CptS 322- Software Engineering Principles I

Coverage-Based Testing

Instructor: Sakire Arslan Ay

Fall 2023

WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

Recall: Kinds of testing

- **unit testing**: Does each unit work as specified?

- **integration testing**: Do the units work when put together?

- **system testing**: Does the system work as a whole?

Test Effectiveness

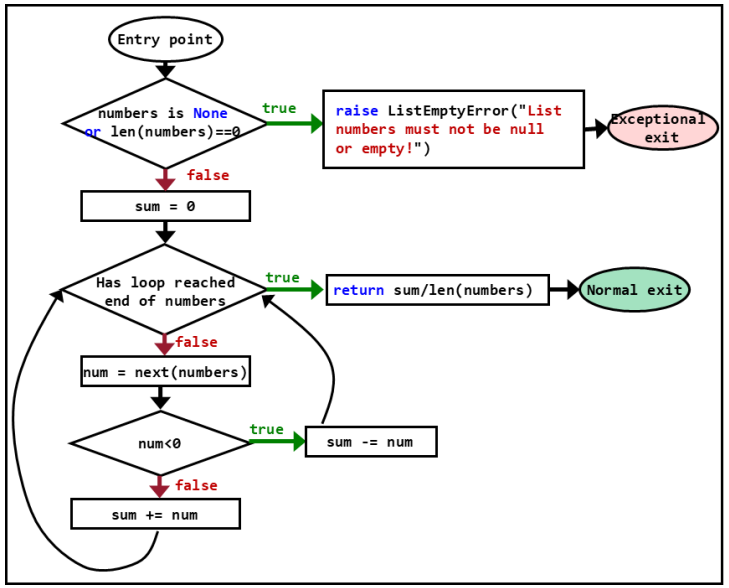
- **Ratio of detected defects (bugs) is the best effectiveness metric!**

- **Problem**
  - The set of defects (bugs) is unknowable.
- **Solution**
  - Use a proxy metric, for example code coverage.

Code Coverage

- Code coverage is a metric that can help you understand how much of your source is tested.
  - help you assess the quality of your test suite

- The common metrics include:
  - **Statement coverage:** how many of the statements in the program have been executed.
  - **Decision (branch) coverage:** how many of the branches of the control structures (if statements for instance) have been executed.
  - **Condition coverage:** how many of the boolean sub-expressions have been tested for a true and a false value.



Structural Code Coverage: motivating example

- **Average of the absolute values of an array of doubles**

```
class ListEmptyError(Exception):
        """Raised when the input value is too small"""
        pass

class Avg():
    """Compute the average of the absolute values of an array of doubles """
    def avgAbs(self,numbers):
```

No peeking ☺

What tests should we write for this method?

- Black-box tests

Structural Code Coverage: motivating example

- **Average of the absolute values of an array of doubles**

```
class ListEmptyError(Exception):
        """Raised when the input value is too small"""
        pass

class Avg():
    """Compute the average of the absolute values of an array of doubles """
    def avgAbs(self,numbers):
        # We expect the array to be non-null and non-empty
        if (numbers is None or len(numbers) == 0):
            raise ListEmptyError("List numbers must not be null or empty!");

        sum = 0
        for d in numbers:
            if (d < 0):
                sum -= d
            else:
                sum += d

        return sum/len(numbers)
```

What tests should we write for this method?

- Black-box tests
- White-box tests

Structural code coverage: motivating example

- Example code is available on Canvas:
  - https://wsu.instructure.com/courses/1650001/pages/software-testing

