Quiz 1

Test what you have learnt so far.

Question # 1

Consider the below snippet:

```
def thread_task():
    print("{0} executing".format(current_thread().getName()))


myThread = Thread(group=None,  # reserved
                  target=thread_task(),
                  name="childThread")

myThread.start()
myThread.join()
```

Q What will be the output of the above snippet?

- A) MainThread executing
- B) childThread executing

C) Runtime Error occurs

Submit Answer

Reset Quiz

```
1  from threading import Thread
2  from threading import current_thread
3
4
5  def thread_task():
6      print("{0} executing".format(current_thread().getName()))
7
8
9  myThread = Thread(group=None,  # reserved
10                   target=thread_task(),
11                   name="childThread")
12
13 myThread.start()
14 myThread.join()
15
```

Question

What are the different ways of creating threads?

Q

A) Using the **`Thread`** class

B) By subclassing `Thread` class

C) Both A and B

Submit Answer

Reset Quiz

Question

A Python program will wait for a daemon thread to finish before exiting.

Q

A) True

B) False

Submit Answer

Reset Quiz

Question # 4

Question # 4

Kim is trying to change the daemonic property of a thread she spawns in the `thread_task()` below. What do you think would be the out come of running the snippet below?

```
def thread_task():
    # Attempt to make the thread daemonic
    current_thread().setDaemon(True)
    while True:
        print("{0} executing".format(current_thread().getName
()))

myThread = Thread(target=thread_task(),
                  name="childThread")

myThread.start()
myThread.join()
```

Q

- A) RuntimeError
- B) Successful completion of program
- C) Program waits for daemon thread to end

Submit Answer

Reset Quiz

```
from threading import Thread
from threading import current_thread

def thread_task():
    # Attempt to make the thread daemonic
    current_thread().setDaemon(True)
    while True:
        print("{0} executing".format(current_thread().getName()))

myThread = Thread(target=thread_task(),
                  name="childThread")

myThread.start()
myThread.join()
```

Question

Consider the snippet below:

```
from threading import Lock

lock = Lock()
lock.acquire()
lock.release()
lock.release()
```

Q What would be the output of running the above snippet?

A) Successful completion

B) Deadlock

C) RuntimeError

Submit Answer

Reset Quiz

```
1  from threading import Lock
2
3  lock = Lock()
4  lock.acquire()
5  lock.release()
6  lock.release()
```

Question

Consider the snippet below:

```
from threading import Lock

lock = Lock()
lock.acquire()
lock.acquire()
lock.release()
```

Q What will the outcome of running the above snippet?

A) Successful completion

B) Deadlock

C) RuntimeError

Submit Answer

Reset Quiz

```
1  from threading import Lock
2
3  lock = Lock()
4  lock.acquire()
5  lock.acquire()
6  lock.release()
```

Running the above snippet will result in an error because the code widget times out waiting for the program to finish, which is deadlocked.

Question

Consider the setup below:

```
from threading import Thread
from threading import Condition
import time

flag = False
lock = Lock()
cond_var = Condition()


def child_thread():
    cond_var.acquire()
    while not flag:
        cond_var.wait()

    # enter a useless loop, till flag becomes false
    while flag:
        None


childThread = Thread(target=child_thread)
childThread.start()

# Let the child thread wait on the condition variable
time.sleep(1)

cond_var.acquire()
flag = True
cond_var.notify()
cond_var.release()

time.sleep(1)

cond_var.acquire()
flag = False
cond_var.notify()
cond_var.release()

childThread.join()
```

```
print("Program successfully exits")
```

Q What would be the outcome of the program?

(/learn)

A) The program hangs

B) Program successfully exits

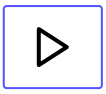
C) Timeout or Error thrown depending on implementation

Submit Answer

Reset Quiz

```
from threading import Thread
from threading import Condition
import time

flag = False
cond_var = Condition()


def child_thread():
    cond_var.acquire()
    while not flag:
        cond_var.wait()

    # enter a useless loop, till flag becomes false
    while flag:
        None


```

```
childThread = Thread(target=child_thread)
childThread.start()

# Let the child thread wait on the condition variable
time.sleep(1)

cond_var.acquire()
flag = True
cond_var.notify()
cond_var.release()
```






Back

Next

With Quiz 2

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/quiz-1__threading-module__python-concurrency-for-senior-engineering-interviews)