Asynchronous Lab2



1-1-simple-sync.py

```
class5 > 1-1-simple-sync.py > ...
     # 1-1-simple-sync.py
     import time
 3
 4
     def sleep():
         print(f'Time: {time.time() - start:.2f}')
         time.sleep(1)
 6
 7
 8
 9
     def sum(name, numbers):
10
          total = 0
          for number in numbers:
11
             print(f'Task {name}: Computing {total}+{number}')
12
13
             sleep()
14
             total += number
15
         print(f'Task {name}: Sum = {total}\n')
16
17
18
     start = time.time()
19
     #
20
     tasks = [
21
         sum("A", [1, 2]),
         sum("B", [1, 2, 3]),
22
23
24
     end = time.time()
     print(f'Time: {end-start:.2f} sec')
```

1-2-simple-async-wrong.py

```
class5 > 1-2-simple-async-wrong.py > ...
      import asyncio
     import time
     async def sleep():
          print(f'Time: {time.time() - start:.2f}')
 6
          time.sleep(1)
     async def sum(name, numbers):
 9
          total = 0
10
          for number in numbers:
              print(f'Task {name}: Computing {total}+{number}')
11
12
              await sleep()
13
              total += number
          print(f'Task {name}: Sum = {total}\n')
14
15
16
      start = time.time()
17
18
     loop = asyncio.get_event_loop()
     tasks = [
19
         loop.create_task(sum("A", [1, 2])),
20
          loop.create_task(sum("B", [1, 2, 3])),
21
22
23
     loop.run_until_complete(asyncio.wait(tasks))
     loop.close()
24
25
26
     end = time.time()
     print(f'Time: {end-start:.2f} sec')
```

1-3-simple-async-right.py

```
class5 > 1-3-simple-async-right.py > ...
 1 ∨ import asyncio
     import time
 4 \times async def sleep():
          print(f'Time: {time.time() - start:.2f}')
 6
          await asyncio.sleep(1)
   v async def sum(name, numbers):
 9
          total = 0
10 ~
          for number in numbers:
11
              print(f'Task {name}: Computing {total}+{number}')
              await sleep()
12
              total += number
13
          print(f'Task {name}: Sum = {total}\n')
14
15
      start = time.time()
16
17
     loop = asyncio.get_event_loop()
18
19 \vee tasks = [
          loop.create_task(sum("A", [1, 2])),
20
          loop.create_task(sum("B", [1, 2, 3])),
21
22
     loop.run_until_complete(asyncio.wait(tasks))
23
     loop.close()
24
25
26
      end = time.time()
     print(f'Time: {end-start:.2f} sec')
27
```

1-4-simple-async-right2.py

```
class5 > 1-4-simple-async-right2.py > ...
 1 ∨ import asyncio
      import time
 3
 4 \sim async def sleep():
          print(f'Time: {time.time() - start:.2f}')
          await asyncio.sleep(1)
 6
 8 vasync def sum(name, numbers):
          total = 0
          for number in numbers:
10 ~
              print(f'Task {name}: Computing {total}+{number}')
11
12
              await sleep()
              total += number
13
          print(f'Task {name}: Sum = {total}\n')
14
15
16 \vee async def main():
          await asyncio.gather(sum("A", [1, 2]), sum("B", [1, 2, 3]))
17
18
19 \( \times \text{if } __name__ == '__main__':
          start = time.time()
20
          asyncio.run(main())
21
          end = time.time()
22
          print(f'Time: {end-start:.2f} sec')
23
```

1-5-simple-thread.py

```
class5 > 1-5-simple-thread.py > ...
 1 ∨ import asyncio
      import time
     from concurrent.futures.thread import ThreadPoolExecutor
 5 \vee def sleep():
          print(f'Time: {time.time() - start:.2f}')
 6
          time.sleep(1)
 9 vasync def sum(name, numbers):
          _executor = ThreadPoolExecutor(2)
10
          total = 0
11
12 ~
          for number in numbers:
              print(f'Task {name}: Computing {total}+{number}')
13
14
              await loop.run_in_executor(_executor, sleep)
15
              total += number
16
          print(f'Task {name}: Sum = {total}\n')
17
18
      start = time.time()
19
     loop = asyncio.get_event_loop()
20
21 \vee tasks = [
22
          loop.create_task(sum("A", [1, 2])),
          loop.create_task(sum("B", [1, 2, 3])),
23
24
      loop.run_until_complete(asyncio.wait(tasks))
25
26
      loop.close()
27
28
     end = time.time()
     print(f'Time: {end-start:.2f} sec')
```

