Introduction to Computing: Tutorial 7

BSDS I Year, 2026–2027

November 19, 2026

Pandas

(adapted from `https://pynative.com/python-pandas-exercise/`)

0. Download `autodata.zip` from the course page, and unzip it. You should get 4 CSV files, with more or less self-explanatory names.
1. Write the correct `pd.read_csv()` statements for the 4 files, and save the results in `df1`, `df2`, etc. Verify that the shape attribute of all DataFrames are the same, and what you expect.
2. Print the first 3 and last 3 rows of the DataFrame.
3. Find which rows and columns contain missing / invalid values.
4. Print the company name and price of the most expensive car(s).
5. Print the distinct body-styles available in Toyota cars.
6. Print the number of car models available for each company.

Revision: `timeit.timeit()`

Run the following code line-by-line using Python / IPython (recommended).

```
from timeit import *
x = 0
timeit(stmt="print(x)", setup="x += 1", number=3, globals=globals())
timeit(stmt="print(x); x+=1", setup="x = 1", number=3, globals=globals())

abcd = 10
timeit(stmt="print(abcd)", number=3, globals=globals())
timeit(stmt="print(abcd); abcd+=1", number=3, globals=globals())
timeit(stmt="print(abcd)", setup="abcd += 1", number=3, globals=globals())
timeit(stmt="print(abcd); abcd = abcd + 1", setup="abcd = 10", number=3)
timeit(stmt="print(abcd); abcd = abcd + 1", setup="global abcd", number=3)
timeit(stmt="print(abcd); abcd = abcd + 1", setup="global abcd", number=3, globals=globals())
repeat(stmt="print(abcd)", setup="global abcd; abcd += 1", number=3, globals=globals())
```

Which of the above lines works, which do not, and why?

[See material added to the Misc. topics slides for hints.]

Asymptotic time complexity

1. It is reasonable to assume that the functions that we deal with in the context of time complexity are non-negative, and monotonic. Formally verify whether the claims on Slide 16 are correct, under these assumptions about $f(n), g(n)$.

   - If $g(n) \in O(f(n))$, then $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ exists, and may be 0.
   - If $g(n) \in \Omega(f(n))$, then $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ is $c > 0$ or ∞.
   - If $g(n) \in \Theta(f(n))$, then $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ is $c > 0$.
   - $O(1) \subset O(\log n) \subset O(n^k)(k < 1) \subset O(n) \subset O(n^l)(l > 1) \subset O(a^n)(a > 1)$

2. Look carefully at the call to `timeit()` around line 85 of `sorting.py` (available from the course page).

   (a) The `stmt` parameter is `"A = rng.uniform(-1e6, 1e6, N); f(A)"`. This means that the different algorithms that we are timing are working on different input arrays. Why is this less than ideal?

   (b) Experimentally check what happens when one single array `A` is generated after line 83 for a given value of `N`, and all sorting algorithms are given `A` as input for a fair comparison, i.e., the `stmt` parameter in the `timeit()` call is changed to simply `"f(A)"`.

   (c) Think about how you would fix the observed problem.

3. You are given N binary strings $\{s_1, s_2, \ldots, s_N\}$ (i.e., the strings consisting of the characters 0 and 1 only). Each string is exactly m characters long. Your task is to determine whether the given collection contains at least one duplicate, i.e., whether there exists $i \neq j$ such that $s_i = s_j$.

   (a) What is the smallest value of N for which the answer to the above question will always be YES?

   (b) If the value of N is such that the answer is not guaranteed to be YES, how would you determine whether the set contains any duplicates? Analyse the time complexity of your algorithm.

   [At the moment, you need not worry about the efficiency of your algorithm. Focus on doing the complexity analysis carefully.]