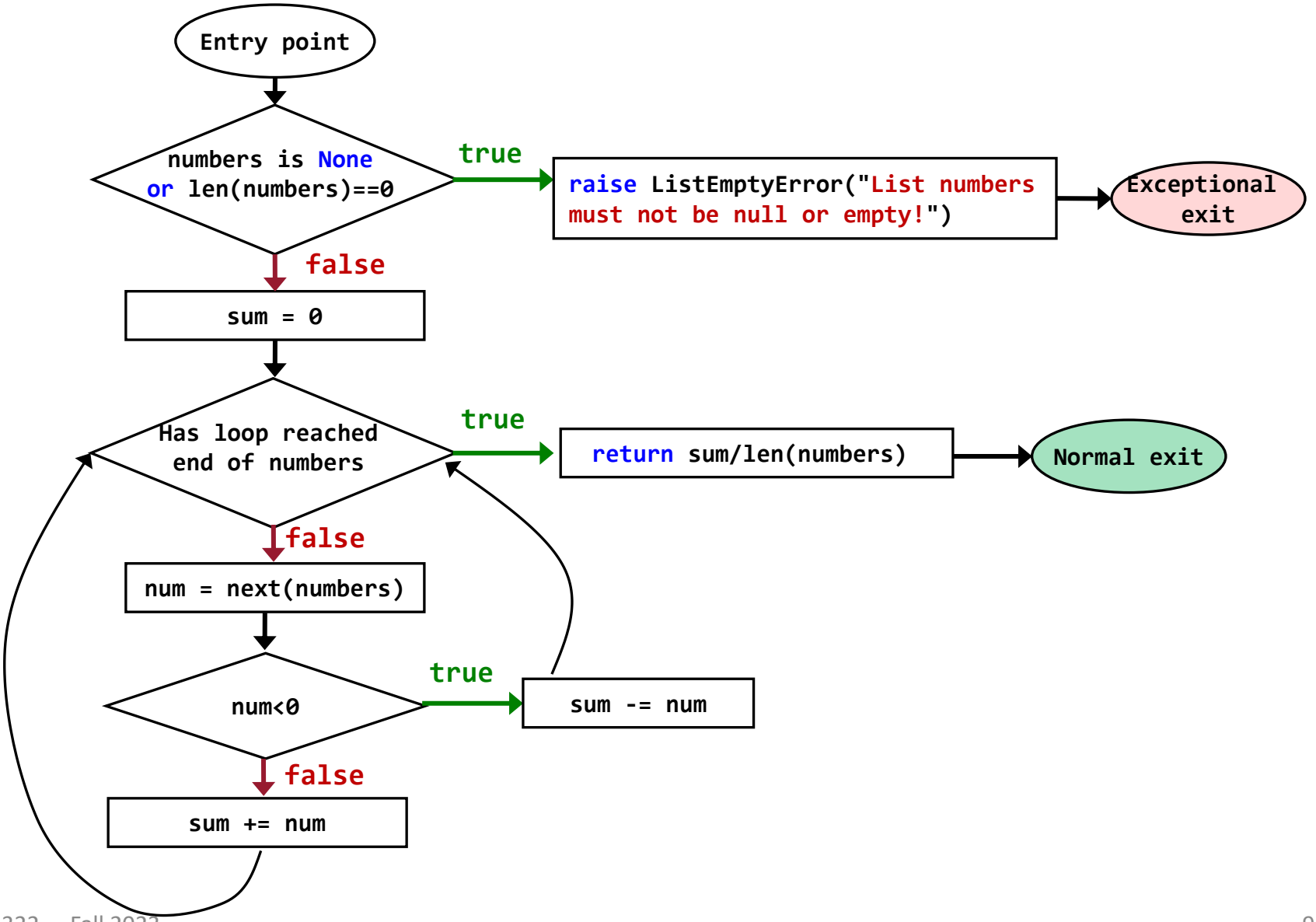
Structural Code Coverage: basics

- **Average of the absolute values of an array of doubles**

```
class ListEmptyError(Exception):
        """Raised when the input value is too small"""
        pass

class Avg():
    """Compute the average of the absolute values of an array of doubles """
    def avgAbs(self,numbers):
        # We expect the array to be non-null and non-empty
        if (numbers is None or len(numbers) == 0):
            raise ListEmptyError("List numbers must not be null or empty!");

        sum = 0
        for num in numbers:
            if (num < 0):
                sum -= num
            else:
                sum += num

        return sum/len(numbers)
```

