CSC 453 Milestones 2 (Partial Scanner) and 3 (Full Scanner)

Goal:

These milestones are writing a scanner for Tau.

Specifications

1. The scanner is a class called `Scanner`.
2. The stubbed `Scanner` class is given in `scanner.py`.
3. The classes for tokens and coordinates is given in `tau/tokens.py`, which is provided for you. You MUST NOT EDIT this file.
4. The syntax error reporting is done by throwing a `tau.error.Scanerror`, which is defined in `tau/error.py`. You MUST NOT EDIT this file.
5. This milestone will not prescribe any particular error message for any particular error. You are free to choose your own error messages.

The `Scanner` class should implement the `Iterable[Token]` interface:

~~~
```python
class Scanner():
    def __init__(self, input: str):
        ...
    def __iter__(self) -> Iterator[Token]:
        ...
```
~~~

Tau Language Tokens

1. Identifiers are sequences of letters and digits that start with a letter. ('ID'). Letters are defined by Python's `string.ascii_letters` and digits are defined by Python's `string.digits`.
2. Tau includes many keywords. Tokens returned for keywords will have the keyword as both the `kind` and the `value`.
3. Integer literals are sequences of digits. ('INT')
4. Tau's punctuation and operators are given in `tau/tokens.py`.
5. Skip over comments. Comments start with `//` and go to the end of the line.
6. Skip over whitespace. Whitespace is defined by Python's `string.whitespace`.
7. The scanner should return `Token("EOF", "", span)` when it reaches the end of the input string, where `span` is created with the coordinates where the end was reached for both beginning and end.

Milestone 2 subset

Milestone 2 requires that you only handle the following:

- Scan identifiers
- Scan integer literals
- Skip whitespace
- Skip comments
- Return `Token("EOF", "", span)` when the end of the input is reached.
- Compute correct spans for tokens.

Milestone 3 full scanner

Milestone 3 requires that you handle the entire language.

Restrictions

1. You may not use regular expressions to implement the scanner.

2. You may not use any external libraries to implement the scanner.
3. The intention is that you hand-code the scanner. Please refrain from using shortcuts. If you have questions, please ask.

Standard Requirements

Your program must meet all the requirements outlined in the common requirements document.