat. Zara Larsson)', 'Clean Bandit', 1567),
27     Canción('lil freak', 'bbnoS', 25),
28     Canción('Bad Boy', 'Yung Bae, ...', 94),
29     Canción('Virtual Insanity', 'Jamiroquai', 293)])
30 lista_reproducción.forma_reproducción = 'populares'
31
32 for canción in lista_reproducción:
33     print(f'Escuchando {canción.nombre} de {canción.artista}')
34 print('¡Escuchaste toda la lista de reproducción!')
```

A) Si se ejecuta la excepción "`StopIteration`" en la línea 20, se detendrá la ejecución del programa.

B) Si se ejecuta el "`filter`" de la línea 16, se actualizarán las canciones del atributo "`canciones`" de la clase "`IteradorListaReproducción`".

C) Cuando se itera sobre la variable "`lista_reproducción`" en las líneas 33-34, se produce un error debido a que la clase "`IteradorListaReproducción`" no tiene definido el método "`__iter__`".

D) Cuando se instancia la clase "`ListaReproducción`" en las líneas 24-29, se produce un error debido a que no se el entrega el valor del argumento "`forma_reproducción`".

E) Cuando se ejecuta el código completo, se imprimirá 5 veces la información de una canción y posteriormente el texto "`¡Escuchaste toda la lista de reproducción!`".

30. Para el siguiente código, ¿cuál es la alternativa que genera un *output* **no vacío** en la consola?

```
def foo(*args, **kwargs):
    print(kwargs)

datos = {'primero': 123, 'segundo': 53, 'ultimo': 124}
```

A) `foo(datos)`

B) `foo(*datos)`

C) `foo(datos, *datos)`

D) `foo(**datos)`

E) El *output* siempre sera vacío.

Preguntas bonus - Tópicos Avanzados 2

A continuación te encontrarás con 6 preguntas, la primera corresponde a la **pregunta 31** que no tiene puntaje asociado, pero nos permitirá identificar el tema que deseas abordar como pregunta *bonus*. Puedes escoger el tema que estimes conveniente y es obligación responder únicamente 1 alternativa. En caso de no seguir las instrucciones o no responder esta pregunta, el *bonus* será anulado.

31 ¿Cuál es el tema que responderás como pregunta 32?

A) Tema 1: Sistemas Recomendadores.

B) Tema 2: Simulación.

C) Tema 3: Aplicaciones Web a lo largo de la historia.

D) Tema 4: Seguridad computacional y *Scraping*.

E) Tema 5: *Machine Learning* y *Deep Learning* en la actualidad.

A continuación se presentan las 5 preguntas, una por tema, pero **solo debes responder la pregunta asociada al tema elegido previamente**. La respuesta la debes escribir en la hoja de respuesta como la **pregunta 32**:

Tema 1. Se ha estrenado un anime totalmente nuevo que, en términos de metadata, es muy diferente a los que existen hoy en día. No obstante, estaba tan mal animado que se hizo viral, al punto de que todo el mundo terminó viéndolo, pero calificándolo con 1 de 10 estrellas. ¿Cuál de los siguientes algoritmos **recomendaría** dicho anime a un usuario que no ha visto ningún anime en su vida?

A) *Most Popular*.

B) *Item Average*.

C) Filtrado Colaborativo basado en *rating*.

D) Basado en contenido.

E) Ninguno de los algoritmos va a recomendar ese anime.

Tema 2. Te encuentras modelando un problema, donde te dan el siguiente contexto:

> Se desea modelar la caleta de pescadores de un pequeño pueblo y sacar estadísticas sobre cuánto pescan en un día. Los pescadores salen del puerto a las 6:00 am y pescarán durante un lapso de 3 a 4 horas, permitiéndoles así volver para el horario de apertura de la caleta. La probabilidad de aparición de un cardumen estará dada por una distribución Poisson, mientras que la cantidad de peces de dicho cardumen estará dada por una distribución normal.

¿Qué paradigma de programación es el **más adecuado** para implementar el código pedido?

A) Simulación Síncrona.

B) Simulación por Eventos Discretos.

C) Ambos paradigmas son equivalentes.

D) Ambos paradigmas de forma conjunta.

E) Ninguna de las anteriores.

Tema 3. ¿Cuál de la siguientes alternativas corresponde a una comparación **correcta** entre una **página web** y una **aplicación web**?

A) Una página web solo puede revisarse desde un computador de escritorio o una *laptop*, mientras que una aplicación web puede ser vista desde otros dispositivos, como celulares.

B) Es posible ver todo el contenido de una página web sin iniciar sesión, mientras que en una aplicación web es necesario registrarse.

C) Una página web solo usa HTML, mientras que una aplicación web usa HTML, CSS y JavaScript.

D) Una aplicación web muestra contenido dinámico según cómo interactúe el usuario en esta, la página web siempre muestra lo mismo.

E) Ninguna de las anteriores es una comparación correcta.

Tema 4. ¿Cuál de la siguientes alternativas corresponde a las **funcionalidades principales** de un ataque de tipo **Gusano** (*Worm*)?

A) Engañar a una persona apelando a sus emociones y extraer información confidencial de ella.

B) Infectar un sistema y replicarse a lo largo de una red.

C) Simular el comportamiento de un *software* no malicioso e infectar un sistema.

D) Inyectar código *SQL* y guardar lo extraído.

E) Interceptar una comunicación y encriptar los mensajes.

Tema 5. Respecto a la división de datos, los cuales se le pasarán a un modelo de aprendizaje profundo. ¿Cuál es la distribución adecuada para datos de entrenamiento y datos de test?

A) 50 % entrenamiento, 50 % test.

B) 80 % entrenamiento, 20 % test.

C) 20 % entrenamiento, 80 % test.

D) 100 % entrenamiento, 0 % test.

E) 40 % entrenamiento, 60 % test.