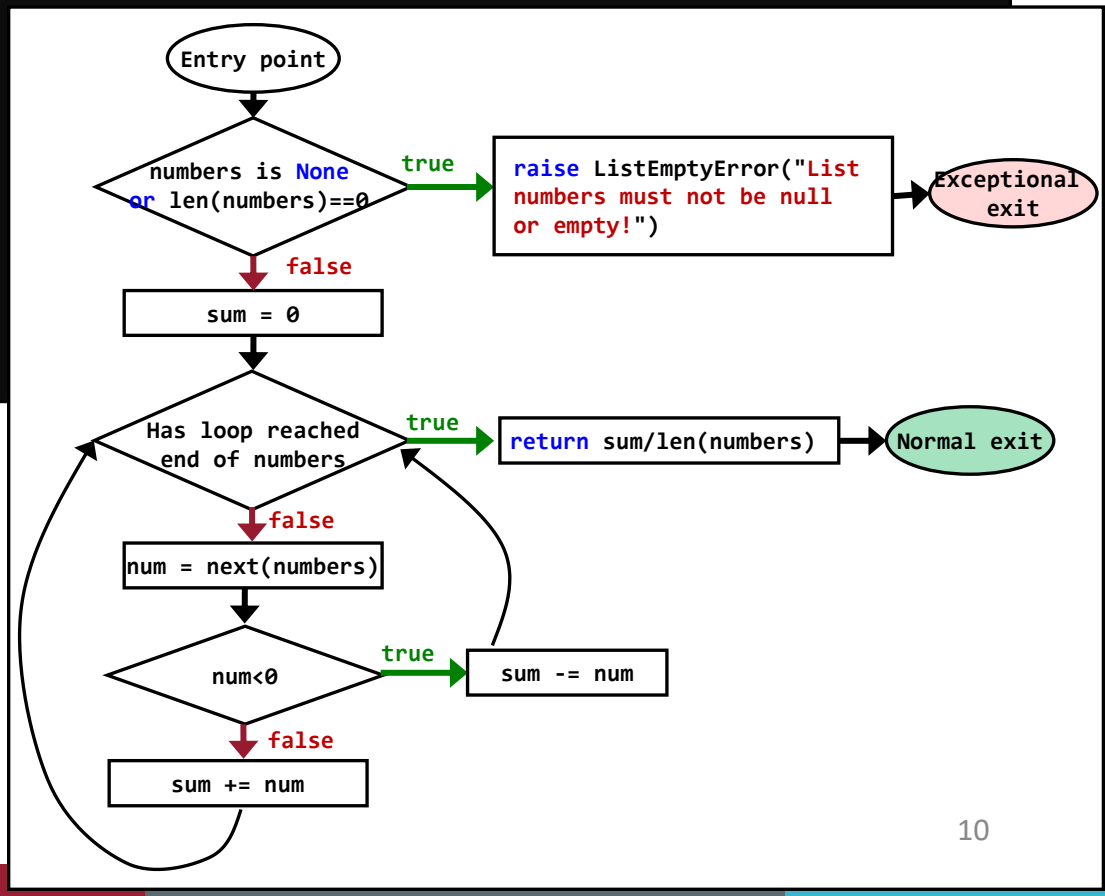
What is the control flow graph (CFG) for this method?

Structural Code Coverage: basics



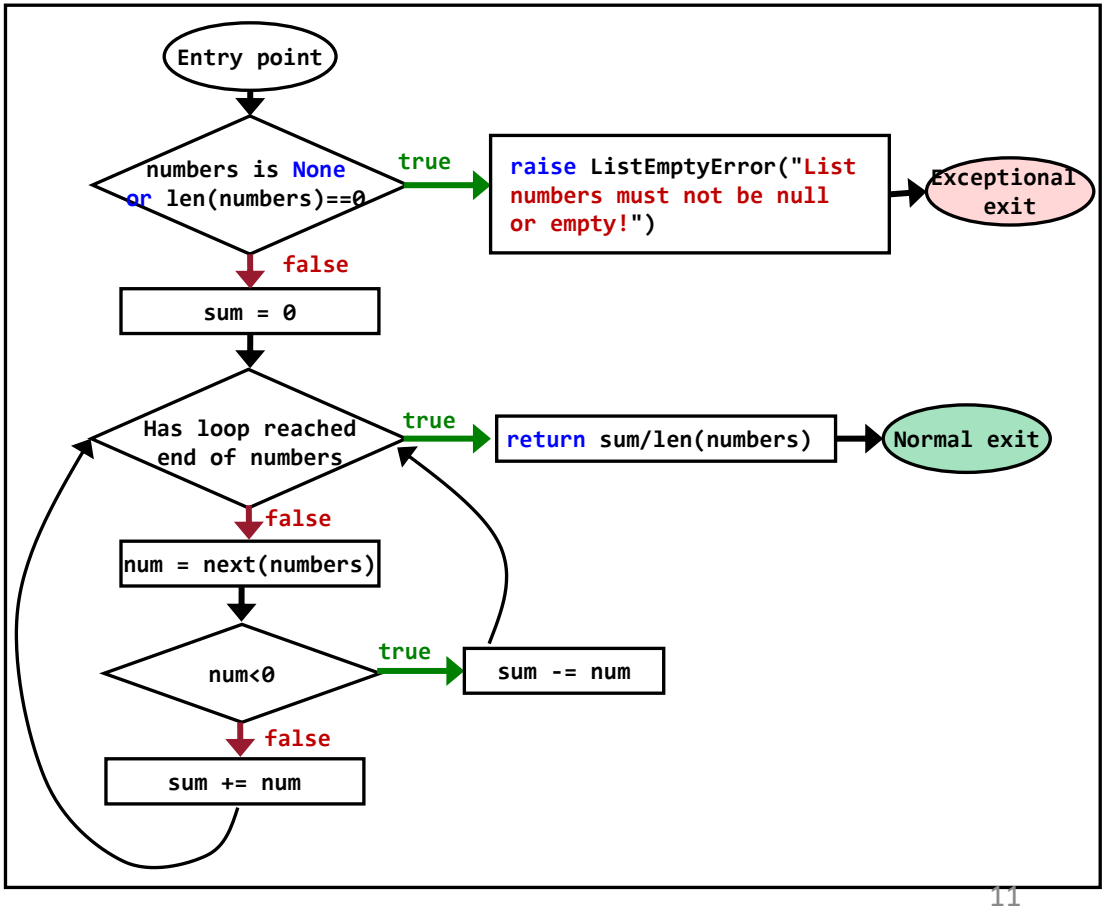
Average of the absolute values of an array of doubles

```
class ListEmptyError(Exception):
        """Raised when the input value is too small"""
        pass

class Avg():
    """Compute the average of the absolute values of an array of doubles """
    def avgAbs(self,numbers):
        # We expect the array to be non-null and non-empty
        if (numbers is None or len(numbers) == 0):
            raise ListEmptyError("List numbers must not be null or empty!");

        sum = 0
        for num in numbers:
            if (num < 0):
                sum -= num
            else:
                sum += num

        return sum/len(numbers)
```

