**1. asyncio คืออะไร และทำไมมันถึงเหมาะสมกับการใช้งาน non-blocking I/O application?**  
= asyncio เป็นโมดูลใน Python ที่ช่วยจัดการกับการทำงานแบบ asynchronous concurrency (การทำงานพร้อมกันแบบไม่รอ) โดยเฉพาะในกรณีที่มีการทำงานกับ I/O แบบ non-blocking เช่น การรอรับข้อมูลจากเครือข่ายหรือการอ่าน/เขียนไฟล์ ซึ่งไม่ต้องให้โปรแกรมหยุดทำงานในขณะที่รอ ทำให้สามารถเพิ่มประสิทธิภาพในการรันโปรแกรมที่มี I/O หนัก ๆ ได้อย่างมีประสิทธิภาพ

**2.อธิบายความหมายของ coroutine ในบริบทของ Python และการใช้งาน asyncio**  
= coroutine เป็นฟังก์ชันที่ถูกสร้างขึ้นด้วยคีย์เวิร์ด async def และสามารถหยุดทำงานชั่วคราวได้ด้วยคำสั่ง await เพื่อรอให้ฟังก์ชัน asynchronous อื่นทำงานเสร็จก่อนที่จะทำงานต่อ Coroutines ใน asyncio เป็นฟังก์ชันที่ทำงานอย่างไม่เป็นลำดับ เพื่อจัดการกับการทำงานแบบ concurrent

**3. นิยาม task ใน asyncio และบอกหน้าที่หลักของมัน**  
= Task เป็น object ที่สร้างขึ้นเพื่อจัดการและดำเนินการ coroutine ในรูปแบบ asynchronous มันจะทำให้ coroutine ทำงานควบคู่ไปกับส่วนอื่น ๆ ของโปรแกรมโดยไม่ต้องรอให้เสร็จสิ้นก่อนจึงจะดำเนินการต่อ

**4.อธิบายฟังก์ชัน asyncio.run() ทำหน้าที่อะไร และเมื่อใดที่ควรใช้**  
= asyncio.run() เป็นฟังก์ชันที่ใช้ในการเรียกใช้ coroutine หลักและควบคุมการรัน event loop โดยอัตโนมัติ โดยจะสร้างและจัดการ event loop สำหรับรัน coroutine เมื่อรันเสร็จแล้วจะปิด loop ทันที นิยมใช้เมื่อคุณต้องการรัน coroutine หลักในโปรแกรม

**5. อธิบายการใช้ await ใน asyncio และมันทำงานอย่างไรกับ coroutine**  
= คำสั่ง await ใช้เพื่อระงับการทำงานของ coroutine จนกว่าจะมีการคืนค่าให้กับ asynchronous function (เช่น coroutine อื่น, task, หรือ future) คำสั่งนี้จะหยุดการทำงานชั่วคราวและอนุญาตให้รันส่วนอื่น ๆ ใน event loop ไปก่อน

**6. อธิบายวิธีสร้าง coroutine ด้วยการใช้ async def และการรันด้วย await**  
คุณสามารถสร้าง coroutine ด้วยการใช้ async def ดังนี้:

A black and grey rectangular object

Description automatically generated

และเพื่อรัน coroutine คุณสามารถใช้คำสั่ง await ใน event loop:

A black and grey rectangular object

Description automatically generated

**7.อธิบายความหมายของ async comprehension และยกตัวอย่างการใช้งาน**Async comprehension คือการใช้ comprehension เช่น list หรือ set ที่ภายในใช้ async for เพื่อ iterate ข้อมูลจาก asynchronous generator ตัวอย่าง:

A close up of a number

Description automatically generated

**8.นิยามของ async for-loop คืออะไร และมันต่างจาก for-loop ปกติอย่างไร?**Async for-loop ใช้ในการ loop ข้อมูลที่มาจาก asynchronous iterator หรือ asynchronous generator ซึ่งมีการใช้ await ในการดึงค่าจากแต่ละ step แตกต่างจาก for-loop ปกติที่ใช้กับ iterator ปกติ ตัวอย่าง:

A black and grey rectangular object

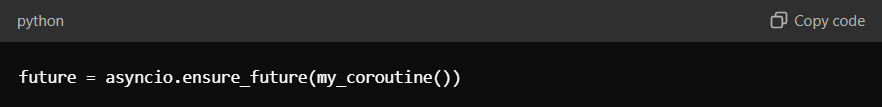
Description automatically generated

**9 .อธิบายการใช้งานของ asyncio.create\_task() ในการสร้างและจัดการ task**  
asyncio.create\_task() ใช้เพื่อสร้าง task ใหม่จาก coroutine และทำให้ coroutine นั้นทำงานแบบ asynchronous โดย task จะถูกรันควบคู่ไปกับ event loop อื่น ๆ โดยไม่ต้องรอให้ task นั้นเสร็จสิ้นก่อน:

