3.3 Exceptions

ALTERNATIVES:

- **Fail Silently:**
  - "No return value" as a sign that smth bad happened.
  - Approach doesn't work for all methods as some don't return anything when they are working fine
- **Raise a User-defined Expression:**
  - Raise an error when smth goes wrong, w/o giving away implementation
  - Defining and raising our own errors enables us to give descriptive messages to the user when they have used our own class incorrectly
  - We can customize the error message by overriding the inherited _ _ str _ _ method

```
class EmptyStackError(Exception):
    """Exception raised when calling pop on an empty stack."""

    def __str__(self) -> str:
        """Return a string representation of this error."""
        return 'You called pop on an empty stack. :('
```

```
    def pop(self) -> Any:
        """Remove and return the element at the top of this stack.

        Raise an EmptyStackError if this stack is empty.

        >>> s = Stack()
        >>> s.push('hello')
        >>> s.push('goodbye')
        >>> s.pop()
        'goodbye'
        """
        if self.is_empty():
            raise EmptyStackError
        else:
            return self._items.pop()
```

```
>>> s = Stack()
>>> s.pop()
Traceback (most recent call last):
  File "<input>", line 1, in <module>
  File "...", line 60, in pop
    raise EmptyStackError
EmptyStackError: You called pop on an empty stack. :(
```

EXCEPTIONS INTERRUPT THE NORMAL FLOW OF CONTROL:

- **When function is called:**
  - Program pushes stack frame
- **When function returns/reaches end:**
  - Program pops current top stack frame
- **When exceptions is raised:**
  - Function ends immediately & pops current top stack frame
  - Sends exception back to the caller
- Exceptions for error handling: Taking responsibility for handling and catching exceptions!

```
if __name__ == '__main__':
    option = 'y'
    while option == 'y':
        value = input('Give me an integer to check if it is a divisor of 42: ')
        try:
            is_divisor = (42 % int(value) == 0)
            print(is_divisor)
        except ZeroDivisionError:
            print("Uh-oh, invalid input: 0 cannot be a divisor of any number!")
        except ValueError:
            print("Type mismatch, expecting an integer!")
        finally:
            print("Now let's try another number...")
        option = input('Would you like to continue (y/n): ')
```