daytimeiodelay/daytimeclient.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# daytimeclient.py
# Author: Bob Dondero
#-----------------------------------------------------------------------
# Try this:
# daytimeclient.py time-a.nist.gov 13
#-----------------------------------------------------------------------

import sys
import socket

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 3:
        print(f'usage: python {sys.argv[0]} host port', file=sys.stderr)
        sys.exit(1)

    try:
        host = sys.argv[1]
        port = int(sys.argv[2])

        with socket.socket() as sock:
            sock.connect((host, port))
            with sock.makefile(mode='r', encoding='ascii') as flo:
                for line in flo:
                    print(line, end='')

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

blank (Page 1 of 1)

This page is intentionally blank.

daytimeiodelay/daytimeserver.py (Page 1 of 2)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# daytimeserver.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import os
import sys
import socket
import time

#-------------------------------------------------------------------------

IODELAY = int(os.environ.get('IODELAY', '0'))

#-------------------------------------------------------------------------

def handle_client(sock):

    # Simulate an I/O-bound server.
    time.sleep(IODELAY)

    datetime = time.asctime(time.localtime())
    with sock.makefile(mode='w', encoding='ascii') as flo:
        flo.write(datetime + '\n')
        #flo.flush()

#-------------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print(f'usage: python {sys.argv[0]} port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])

        server_sock = socket.socket()
        print('Opened server socket')
        if os.name != 'nt':
            server_sock.setsockopt(
                socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_sock.bind(('', port))
        print('Bound server socket to port')
        server_sock.listen()
        print('Listening')

        while True:
            try:
                sock, _ = server_sock.accept()
                with sock:
                    print('Accepted connection')
                    print('Opened socket')
                    handle_client(sock)
                print('Closed socket')
            except Exception as ex:
                print(f'{sys.argv[0]}: {ex}', file=sys.stderr)

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------
```

daytimeiodelay/daytimeserver.py (Page 2 of 2)

```


if __name__ == '__main__':
    main()
```

daytimeiodelayp/daytimeserver.py (Page 1 of 2)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# daytimeserver.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import os
import sys
import socket
import time
import multiprocessing

#-------------------------------------------------------------------------

IODELAY = int(os.environ.get('IODELAY', '0'))

#-------------------------------------------------------------------------

def handle_client(sock):

    print('Forked child process')

    try:

        # Simulate an I/O-bound server.
        time.sleep(IODELAY)

        with sock:
            datetime = time.asctime(time.localtime())
            with sock.makefile(mode='w', encoding='ascii') as flo:
                flo.write(datetime + '\n')
                #flo.flush()

        print('Closed socket in child process')
        print('Exiting child process')

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print(f'usage: python {sys.argv[0]} port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])

        server_sock = socket.socket()
        print('Opened server socket')
        if os.name != 'nt':
            server_sock.setsockopt(
                socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_sock.bind(('', port))
        print('Bound server socket to port')
        server_sock.listen()
        print('Listening')

        while True:
            sock, _ = server_sock.accept()
            with sock:
```

daytimeiodelayp/daytimeserver.py (Page 2 of 2)

```
                print('Accepted connection')
                print('Opened socket')
                process = multiprocessing.Process(
                    target=handle_client,
                    args=[sock])
                process.start()
            print('Closed socket in parent process')

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

daytimeiodelayt/daytimeserver.py (Page 1 of 2)

```
#!/usr/bin/env python

#--------------------------------------------------------------------------
# daytimeserver.py
# Author: Bob Dondero
#--------------------------------------------------------------------------

import os
import sys
import socket
import time
import threading

#--------------------------------------------------------------------------

IODELAY = int(os.environ.get('IODELAY', '0'))

#--------------------------------------------------------------------------

class ClientHandlerThread (threading.Thread):

    def __init__(self, sock):
        threading.Thread.__init__(self)
        self._sock = sock

    def run(self):
        print('Spawned child thread')

        # Simulate an I/O-bound server.
        time.sleep(IODELAY)

        with self._sock:
            datetime = time.asctime(time.localtime())
            with self._sock.makefile(mode='w', encoding='ascii') as flo:
                flo.write(datetime + '\n')
                #flo.flush()

        print('Closed socket in child thread')
        print('Exiting child thread')

