Í þessu verkefni ætlum við að gera tvo hluti. Fyrst ætlum við að búa til tól til að mæla keyrslutíma í kóða sem við skrifum, seinna prófa tólið á nokkur dæmi til að fá tilfinningu fyrir því hvernig mismunandi gagnagrindur geta haft stór áhrif á keyrslutíma þegar unnið er með mjög mikið af gögnum. Ef unnið vel getur þetta tól verið notað í mörgum verkefnum námskeiðsins til þess að prófa hraða gagnagrinda sem við komum til með að læra/gera.

Stopwatch

Búðu til tólið(klasa) Stopwatch sem tekur tímann á keyrslu ákveðins kóðabúts.
Dæmi um aðgerðir(föll) sem skeiðklukkan getur haft (* - aðgerðir sem eru nauðsynlegar):

- *StartTimer
- *GetDuration
- *StopTimer
- PauseTimer (möguleiki að sleppa StopTimer ef þetta er útfært)
- GetPausedDuration
- GetNumberOfPauses
- GetTimerHistory (klukkan heldur utan um fyrri mælingar sem hafa verið stöðvaðar og prentar út lista af öllum mælingum sem hafa verið gerður á þessu instance af klukkunni)

Nokkrir hlutir til að hafa í huga

1. Finna leiðir til að framkvæma tímamælingar í python
    - Td hægt að nota datetime/timedelta eða time sem fylga með python
    - Finna documentation eða dæmi á netinu um hvernig væri hægt að nota þessi library

2. Hanna klasa sem heldur utan um tímamælingar
    - Hér þarf að ákveða hvaða aðgerðir hægt er að gera
    - Ákveða hvað gerist þegar hver aðgerð er keyrð
    - Hvernig styðja þessar aðgerðir það að hefja, enda og fá upplýsingar um tímamælingu?

3. Hanna og skilgreina útreikninga til að setja tíma fram á skýran hátt
    - Ef tími kemur inn sem ein tala, hvað táknar sú tala
    - Hvernig er hægt að setja hana fram á skýrari og læsilegri hátt?
        - Klst, mín, sek, hundraðshlutar, millisekúndur, míkrósek
        - Er þörf á þessu öllu, eða misjafnt eftir tölum/notkun?
    - Setjið fram (handskrifað) þá útreikninga sem þarf að framkvæma til að þetta sé mögulegt.

Mælingar
Nú ættir þú að vera kominn með virka lausn sem hægt er að nota til þess að mæla hve langan tíma einhver forritsaðgerð tekur.

Flottast væri að flytja tímamælingaklasann í sérskrá sem inniheldur ekkert annað. Í skránni þar sem nota á mælingarnar er þá hægt að gera **import** *setningu til að vísa í klasann þar og þá er engum kóða blandað saman, þ.e. tímamælingakóða og kóða aðgerðanna sjálfra.*

Aðgerðir til tímamælinga
Langbest er að búa til föll (eða jafnvel klasa) í kringum þessar aðgerðir, svo að þegar tímamælingin fer fram sé bara kallað á upphafsfall á tímamælingaklasa, síðan á fallið sem á að mæla í einni línu og að lokum á lokafallið í tímamælingunni. ***Venja sig á snyrtilega forritun!***

1. **Stak sótt í lista**
   Búið til lista í python (`some_lis = []`)
   Setjið ákveðinn fjölda staka í listann (10, 100, 1000, 10000, etc.)
   Lúppa i frá 0 upp í 9 (eða 99 eða 999) og setja random stak í lis[i].
   Sækið ákveðið stak í listann (some_var = lis[index]).
   Það má búa til index með random. Það getur líka verið sniðugt að framkvæma aðgerðina oftar, t.d. 1000 sinnum og tímamæla það, til að fá jafnara meðaltal. Ein slík aðgerð tekur svo ótrúlega stuttan tíma að það er óvíst að það sé að marka.
   Mælið hve langan tíma það tekur að sækja stak (eða 1000 stök).
   Spurning: *Hefur áhrif á þennan tíma hversu mörg stök eru í listanum?*
   Hversu vel er hægt að einangra aðgerðina svo að aðrar aðgerðir hafi ekki áhrif á tímamælinguna?

2. **Stak fundið í lista**
   Setjið aftur stök í lista og keyrið núna setninguna:
   ```
   if some_value in lis:
       #Gera eitthvað mjög lítið og einfalt (a = 13 * 17)
       pass
   ```
   Prófið þetta með misstórum listum og reynið að fá jafna og góða tímamælingu.
   Spurning: *Hefur áhrif á þennan tíma hversu mörg stök eru í listanum?*

3. **Lykill fundinn í dictionary**
   Búið núna til dictionary og setjið lykla inn í það á sama hátt og þið settuð áður stök inn í lista. Setjið bara einhver slembigildi inn sem gildin.
   Dæmi: `my_dictionary[some_value] = some_random_string`
   Prófið núna að keyra:
   ```
   if some_value in my_dictionary:
       #Gera eitthvað mjög lítið og einfalt
       pass
   ```

Prófið þetta með misstórum listum og reynið að fá jafna og góða tímamælingu.
Spurning: *Hefur áhrif á þennan tíma hversu mörg stök eru í listanum?*

4. **Insert vs Append á lista**
   Hvort er hraðara að fylla til lista með (100, 1000, 10000 etc.) stök með insert eða append
   Prófið að gera insert aðgerðina á aðra staði heldur en bara fremst í listann.
   Spurning: *Er munur á því hvort insert aðgerðin sé gerð í miðjan lista, fremst eða aftast?*

Flóknari aðgerðir og innbyggðar aðgerðir

Hér er nokkrar fleiri innbyggðar aðgerðir á python listum sem hægt væri að prófa sig á.

- **remove**
- **sort**
- **count**
- **reverse**
- **copy**
- **Clear**
- **Pop**
- **Del** syntax: del lis[i]

Spurningar sem hægt er að spyrja sig fyrir þessi föll:

1. *Hefur stærð listans áhrif á keryslutíma?*
2. *Ef fallið tekur inn index sem parameter (td del), skiptir máli hver indexinn er? (hvort hann sé aftarlega í listanum eða framarlega)*
3. *Eru einhverjar aðgerðir sem eru alltaf jafn fljótar?*
4. *Eru einhverjar aðgerðir sem eru næstum alltaf jafn fljótar en stundum hægar?*