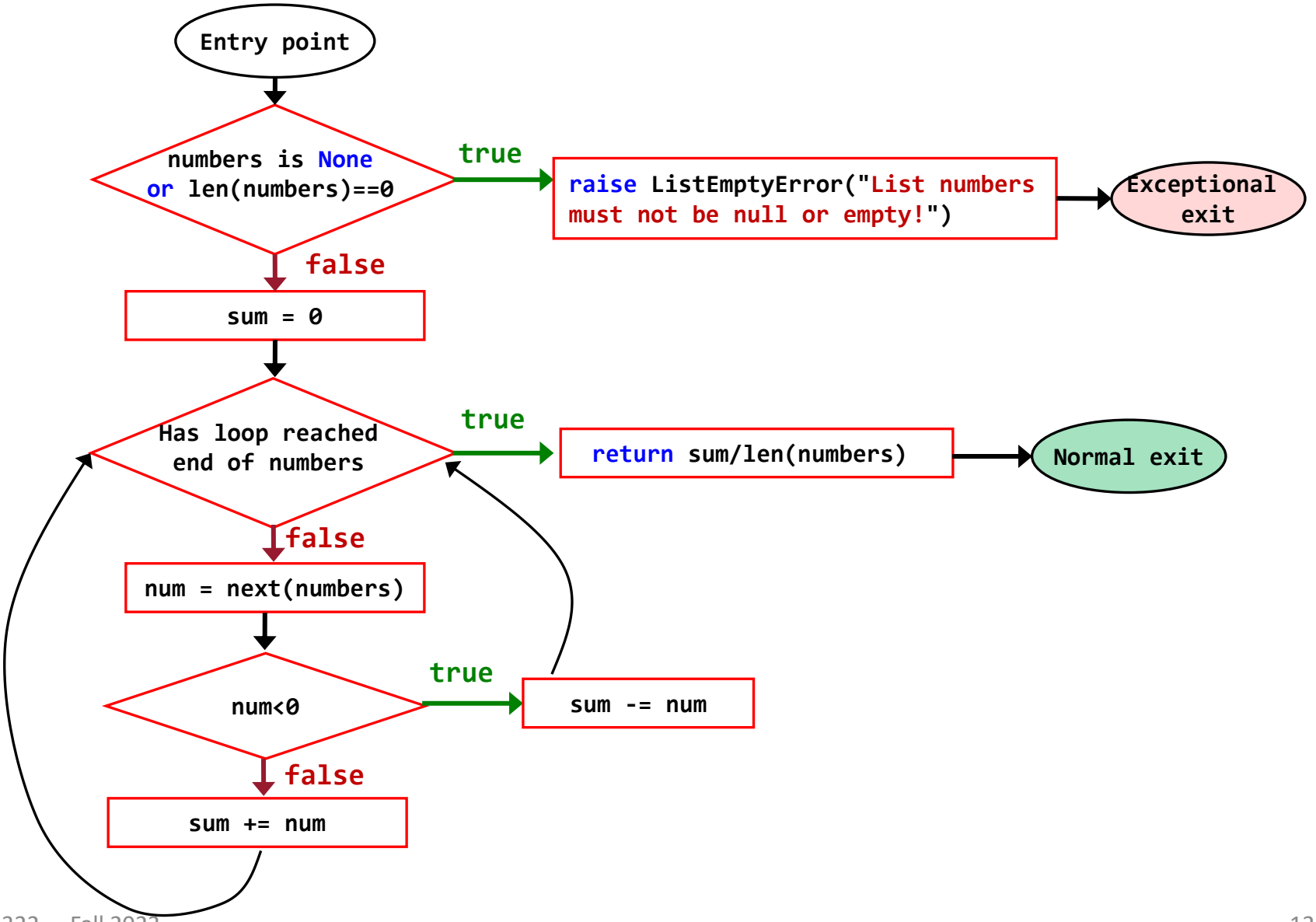


Statement Coverage

- **Every statement** in the program must be **executed at least once**.



Statement Coverage



Statement Coverage

