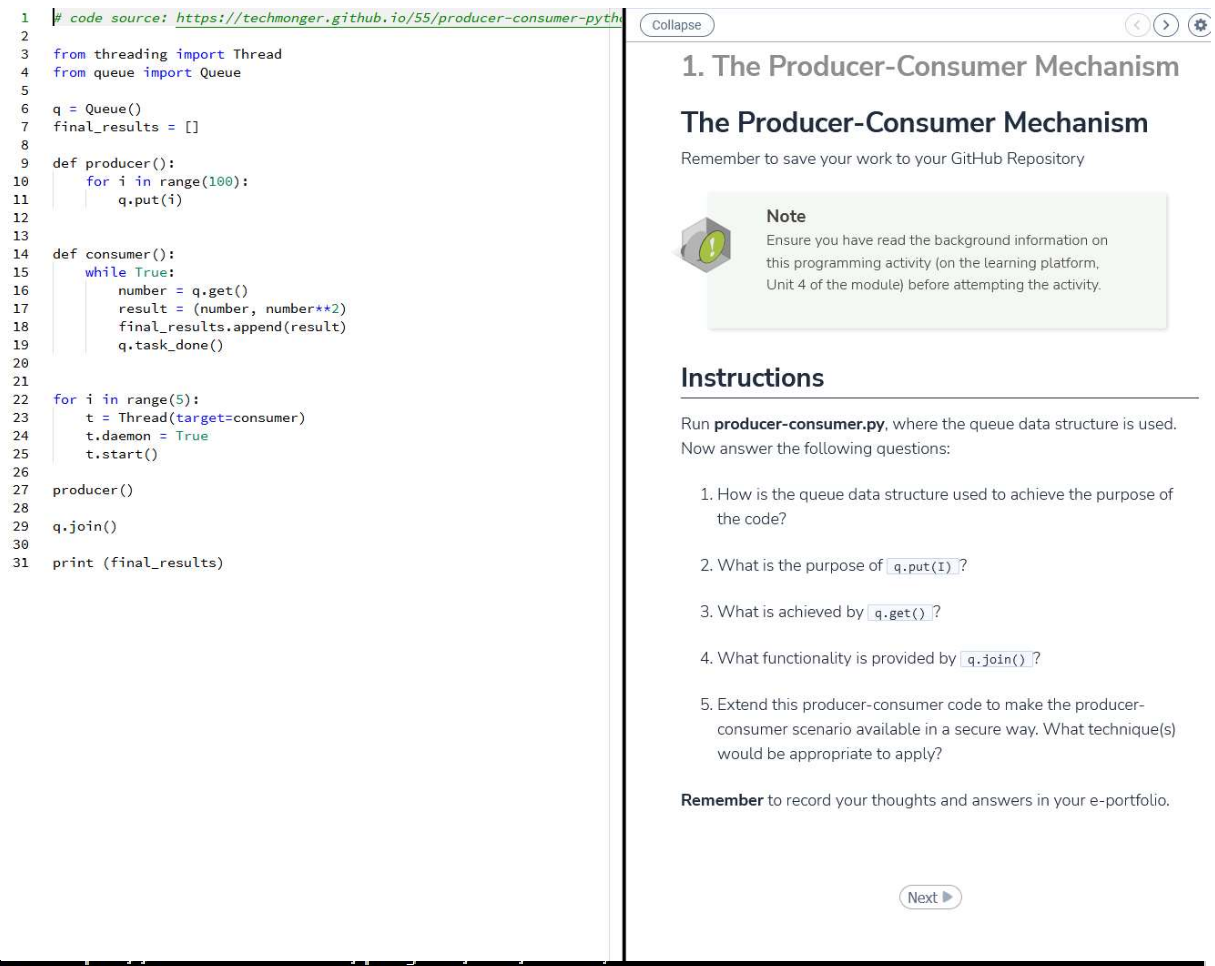


```python
# code source: https://techmonger.github.io/55/producer-consumer-pyth

from threading import Thread
from queue import Queue

q = Queue()
final_results = []

def producer():
    for i in range(100):
        q.put(i)


def consumer():
    while True:
        number = q.get()
        result = (number, number**2)
        final_results.append(result)
        q.task_done()


for i in range(5):
    t = Thread(target=consumer)
    t.daemon = True
    t.start()

producer()

q.join()

print (final_results)
```

