Dokumentálás és mérések

Első program: Szöveg adatok kigyűjtése. Leghosszabb, legrövidebb, leggyakoribb szó, átlagos szó hossz és szó keresés egy fájlban.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Többszálon történő futtatás: A Threadpoolexecutor segítségével létrehozunk egy thread pool-t A thread pool megkönnyíti több thread létrehozását és join-olását. A végrehajtott task-ok egy future objektumot adnak vissza, amelyek később, a task befejezésekor kapnak értéket. A Threadpoolexecutor az executor osztályt egészíti ki, amely rendelkezik 3 függvénnyel, amelyek a threadek kezelésére szolgálnak: submit(), map() és shutdown(). A submit()-nak átadva a függvényt, a Threadpoolexecutor elindítja a szál végrehajtását, és a kapott future objektumot a results listában tároljuk. Miután elindultak a szálak, a concurrent.futures.as\_completed() függvénnyel adjuk vissza a future objektumokat, akkor, ha biztosan befejeződött a végrehajtás. A future objektum értékét pedig a result() függvénnyel kapjuk meg.

A (c)Pythonban GIL (Global Interpreter Lock) miatt egy processzoron egyszerre csak egy szál futhat, ezért a szálak menedzselési költsége miatt hosszabb futási idő, mint a szekvenciális változat.

Alternatívaként lehet használni a ProcessPoolExecutor-t, amely szálak helyett processzeket használ. Az öt függvény számára 5 processz-t indítunk és ezzel meglehetősen csökken a program futási ideje.

Text

Description automatically generated

Második Python program: Képek letöltése az internetről és grayscale-t szerkeszteni rájuk a pillow, aiohttp és aiofiles modulok segítségével.

Szekvenciális változat:

Text

Description automatically generated

A requests modul get függvényével a megadott endpoint url címén lévő adatokat letöltjük és a contents() függvény segítségével a nyers biteket eltároljuk, majd ezt kiírjuk egy jpg fájlba. Ezután a pillow modul Image objektumát használva megnyitjuk és megszerkesztjük a képet, majd visszaírjuk.

A konkurens változat:

Text

Description automatically generated

Az async kulcsszóval deklarálhatunk coroutine-okat és az await kulcsszóval várunk a coroutine-ok befejeződésére. Az await kulcsszóval lehet várni coroutine, task és future objektumokra. Az event loop futtat aszinkron taskokat, callback-eket, hálózati IO operációkat és szubprocesszeket futtat.

Az asyncio.get\_event\_loop() visszaadja a jelenlegi event loop-ot, ha nem létezik, akkor létrehoz egyet.

loop.run\_until\_complete() addig fut a megadott main függvény, ameddig az összes future típusú objektum be nem fejeződik.

asyncio.Queue: get() Kiveszi az elemet a sorból.

join() föggvény hívásával blokkol, ameddig az összes elemet ki nem vették a sorból.

task\_done(): Jelzi, hogy a task befejeződött.

producers egy coroutine-okat tartalmazó lista.

asyncio.ensure\_future() coroutinok esetén a create\_task()- al létrehozza taskokat.

asyncio.gather: Futtatja az összes termelőt, az exception-öket is visszaadja.

Miután a consumerek befejezték a képek szerkesztését le kell őket állítani a cancel() függvénnyel.

Harmadik program: Sudoku rekurzív megoldó probram backtracking algoritmussal.

Text

Description automatically generated

Text

Description automatically generated with medium confidence

A sudoku rekurzív megoldásában a párhuzamosítás a master szálon kezdődik, egy osztott változóval, a sudoku táblával. A solve függvényen belül először meghívunk egy másik függvényt, ami addig megy, amíg nem nullát nem talál a táblában és mellékhatásként módosítja a pointerek által tárolt változók értékét. Ha már nem talált üres mezőt, akkor vége az algoritmusnak. Ha talált üres mezőt, akkor végigmegyünk a számokon 1-től 9-ig és megnézzük, hogy ilyen szám található a sorban, az oszlopban vagy a négyzetben. Ha nincs ilyen szám, akkor létrehoz egy taskot, amely másolatot készít a tábláról az új számot az üres mezőbe helyezve, majd meghívja újra a solve függvényt. Ha kipróbáltuk az összes számot, megvárjuk, hogy az adott szinten indított taskok visszatérjenek és megyünk a következő szintre.

default(none): Kötelezően meg kell adni, hogy milyen kikötések legyenek a változókra.

firstprivate(): A threadnek legyen privát másolata a változóról és inicializálja a korábbi értéket használva.

shared(): Minden thread rendelkezzen a változóval.

taskwait: Megvárja az összes task befejezését mielőtt visszaadná az értéket (Megy a következő mezőre)

Negyedik program: Szó előfordulásának megszámlálása szövegben.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A szöveget először beolvassuk egy string tömbbe, majd a tömb mérete alapján felosztjuk a tömböt a szálak számára.