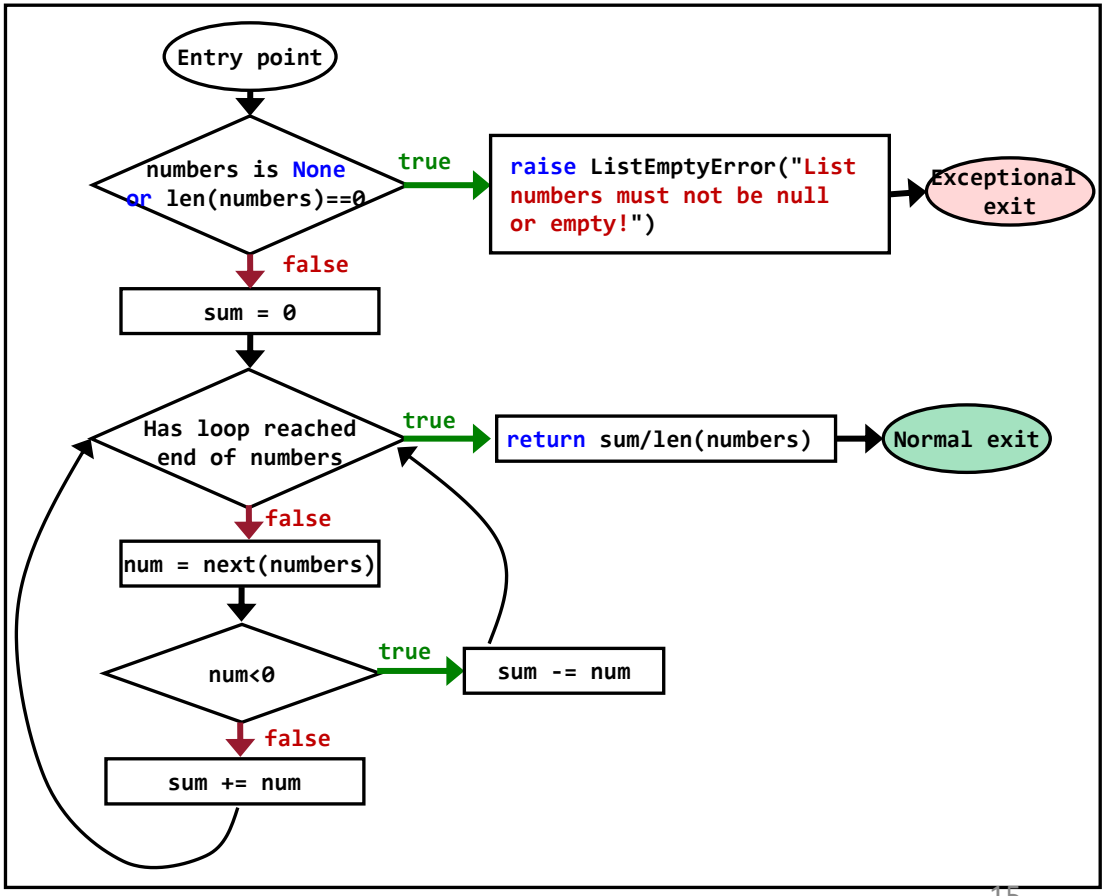
- Every statement in the program must be executed at least once.
- Given the control-flow graph (CFG), this is equivalent to node coverage.

