1. Każdy proces składa się z wątków. Wątki realizują podzadania danego programu/procesu.
2. Każdy proces musi posiadać chociaż jeden wątek
3. Wątek uruchamiany na początku działania programu
4. Każdy proces ma przydzielony na własność pewien obszar pamięci i obszar ten nie jest współdzielony z innym procesem. Wątki należące do jednego procesu współdzielą między sobą zasoby i mogą wymieniać informacje.
5. Żeby przyspieszyć działanie programu
6. Musi mieć więcej niż jeden rdzeń.
7. Może wykonywać się więcej procesów ale będą one wykonywane współbieżnie a nie równolegle.
8. GIL – Global Interpreter Lock –uniemożliwia wykonywanie kilku wątków równocześnie.
9. Przyspiesza czas wykonania programu jednowątkowego.
10. Data race - występuje, gdy pierwszy wątek modyfikuje pewien obszar pamięci, a drugi chce równocześnie coś z niego odczytać.

Race condition - występuje, gdy wątki wykonują na danym zasobie pamięci operacje w nieustalonej kolejności.

1. Threading
2. import threading

def nazwa\_funkcji():

pass

def main():

t1=threading.Thread(target=nazwa\_funkcji)

t1.start()

t1.join()

if \_\_name\_\_ == ’’\_\_main\_\_’’:

main()

1. t1=threading.Thread(target=nazwa\_funkcji, args=[arg1, arg2, arg3])
2. Można zastosować locka. Lock zabezpiecza dany obszar kodu przed równoczesnym wykonywaniem przez dwa wątki – blokowanie dostepu do danej sekcji kodu.
3. import threading

lock = threading.Lock()

def funkcja():

lock.acquire()

for i in range(100):

value +=1

lock.release()

def main():

threads =[ threading.Thread(target=funkcja), threading.Thread(target=funkcja)]

for thread in threads:

thread.start()

for thread in threads:

thread.join()

if \_\_name\_\_ == ‘’\_\_main\_\_’’:

main()

1. Problem deadlock.Wątek A czeka aż wątek B skończy swoją operację a wątek B czeka aż wątek A zakończy akcję.
2. Występuje on, gdy dwa procesy, aby uniknąć deadlock-a zatrzymują wykonywanie kodu, aby dać możliwość innym wątkom na wykonanie się. Aby ułatwić zrozumienie, wyobraź sobie sytuację, gdy dwie osoby na korytarzu, aby minąć siebie, wybierają ciągle tą samą trasę, kończąc na ciągłej wzajemnej blokadzie.
3. from multiprocessing import Pool

def funkcja1():

pass

def fukcja2():

pass

if \_\_name\_\_== ’’\_\_main\_\_’’:

pool = Pool(processes=2)

start = time.time()

p1 = pool.apply\_async(funkcja1)

p2 = pool.apply\_async(funkcja2)

pool.close()

pool.join()