TEST

CSM 61A

August 30th - September 3rd, 2021

Exceptions allow you to interrupt the normal flow of execution of a program in the case of an error or exceptional circumstance

Exceptions are any objects that inherit from the `BaseException` class. To raise an exception (and interrupt the code), use the **`raise`** statement.

```
>>> raise Exception('An error occurred')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
Exception: an error occurred
```

You can also use the `assert` statement to raise an `AssertionError`.

```
>>> assert 4 > 5
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AssertionError
```

However, your code doesn't need to fully crash when an exception is raised. We can "handle" exceptions with a **`try`**-**`except`** block in the following format.

```
try:
    <try suite>
except <exception class> as <name>:
    <except suite>
...
```

We can have as many **`except`** clauses as we would like. This is the corresponding behavior when running this block:

1. Python runs the **`try`** suite.
2. If it encounters an Exception during this, it interrupts the **`try`** suite.
3. Python finds the first **`except`** block which corresponds to the class of the exception.
    - If no such **`except`** block is found, the exception is not handled.

4. If Python can find a corresponding block, the exception object is bound to `<name>`, and the **`except`** block is run.

Also note that if we just use **`except`** by itself, it just catches any possible Exception. Generally, this is bad practice and shouldn't be used.

Here is an example of how we can handle exceptions.

```
>>> try:
        positions = {'peter': 'SCM', 'cyrus': 'SCM'}
        x = positions['josh']
    except KeyError as e:
        print('handling a', type(e))
        x = 'Coord'
handling a <class 'KeyError'>
>>> x
'Coord'
```