#--------------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print(f'usage: python {sys.argv[0]} port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])

        server_sock = socket.socket()
        print('Opened server socket')
        if os.name != 'nt':
            server_sock.setsockopt(
                socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_sock.bind(('', port))
        print('Bound server socket to port')
        server_sock.listen()
        print('Listening')

        while True:
            try:
                sock, _ = server_sock.accept()
                print('Accepted connection')
```

daytimeiodelayt/daytimeserver.py (Page 2 of 2)

```
                print('Opened socket')
                client_handler_thread = ClientHandlerThread(sock)
                client_handler_thread.start()
            except Exception as ex:
                print(f'{sys.argv[0]}: {ex}', file=sys.stderr)

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#--------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

daytimecdelay/daytimeserver.py (Page 1 of 2)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# daytimeserver.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import os
import sys
import socket
import time

#-------------------------------------------------------------------------

CDELAY = int(os.environ.get('CDELAY', '0'))

#-------------------------------------------------------------------------

def consume_cpu_time(delay):

    initial_thread_time = time.thread_time()
    while (time.thread_time() - initial_thread_time) < delay:
        pass

#-------------------------------------------------------------------------

def handle_client(sock):

    # Simulate a compute-bound server.
    consume_cpu_time(CDELAY)

    datetime = time.asctime(time.localtime())
    with sock.makefile(mode='w', encoding='ascii') as flo:
        flo.write(datetime + '\n')
        #flo.flush()

#-------------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print(f'usage: python {sys.argv[0]} port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])

        server_sock = socket.socket()
        print('Opened server socket')
        if os.name != 'nt':
            server_sock.setsockopt(
                socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_sock.bind(('', port))
        print('Bound server socket to port')
        server_sock.listen()
        print('Listening')

        while True:
            try:
                sock, _ = server_sock.accept()
                with sock:
                    print('Accepted connection')
                    print('Opened socket')
                    handle_client(sock)
                print('Closed socket')
```

daytimecdelay/daytimeserver.py (Page 2 of 2)

```
            except Exception as ex:
                print(f'{sys.argv[0]}: {ex}', file=sys.stderr)

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

daytimecdelayp/daytimeserver.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# daytimeserver.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import sys
import socket
import time
import multiprocessing

#-----------------------------------------------------------------------

CDELAY = int(os.environ.get('CDELAY', '0'))

#-----------------------------------------------------------------------

def consume_cpu_time(delay):

    initial_thread_time = time.thread_time()
    while (time.thread_time() - initial_thread_time) < delay:
        pass

#-----------------------------------------------------------------------

def handle_client(sock):

    print('Forked child process')

    try:

        # Simulate a compute-bound server.
        consume_cpu_time(CDELAY)

        with sock:
            datetime = time.asctime(time.localtime())
            with sock.makefile(mode='w', encoding='ascii') as flo:
                flo.write(datetime + '\n')
                flo.flush()

        print('Closed socket in child process')
        print('Exiting child process')

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print(f'usage: python {sys.argv[0]} port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])

        server_sock = socket.socket()
        print('Opened server socket')
        if os.name != 'nt':
            server_sock.setsockopt(
                socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

daytimecdelayp/daytimeserver.py (Page 2 of 2)

```
        server_sock.bind(('', port))
        print('Bound server socket to port')
        server_sock.listen()
        print('Listening')

        while True:
            sock, _ = server_sock.accept()
            with sock:
                print('Accepted connection')
                print('Opened socket')
                process = multiprocessing.Process(
                    target=handle_client,
                    args=[sock])
                process.start()
            print('Closed socket in parent process')

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

daytimecdelayt/daytimeserver.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# daytimeserver.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import sys
import socket
import time
import threading

#-----------------------------------------------------------------------

CDELAY = int(os.environ.get('CDELAY', '0'))

#-----------------------------------------------------------------------

def consume_cpu_time(delay):

    initial_thread_time = time.thread_time()
    while (time.thread_time() - initial_thread_time) < delay:
        pass

#-----------------------------------------------------------------------

class ClientHandlerThread (threading.Thread):

    def __init__(self, sock):
        threading.Thread.__init__(self)
        self._sock = sock

    def run(self):
        print('Spawned child thread')

        # Simulate a compute-bound server.
        consume_cpu_time(CDELAY)

        with self._sock:
            datetime = time.asctime(time.localtime())
            with self._sock.makefile(mode='w', encoding='ascii') as flo:
                flo.write(datetime + '\n')
                #flo.flush()

        print('Closed socket in child thread')
        print('Exiting child thread')

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print(f'usage: python {sys.argv[0]} port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])

        server_sock = socket.socket()
        print('Opened server socket')
        if os.name != 'nt':
            server_sock.setsockopt(
                socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_sock.bind(('', port))
```

