2.2 Representation Invariants

- Writing methods sensibly by documenting rules and limits

REPRESENTATION INVARIANT:

- Property of the instance attributes that *every class must satisfy*
- Written in the docstring of a class & underneath its attributes
- Written as boolean expressions (can be checked in the program)
- Type annotation is a representation invariant

```
class Tweet:
    """A tweet, like in Twitter.

    === Attributes ===
    content: the contents of the tweet.
    userid: the id of the user who wrote the tweet.
    created_at: the date the tweet was written.
    likes: the number of likes this tweet has received.

    === Representation Invariants ===
    - len(self.content) <= 280
    """
    # Attribute types
    content: str
    userid: str
    created_at: date
    likes: int
```

ENFORCING REPRESENTATION INVARIANTS:

- Precondition:
  - Right when method is called : assume all representation invariants have been

satisfied
- Postcondition :
  - Right before method returns : ensure all representation invariants are satisfied

- Initializer method:
  - Doesn't have preconditions on the attributes (*haven't been created yet*)

Options to enforce:
- Adding code that adds this condition
- Writing a precondition of the docstring

ZEN OF PYTHON:

"Explicit is better than implicit"

STRATEGIES:
1. **Use preconditions:**
   - If your work isn't too intuitive your user will likely break these preconditions
2. **Ignore bad inputs:**
   - In between strategy but then the user might get confused if your code does nothing after you input smth
3. **Fix bad inputs:**
   - More wholesome but more work and code can get pretty complex...