2-1-basic.py

```
class5 > 2-1-basic.py > ...
     import asyncio
 1
 2
     import time
 3
 4
     async def hello(i):
 5
         print(f"{time.ctime()} hello {i} started")
         await asyncio.sleep(4)
 6
         print(f"{time.ctime()} hello {i} done")
 7
 8
 9
     async def main():
10
         task1 = asyncio.create_task(hello(1)) # returns immediately, the task is created
11
         #await asyncio.sleep(3)
12
         task2 = asyncio.create_task(hello(2))
13
         await task1
14
         await task2
15
16
     if name == '_main ':
17
         start = time.time()
18
         asyncio.run(main())
         end = time.time()
19
         print(f'Time: {end-start:.2f} sec')
20
21
```

2-2-basic-gather.py

```
class5 > 2-1-basic-gather.py > ...
      import asyncio
 1
 2
      import time
 3
 4
      async def hello(i):
 5
          print(f"{time.ctime()} hello {i} started")
          await asyncio.sleep(4)
 6
          print(f"{time.ctime()} hello {i} done")
 7
 8
 9
      async def main():
10
          task1 = asyncio.create_task(hello(1)) # returns immediately, the task is created
11
         #await asyncio.sleep(3)
12
         task2 = asyncio.create_task(hello(2))
13
          await asyncio.gather(task1, task2)
14
15
      if __name__ == '__main__':
16
          start = time.time()
          asyncio.run(main())
17
         end = time.time()
18
19
          print(f'Time: {end-start:.2f} sec')
20
```

2-3-basic-gather-more.py

```
class5 > 2-2-basic-gather-more.py > ...
      import asyncio
 1
 2
      import time
 3
 4
     async def hello(i):
 5
          print(f"{time.ctime()} hello {i} started")
 6
          await asyncio.sleep(4)
          print(f"{time.ctime()} hello {i} done")
 7
 8
 9
     async def main():
10
          coros = [hello(i) for i in range(10)]
11
          await asyncio.gather(*coros)
12
13
      if name == ' main ':
14
          start = time.time()
15
          asyncio.run(main())
16
          end = time.time()
17
          print(f'Time: {end-start:.2f} sec')
18
```

3-basic-fac.py

```
class5 > 3-basic-fac.pv > ...
      import asyncio
 1
     import time
 2
 3
     async def factorial(n):
 4
         f = 1
 5
          for i in range(2, n + 1):
 6
              print(f"Computing factorial({n}), currently i={i}...")
 7
              await asyncio.sleep(1)
 8
              f *= i
 9
10
          return f
11
12
     async def main():
13
          L = await asyncio.gather(factorial(2), factorial(3), factorial(4))
14
         print(L) # [2, 6, 24]
15
16
      if __name__ == '__main__':
17
          start = time.time()
          asyncio.run(main())
18
19
         end = time.time()
          print(f'Time: {end-start:.2f} sec')
20
21
```

Chess

- Chess master Judit Polgár hosts a chess exhibition in which she plays multiple amateur players. She has two ways of conducting the exhibition: synchronously and asynchronously.
 - Assumptions:
 - 24 opponents
 - Judit makes each chess move in 5 seconds
 - Opponents each take 55 seconds to make a move
 - Games average 30 pair-moves (60 moves total)

Show-it

- Synchronous version: Judit plays one game at a time, never two at the same time, until the game is complete.
 Each game takes (55 + 5) * 30 == 1800 seconds, or 30 minutes.
 - The entire exhibition takes == ??? minutes.
- **Asynchronous version:** Judit moves from table to table, making one move at each table. She leaves the table and lets the opponent make their next move during the wait time. One move on all 24 games takes Judit 24 * 5 == 120 seconds, or 2 minutes.
 - The entire exhibition is now cut down to ???? seconds.