daytimecdelayt/daytimeserver.py (Page 2 of 2)

```
        print('Bound server socket to port')
        server_sock.listen()
        print('Listening')

        while True:
            try:
                sock, _ = server_sock.accept()
                print('Accepted connection')
                print('Opened socket')
                client_handler_thread = ClientHandlerThread(sock)
                client_handler_thread.start()
            except Exception as ex:
                print(f'{sys.argv[0]}: {ex}', file=sys.stderr)

    except Exception as ex:
        print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
        sys.exit(1)

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

deadlock.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# deadlock.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self, title):
        self._title = title
        self._balance = 0
        self._lock = threading.RLock()

    def transfer_to(self, other, amount):
        self._lock.acquire()
        other._lock.acquire()
        self._balance -= amount
        other._balance += amount
        print(self._title + ': ' + str(self._balance))
        print(other._title + ': ' + str(other._balance))
        other._lock.release()
        self._lock.release()

#-----------------------------------------------------------------------

class AliceToBobThread (threading.Thread):

    def __init__(self, alice_acct, bob_acct):
        threading.Thread.__init__(self)
        self._alice_acct = alice_acct
        self._bob_acct = bob_acct

    def run(self):
        for _ in range(1000):
            self._alice_acct.transfer_to(self._bob_acct, 1)

#-----------------------------------------------------------------------

class BobToAliceThread (threading.Thread):

    def __init__(self, bob_acct, alice_acct):
        threading.Thread.__init__(self)
        self._bob_acct = bob_acct
        self._alice_acct = alice_acct

    def run(self):
        for _ in range(1000):
            self._bob_acct.transfer_to(self._alice_acct, 1)

#-----------------------------------------------------------------------

def main():

    alice_acct = BankAcct('Alice')
    bob_acct = BankAcct('Bob')

    alice_to_bob_thread = AliceToBobThread(alice_acct, bob_acct)
    bob_to_alice_thread = BobToAliceThread(bob_acct, alice_acct)

    alice_to_bob_thread.start()
```

deadlock.py (Page 2 of 2)

```
    bob_to_alice_thread.start()

    alice_to_bob_thread.join()
    bob_to_alice_thread.join()

    print('Finished')

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

deadlockw.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# deadlockw.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self, title):
        self._title = title
        self._balance = 0
        self._lock = threading.RLock()

    def transfer_to(self, other, amount):
        with self._lock:
            with other._lock:
                self._balance -= amount
                other._balance += amount
                print(self._title + ': ' + str(self._balance))
                print(other._title + ': ' + str(other._balance))

#-----------------------------------------------------------------------

class AliceToBobThread (threading.Thread):

    def __init__(self, alice_acct, bob_acct):
        threading.Thread.__init__(self)
        self._alice_acct = alice_acct
        self._bob_acct = bob_acct

    def run(self):
        for _ in range(1000):
            self._alice_acct.transfer_to(self._bob_acct, 1)

#-----------------------------------------------------------------------

class BobToAliceThread (threading.Thread):

    def __init__(self, bob_acct, alice_acct):
        threading.Thread.__init__(self)
        self._bob_acct = bob_acct
        self._alice_acct = alice_acct

    def run(self):
        for _ in range(1000):
            self._bob_acct.transfer_to(self._alice_acct, 1)

#-----------------------------------------------------------------------

def main():

    alice_acct = BankAcct('Alice')
    bob_acct = BankAcct('Bob')

    alice_to_bob_thread = AliceToBobThread(alice_acct, bob_acct)
    bob_to_alice_thread = BobToAliceThread(bob_acct, alice_acct)

    alice_to_bob_thread.start()
    bob_to_alice_thread.start()

