Asynchronous Lab2

https://github.com/hanattaw/class5-asyncio.git

1-1-simple-sync.py

class5 > 1-1-simple-sync.py > ...

```python
# 1-1-simple-sync.py
import time

def sleep():
    print(f'Time: {time.time() - start:.2f}')
    time.sleep(1)

#
def sum(name, numbers):
    total = 0
    for number in numbers:
        print(f'Task {name}: Computing {total}+{number}')
        sleep()
        total += number
    #
    print(f'Task {name}: Sum = {total}\n')

start = time.time()
#
tasks = [
    sum("A", [1, 2]),
    sum("B", [1, 2, 3]),
]
end = time.time()
print(f'Time: {end-start:.2f} sec')
```

1-2-simple-async-wrong.py

class5 > 1-2-simple-async-wrong.py > ...

```python
import asyncio
import time

async def sleep():
    print(f'Time: {time.time() - start:.2f}')
    time.sleep(1)

async def sum(name, numbers):
    total = 0
    for number in numbers:
        print(f'Task {name}: Computing {total}+{number}')
        await sleep()
        total += number
    print(f'Task {name}: Sum = {total}\n')

start = time.time()

loop = asyncio.get_event_loop()
tasks = [
    loop.create_task(sum("A", [1, 2])),
    loop.create_task(sum("B", [1, 2, 3])),
]
loop.run_until_complete(asyncio.wait(tasks))
loop.close()

end = time.time()
print(f'Time: {end-start:.2f} sec')
```

1-3-simple-async-right.py

class5 > 1-3-simple-async-right.py > ...

```python
import asyncio
import time

async def sleep():
    print(f'Time: {time.time() - start:.2f}')
    await asyncio.sleep(1)

async def sum(name, numbers):
    total = 0
    for number in numbers:
        print(f'Task {name}: Computing {total}+{number}')
        await sleep()
        total += number
    print(f'Task {name}: Sum = {total}\n')

start = time.time()

loop = asyncio.get_event_loop()
tasks = [
    loop.create_task(sum("A", [1, 2])),
    loop.create_task(sum("B", [1, 2, 3])),
]
loop.run_until_complete(asyncio.wait(tasks))
loop.close()

end = time.time()
print(f'Time: {end-start:.2f} sec')
```

1-4-simple-async-right2.py

class5 > 1-4-simple-async-right2.py > ...

```python
import asyncio
import time

async def sleep():
    print(f'Time: {time.time() - start:.2f}')
    await asyncio.sleep(1)

async def sum(name, numbers):
    total = 0
    for number in numbers:
        print(f'Task {name}: Computing {total}+{number}')
        await sleep()
        total += number
    print(f'Task {name}: Sum = {total}\n')

async def main():
    await asyncio.gather(sum("A", [1, 2]), sum("B", [1, 2, 3]))

if __name__ == '__main__':
    start = time.time()
    asyncio.run(main())
    end = time.time()
    print(f'Time: {end-start:.2f} sec')
```

1-5-simple-thread.py

class5 > 1-5-simple-thread.py > ...

```python
import asyncio
import time
from concurrent.futures.thread import ThreadPoolExecutor

def sleep():
    print(f'Time: {time.time() - start:.2f}')
    time.sleep(1)

async def sum(name, numbers):
    _executor = ThreadPoolExecutor(2)
    total = 0
    for number in numbers:
        print(f'Task {name}: Computing {total}+{number}')
        await loop.run_in_executor(_executor, sleep)
        total += number
    print(f'Task {name}: Sum = {total}\n')

start = time.time()

loop = asyncio.get_event_loop()
tasks = [
    loop.create_task(sum("A", [1, 2])),
    loop.create_task(sum("B", [1, 2, 3])),
]
loop.run_until_complete(asyncio.wait(tasks))
loop.close()

end = time.time()
print(f'Time: {end-start:.2f} sec')
```

2-1-basic.py

class5 > 2-1-basic.py > ...

```python
import asyncio
import time

async def hello(i):
    print(f"{time.ctime()} hello {i} started")
    await asyncio.sleep(4)
    print(f"{time.ctime()} hello {i} done")

async def main():
    task1 = asyncio.create_task(hello(1))  # returns immediately, the task is created
    #await asyncio.sleep(3)
    task2 = asyncio.create_task(hello(2))
    await task1
    await task2

if __name__ == '__main__':
    start = time.time()
    asyncio.run(main())
    end = time.time()
    print(f'Time: {end-start:.2f} sec')

```

2-2-basic-gather.py

class5 > 2-1-basic-gather.py > ...

```python
import asyncio
import time

async def hello(i):
    print(f"{time.ctime()} hello {i} started")
    await asyncio.sleep(4)
    print(f"{time.ctime()} hello {i} done")

async def main():
    task1 = asyncio.create_task(hello(1))  # returns immediately, the task is created
    #await asyncio.sleep(3)
    task2 = asyncio.create_task(hello(2))
    await asyncio.gather(task1, task2)

if __name__ == '__main__':
    start = time.time()
    asyncio.run(main())
    end = time.time()
    print(f'Time: {end-start:.2f} sec')

```

2-3-basic-gather-more.py

class5 > 2-2-basic-gather-more.py > ...

```python
import asyncio
import time

async def hello(i):
    print(f"{time.ctime()} hello {i} started")
    await asyncio.sleep(4)
    print(f"{time.ctime()} hello {i} done")

async def main():
    coros = [hello(i) for i in range(10)]
    await asyncio.gather(*coros)

if __name__ == '__main__':
    start = time.time()
    asyncio.run(main())
    end = time.time()
    print(f'Time: {end-start:.2f} sec')

```

3-basic-fac.py

class5 > 3-basic-fac.py > ...

```python
import asyncio
import time

async def factorial(n):
    f = 1
    for i in range(2, n + 1):
        print(f"Computing factorial({n}), currently i={i}...")
        await asyncio.sleep(1)
        f *= i
    return f

async def main():
    L = await asyncio.gather(factorial(2), factorial(3), factorial(4))
    print(L)  # [2, 6, 24]

if __name__ == '__main__':
    start = time.time()
    asyncio.run(main())
    end = time.time()
    print(f'Time: {end-start:.2f} sec')

```

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.**