

 Home(/)

0x02. Python - Async Comprehension

 Python My Background(/planning/me)

Weight: 1

Projects(/projects/current)



Project over - took place from Jul 9, 2024 6:00 AM to Jul 10, 2024 6:00 AM



An auto QA review will automatically make corrections(/to_review)



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

In a nutshell...

- **Auto QA review:** 16.0/18 mandatory

- **Altogether:** 88.89%



Curriculums(/dashboards/my_curriculums)

◦ Mandatory: 88.89%

◦ Optional: no optional tasks



Concepts(/concepts)



Conference rooms(/dashboards/video_rooms)



Servers(/servers)

@iamdevloper



Sandboxes(/user_containers/current)



Roses are red

Tools(/dashboards/my_tools)



And so are you

Video on demand(/dashboards/videos)

Violets are blue



Asynchronous operations

Peers(/users/peers)



are great

Discord(https://discord.com/app)

15:53 · 14 Feb 19 · Twitter Web App

My Profile(/users/my_profile)



Resources

Read or watch:



- PEP 530 – Asynchronous Comprehensions (/rltoken/hlwtED-iLsdORSgly8DsyQ)
- What's New in Python: Asynchronous Comprehensions / Generators (/rltoken/0OkbObYzCKtO7ZUAxfKvkw)
- Type-hints for generators (/rltoken/I4Fnno568VbVln9GvrFVtQ)



My Planning(/planning/me)

Learning Objectives



Projects(/projects/current)

At the end of this project, you are expected to be able to explain to anyone (/rltoken/_jK22HqiCeh5NjKJ4ZHBww), **without the help of Google**:



- How to write an asynchronous generator (our own review)
- How to use async comprehensions
- How to type-annotate generators



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Requirements



General

Curriculums(/dashboards/my_curriculums)



- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using `python3` (version 3.7)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/env python3`



A README.md file, at the root of the folder of the project, is mandatory

- Your code should use the `pycodestyle` style (version 2.5.x)
- The length of your files will be tested using `wc`



Servers(/servers)

- All your modules should have a documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)



Sandboxes(/user_containers/current)

- All your functions should have a documentation (`python3 -c 'print(__import__("my_module").my_function.__doc__)'`)



- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module (dashboards/my_tools)

- All your functions and coroutines must be type-annotated.



Video on demand(/dashboards/videos)

Tasks



Peers(/users/peers)

0. Async Generator

mandatory



Discord(<https://discord.com/app>)

Score: 83.33% (Checks completed: 83.33%)



Write a coroutine called `async_generator` that takes no arguments.

The coroutine will loop 10 times, each time asynchronously wait 1 second, then yield a random number between 0 and 10. Use the `random` module.

My Profile(/users/my_profile)

```
bob@dylan:~$ cat 0-main.py
#!/usr/bin/env python3
()
import asyncio


async def print_yielded_values():
    for i in async_generator():
        result.append(i)

asyncio.run(print_yielded_values())

QA Reviews I can make(/corrections/to_review)
bob@dylan:~$ ./0-main.py
[4.403136952967102, 6.9092712604587465, 6.293445466782645, 4.549663490048418, 4.1326571686139915, 2.890585153949023, 6.726734195473811, 2.84331704602206, 1.0067279479988345, 1.3783306401737838]
```

 **Repo:** [Curriculums\(/dashboards/my_curriculums\)](#)

- GitHub repository: alx-backend-python
- Directory: 0x02-python_async_comprehension
- File: 0-async_generator.py

 **Conference rooms(/dashboards/video_rooms)**

[Check submission](#)

[Mark submission](#)


[Get a sandbox](#)


[View results](#)

 **Servers(/servers)**

mandatory

 **Score: 100.00% (as a user container)**

 **Tools(/dashboards/my_tools)**
Import `async_generator` from the previous task and then write a coroutine called `async_comprehension` that takes no arguments.

 **Video on demand(/dashboards/videos)**
The coroutine will collect 10 random numbers using an async comprehensing over `async_generator`, then return the 10 random numbers.

 **Peers(/users/peers)**

 **Discord(<https://discord.com/app>)**



My Profile(/users/my_profile)

bob@dylan:~\$ cat 1-main.py

```
#!/usr/bin/env python3
()
import asyncio

async_comprehension = __import__('1-async_comprehension').async_comprehension

My Planning(/planning/me)
print(await async_comprehension())

Projects(/projects/current)
async_comprehension()

bob@dylan:~$ ./1-main.py
[9.86134195071727, 0.5720355293154995, 1.74677182056248265, 4.0724372912858575, 0.5524750922145316, 8.084266576021555, 8.387128918690468, 1.5486451376520916, 7.713335177885325, 7.673533267041574]
```

?

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Repo:

- Curriculums(/dashboards/my_curriculums)
- GitHub repository: alx-backend-python
- Directory: 0x02-python_async_comprehension
- File: 1-async_comprehension.py

Concepts(/concepts)

Check score (/dashboards/my_current_evaluation_quizzes)

Conference room (/dashboards/my_conference_rooms)

Get handbooks (/dashboards/my_handbooks)

View results (/dashboards/my_results)

2. Run time for four parallel comprehensions

Servers(/servers)

mandatory

Score: 83.33% (Checks completed: 83.33%)

Sandboxes(/user_containers/current)

Import `async_comprehension` from the previous file and write a `measure_runtime` coroutine that will execute `async_comprehension` four times in parallel using `asyncio.gather`.

Tools(/dashboards/my_tools)

`measure_runtime` should measure the total runtime and return it.

Notice that the total runtime is roughly 10 seconds, explain it to yourself.

Video on demand (/dashboards/my_videos)

Peers(/users/peers)

Discord(<https://discord.com/app>)

My Profile(/users/my_profile)



```
bob@dylan:~$ cat 2-main.py
#!/usr/bin/env python3
()
import asyncio


Home(/)
measure_runtime = __import__('2-measure_runtime').measure_runtime

My Planning(/planning/me)
async def main():
    return await(measure_runtime())
Projects(/projects/current)
print(
    asyncio.run(main())
)
QA Reviews I can make(/corrections/to_review)

bob@dylan:~$ ./2-main.py
10.021936893463135
? Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
```


 **Repo:** Curriculums(/dashboards/my_curriculums)

- GitHub repository: alx-backend-python
- Directory: 0x02-python_async_comprehension Concepts(concepts)
- File: 2-measure_runtime.py

 **Conference rooms**(/dashboards/video_rooms)


Check submission

Mark submission

 Get a sandbox

View results


 Servers(/servers)

 Sandboxes(/user_containers/current)

Copyright © 2024 ALX, All rights reserved.

 Tools(/dashboards/my_tools)

 Video on demand(/dashboards/videos)

 Peers(/users/peers)

 Discord(<https://discord.com/app>)



My Profile(/users/my_profile)