- PS4: complete version is on Quercus
- TT3: more on that later... we <u>are</u> open to discussion, but today it would take too much of the time we have!

Last time...

- is_palindrome(s) : <u>WC</u> runtime is $\Theta(n)$

-
  ```
  def pal_prefix(s: str) -> int:
      ''' Return length of a longest prefix of s
      that is a palindrome.'''
      for p in range(len(s), 0, -1):  # p=n,n-1,...,1
          if is_palindrome(s[0:p]):
              return p
  ```

- $WC_{pal\_prefix}(n) = ?$
- **Upper bound:** Goal: show $RT(s) \in O(\_\_)$ for all inputs s of size n

  Approach:
  - overestimate, ignore early termination
  - simplify as we go
  - danger: overcount — upper bound is not tight — check with lower bound

  - loop body takes ~~time~~ $\leq p \leq n = len(s)$
  - # iterations $\leq n$
  - total $\leq n^2$ $\Rightarrow WC(n) \in O(n^2)$
- **Lower bound:** Goal: show $RT(s) \in \Omega(\_\_)$ for some input s of size n

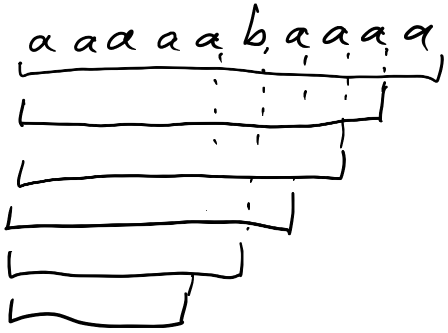
  Approach: - simplify as we go

- underestimate
- danger: leave out too much...
  try to match upper bound

Here, challenge:
- s is not a palindrome — and s's prefixes are not palindromes: for loop makes many iterations
  → then, is_palindrome runs faster
- strings bad ~~for~~ for is_pal (s <u>is</u> a palindrome) are <u>good</u> for pal_prefix: for loop stops early...

<u>Insight</u>: want input s that is <u>almost</u> a palindrome

$$s = \underbrace{aa\cdots a}_{\lceil \frac{n}{2} \rceil} b \underbrace{aa\cdots a}_{\lfloor \frac{n}{2} \rfloor - 1}$$



is_pal : F — time $\frac{n}{2} = n - \lceil \frac{n}{2} \rceil$

is_pal : F — time $n/2 - 1$

F $n/2 - 2$

F $n/2 - 3$

F $n/2 - 4$

T $n/2$

Total time:

$$\left\lceil \frac{n}{2} \right\rceil + \sum_{p=\lceil \frac{n}{2} \rceil + 1}^{n} \left(p - \left\lceil \frac{n}{2} \right\rceil\right) = \frac{n}{2} + \frac{\left(\frac{n}{2}\right)\left(\frac{n}{2} - 1\right)}{2}$$

(last call to is_pal on prefix that does not contain b)

$$\sum_{j=1}^{n - \lceil \frac{n}{2} \rceil} j$$

$j = p - \lceil \frac{n}{2} \rceil$

$\geq \frac{n^2}{8}$ (approx.)

$\Downarrow$

$WC(n) \in \Omega(n^2)$

Last example...

```
def twisty(n: int) -> None:
    while n > 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = 2 * n - 2
```

$\}$ const.

runtime $\in \Theta(\underline{\text{\# iterations}})$

insight: trace multiple iterations

$$\begin{tikzcd}
n \arrow[r, "\text{odd}"] \arrow[rd, "\text{even}"'] & 2n-2 \arrow[r, "\text{even}"] & \frac{2n-2}{2} = \underline{n-1} \\
 & \frac{n}{2} \arrow[r, "\text{odd}"] \arrow[rd, "\text{even}"'] & 2\frac{n}{2}-2 = \underline{n-2} \\
 & & n/4 = \underline{n/4}
\end{tikzcd}$$