A black and white text

Description automatically generated

**10. อธิบายฟังก์ชัน asyncio.ensure\_future() และเปรียบเทียบกับ asyncio.create\_task()**asyncio.ensure\_future() สร้างและจัดการ coroutine ในลักษณะเดียวกับ asyncio.create\_task() แต่มีความยืดหยุ่นในการรับทั้ง task ที่ถูกสร้างขึ้นแล้วหรือ coroutine ที่ยังไม่ได้สร้าง task การใช้ ensure\_future() จะมีประโยชน์ในสถานการณ์ที่ต้องการตรวจสอบวัตถุที่ส่งมาเป็น task อยู่แล้วหรือยังไม่ใช่:



**11. นิยามของ shielded task ใน asyncio คืออะไร และใช้งาน asyncio.shield() อย่างไร?**Shielded task คือ task ที่ถูกป้องกันจากการยกเลิกด้วยการใช้ asyncio.shield(), หากคุณต้องการให้ task ไม่ถูกยกเลิกเมื่อมีการยกเลิก parent task คุณสามารถใช้ shield():

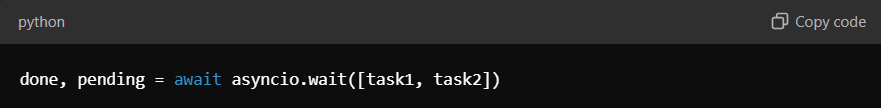


**12. อธิบายฟังก์ชัน asyncio.gather() และเมื่อใดที่ควรใช้งานมัน**  
asyncio.gather() ใช้เพื่อรันหลาย ๆ coroutine พร้อมกันและรวบรวมผลลัพธ์ของมันทั้งหมด เหมาะกับการทำงานที่ต้องรันหลาย task พร้อมกันและต้องการรอให้เสร็จทั้งหมด:

A black and white screen with white text

Description automatically generated

**13. อธิบายวิธีการรอหลาย task พร้อมกันด้วย asyncio.wait()**  
asyncio.wait() ใช้เพื่อรอหลาย task หรือ coroutines พร้อมกัน คุณสามารถรอให้ task ทั้งหมดเสร็จหรือหยุดหลังจาก task แรกเสร็จ:



**14. นิยามของ asyncio.TimeoutError คืออะไร และใช้เมื่อใด?**asyncio.TimeoutError เป็นข้อผิดพลาดที่ถูกยกขึ้นเมื่อการรอ coroutine หรือ task เกินเวลาที่กำหนด (timeout) ใช้ในกรณีที่ต้องการตั้งเวลาจำกัดให้กับงาน

**15. อธิบายวิธีการใช้ asyncio.wait\_for() ในการจัดการ timeout ให้กับ coroutine**  
asyncio.wait\_for() ใช้เพื่อรัน coroutine หรือ task พร้อมกับกำหนดเวลา timeout หากเกินเวลา timeout ที่กำหนดไว้ task จะถูกยกเลิก:

A black and white screen with white text

Description automatically generated

คำตอบเกี่ยวกับ Debugging:

**16. ใน asyncio จะทำอย่างไรเพื่อ ตรวจสอบสถานะ ของ task ที่กำลังทำงาน เช่น รันอยู่, เสร็จสิ้น หรือถูกยกเลิก?**  
คุณสามารถตรวจสอบสถานะของ task ได้ด้วย:

* task.done() ตรวจสอบว่า task เสร็จหรือยัง
* task.cancelled() ตรวจสอบว่า task ถูกยกเลิกหรือไม่
* task.exception() ตรวจสอบว่า task มีข้อผิดพลาดหรือไม่

**17. จะ ยกเลิก task อย่างไรใน asyncio โดยใช้ฟังก์ชัน task.cancel() และตรวจสอบว่าถูกยกเลิกสำเร็จหรือไม่?**  
คุณสามารถยกเลิก task ได้โดยใช้ task.cancel() และตรวจสอบการยกเลิกด้วย task.cancelled():

A black rectangular object with a black border

Description automatically generated

**18. อธิบายวิธีการตรวจสอบ unhandled exceptions ใน task โดยใช้ task.exception()**  
ใช้ task.exception() เพื่อดูข้อผิดพลาดที่เกิดขึ้นใน task หากไม่มีข้อผิดพลาดจะคืนค่า None หากมีข้อผิดพลาดจะคืนค่า exception:

A black screen with white text

Description automatically generated

**19. จะเพิ่มฟังก์ชัน callback เมื่อ task เสร็จสิ้นได้อย่างไร โดยใช้ task.add\_done\_callback()?**ใช้ task.add\_done\_callback() เพื่อเพิ่ม callback ที่จะถูกเรียกเมื่อ task เสร็จ:

A black rectangular object with a grey border

Description automatically generated

**20. วิธีการ ลบ callback ออกจาก task ที่ถูกเพิ่มไว้ก่อนหน้านี้ใน asyncio ควรทำอย่างไร?**  
คุณสามารถลบ callback ออกได้โดยการไม่ใส่ callback อีกต่อไป และแทนที่ task ใหม่โดยไม่ใช้ callback

A screenshot of a computer program

Description automatically generated



A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A white background with black text

Description automatically generated