1. The Producer-Consumer Mechanism

The Producer-Consumer Mechanism

Remember to save your work to your GitHub Repository

Note

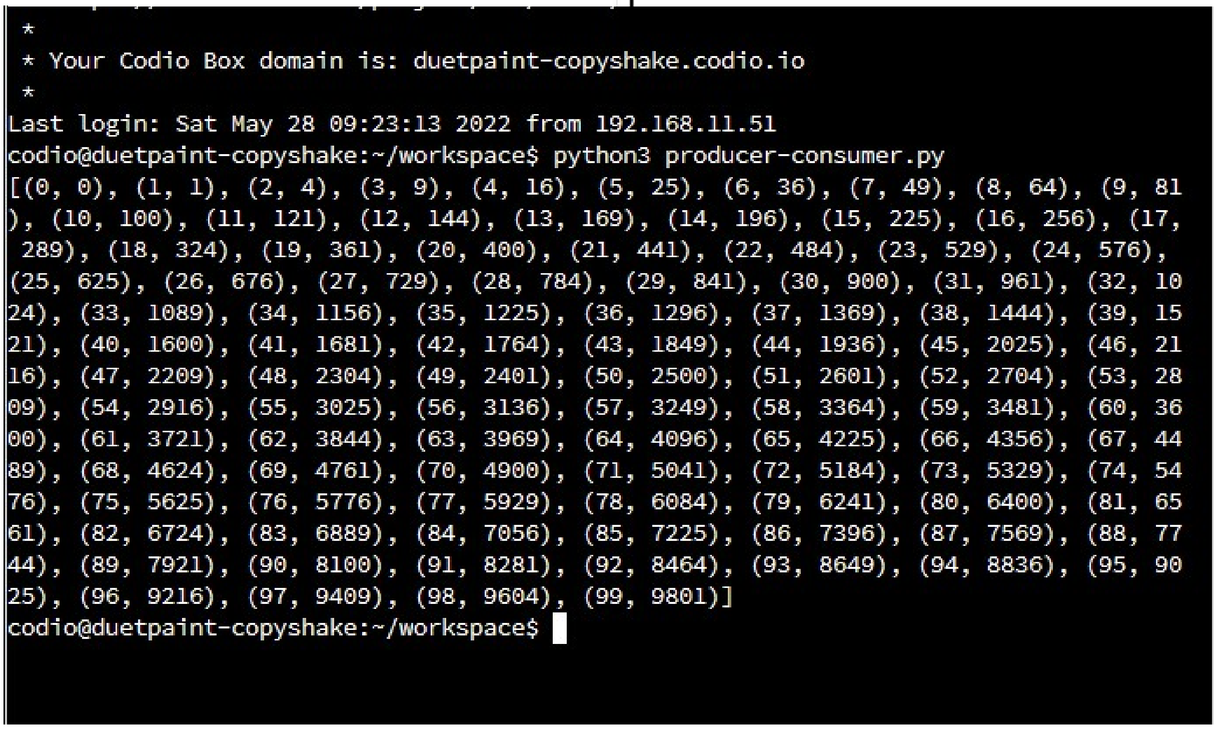
Ensure you have read the background information on this programming activity (on the learning platform, Unit 4 of the module) before attempting the activity.

Instructions

Run **producer-consumer.py**, where the queue data structure is used. Now answer the following questions:

1. How is the queue data structure used to achieve the purpose of the code?
2. What is the purpose of `q.put(I)`?
3. What is achieved by `q.get()`?
4. What functionality is provided by `q.join()`?
5. Extend this producer-consumer code to make the producer-consumer scenario available in a secure way. What technique(s) would be appropriate to apply?

Remember to record your thoughts and answers in your e-portfolio.



```
 *
 * Your Codio Box domain is: duetpaint-copyshake.codio.io
 *
Last login: Sat May 28 09:23:13 2022 from 192.168.11.51
codio@duetpaint-copyshake:~/workspace$ python3 producer-consumer.py
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64), (9, 81
), (10, 100), (11, 121), (12, 144), (13, 169), (14, 196), (15, 225), (16, 256), (17,
 289), (18, 324), (19, 361), (20, 400), (21, 441), (22, 484), (23, 529), (24, 576),
(25, 625), (26, 676), (27, 729), (28, 784), (29, 841), (30, 900), (31, 961), (32, 10
24), (33, 1089), (34, 1156), (35, 1225), (36, 1296), (37, 1369), (38, 1444), (39, 15
21), (40, 1600), (41, 1681), (42, 1764), (43, 1849), (44, 1936), (45, 2025), (46, 21
16), (47, 2209), (48, 2304), (49, 2401), (50, 2500), (51, 2601), (52, 2704), (53, 28
09), (54, 2916), (55, 3025), (56, 3136), (57, 3249), (58, 3364), (59, 3481), (60, 36
00), (61, 3721), (62, 3844), (63, 3969), (64, 4096), (65, 4225), (66, 4356), (67, 44
89), (68, 4624), (69, 4761), (70, 4900), (71, 5041), (72, 5184), (73, 5329), (74, 54
76), (75, 5625), (76, 5776), (77, 5929), (78, 6084), (79, 6241), (80, 6400), (81, 65
61), (82, 6724), (83, 6889), (84, 7056), (85, 7225), (86, 7396), (87, 7569), (88, 77
44), (89, 7921), (90, 8100), (91, 8281), (92, 8464), (93, 8649), (94, 8836), (95, 90
25), (96, 9216), (97, 9409), (98, 9604), (99, 9801)]
codio@duetpaint-copyshake:~/workspace$
```

How is the queue data structure used to achieve the purpose of the code?

- ➔ In the code the function producer is used to put an int (could be any item) in the queue, while the function consumer gets the int from the queue and use it for further calculation.

What is the purpose of `q.put(I)`?

- ➔ The purpose is to put an int inside the queue.

What is achieved by `q.get()`?

- ➔ The consumer get an item out of the queue

What functionality is provided by `q.join()`?

- ➔ The method q.join() is preventing the program from exiting while the method task_done is not executed. Both methods work therefore in cooperation.

Extend this producer-consumer code to make the producer-consumer scenario available in a secure way. What technique(s) would be appropriate to apply?

- ➔ To ensure security in the scenario a synchronous key encryption could be used.