```

deadlockw.py (Page 2 of 2)

```
    alice_to_bob_thread.join()
    bob_to_alice_thread.join()

    print('Finished')

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

nodeadlock.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# nodeadlock.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self, title, sequence_num):
        self._title = title
        self._balance = 0
        self._sequence_num = sequence_num
        self._lock = threading.RLock()

    def transfer_to(self, other, amount):
        if self._sequence_num < other._sequence_num:
            self._lock.acquire()
            other._lock.acquire()
            self._balance -= amount
            other._balance += amount
            print(self._title + ': ' + str(self._balance))
            print(other._title + ': ' + str(other._balance))
            other._lock.release()
            self._lock.release()
        else:
            other._lock.acquire()
            self._lock.acquire()
            self._balance -= amount
            other._balance += amount
            print(self._title + ': ' + str(self._balance))
            print(other._title + ': ' + str(other._balance))
            self._lock.release()
            other._lock.release()

#-----------------------------------------------------------------------

class AliceToBobThread (threading.Thread):

    def __init__(self, alice_acct, bob_acct):
        threading.Thread.__init__(self)
        self._alice_acct = alice_acct
        self._bob_acct = bob_acct

    def run(self):
        for _ in range(1000):
            self._alice_acct.transfer_to(self._bob_acct, 1)

#-----------------------------------------------------------------------

class BobToAliceThread (threading.Thread):

    def __init__(self, bob_acct, alice_acct):
        threading.Thread.__init__(self)
        self._bob_acct = bob_acct
        self._alice_acct = alice_acct

    def run(self):
        for _ in range(1000):
            self._bob_acct.transfer_to(self._alice_acct, 1)

```

nodeadlock.py (Page 2 of 2)

```
#-----------------------------------------------------------------------

def main():

    alice_acct = BankAcct('Alice', 1)
    bob_acct = BankAcct('Bob', 2)

    alice_to_bob_thread = AliceToBobThread(alice_acct, bob_acct)
    bob_to_alice_thread = BobToAliceThread(bob_acct, alice_acct)

    alice_to_bob_thread.start()
    bob_to_alice_thread.start()

    alice_to_bob_thread.join()
    bob_to_alice_thread.join()

    print('Finished')

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

nodeadlockw.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# nodeadlockw.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import threading

#-----------------------------------------------------------------------

class BankAcct:

    def __init__(self, title, sequence_num):
        self._title = title
        self._balance = 0
        self._sequence_num = sequence_num
        self._lock = threading.RLock()

    def transfer_to(self, other, amount):
        if self._sequence_num < other._sequence_num:
            with self._lock:
                with other._lock:
                    self._balance -= amount
                    other._balance += amount
                    print(self._title + ': ' + str(self._balance))
                    print(other._title + ': ' + str(other._balance))
        else:
            with other._lock:
                with self._lock:
                    self._balance -= amount
                    other._balance += amount
                    print(self._title + ': ' + str(self._balance))
                    print(other._title + ': ' + str(other._balance))

#-----------------------------------------------------------------------

class AliceToBobThread (threading.Thread):

    def __init__(self, alice_acct, bob_acct):
        threading.Thread.__init__(self)
        self._alice_acct = alice_acct
        self._bob_acct = bob_acct

    def run(self):
        for _ in range(1000):
            self._alice_acct.transfer_to(self._bob_acct, 1)

#-----------------------------------------------------------------------

class BobToAliceThread (threading.Thread):

    def __init__(self, bob_acct, alice_acct):
        threading.Thread.__init__(self)
        self._bob_acct = bob_acct
        self._alice_acct = alice_acct

    def run(self):
        for _ in range(1000):
            self._bob_acct.transfer_to(self._alice_acct, 1)

#-----------------------------------------------------------------------

def main():

```

nodeadlockw.py (Page 2 of 2)

```
    alice_acct = BankAcct('Alice', 1)
    bob_acct = BankAcct('Bob', 2)

    alice_to_bob_thread = AliceToBobThread(alice_acct, bob_acct)
    bob_to_alice_thread = BobToAliceThread(bob_acct, alice_acct)

    alice_to_bob_thread.start()
    bob_to_alice_thread.start()

    alice_to_bob_thread.join()
    bob_to_alice_thread.join()

    print('Finished')

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()
```