Condition coverage vs. decision coverage

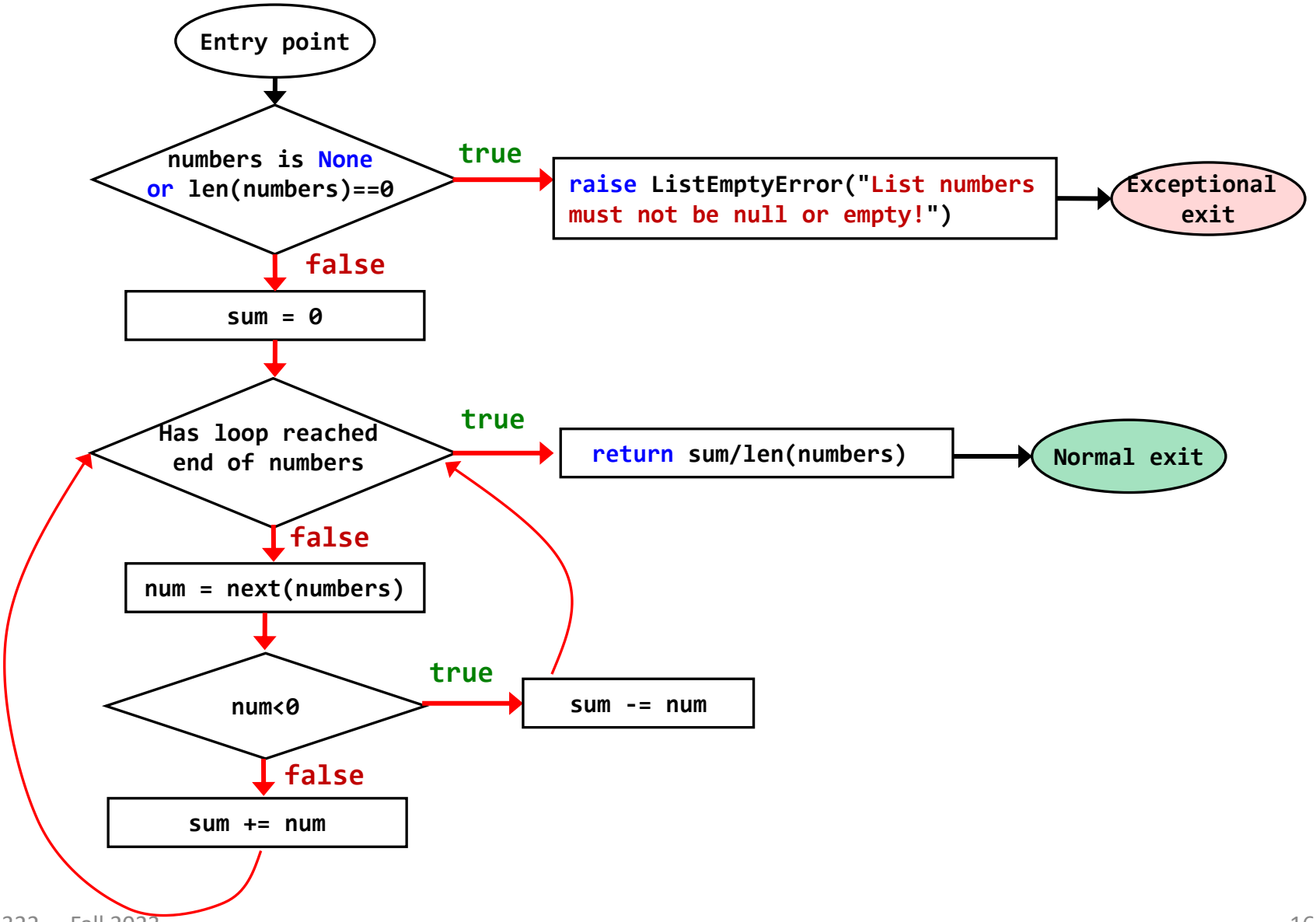
- Terminology
  - Condition: a boolean expression that cannot be decomposed into simpler boolean expressions.
  - Decision: a boolean expression that is composed of conditions, using 0 or more logical connectors
    - a decision with 0 logical connectors is a condition.
  - **Example**:
    - `if (a and b) { … }`
      - a and b are conditions.
      - The boolean expression a & b is a decision.

Decision Coverage

- **Every decision** in the program must take on **all possible outcomes** (true/false) **at least once.**



Decision Coverage



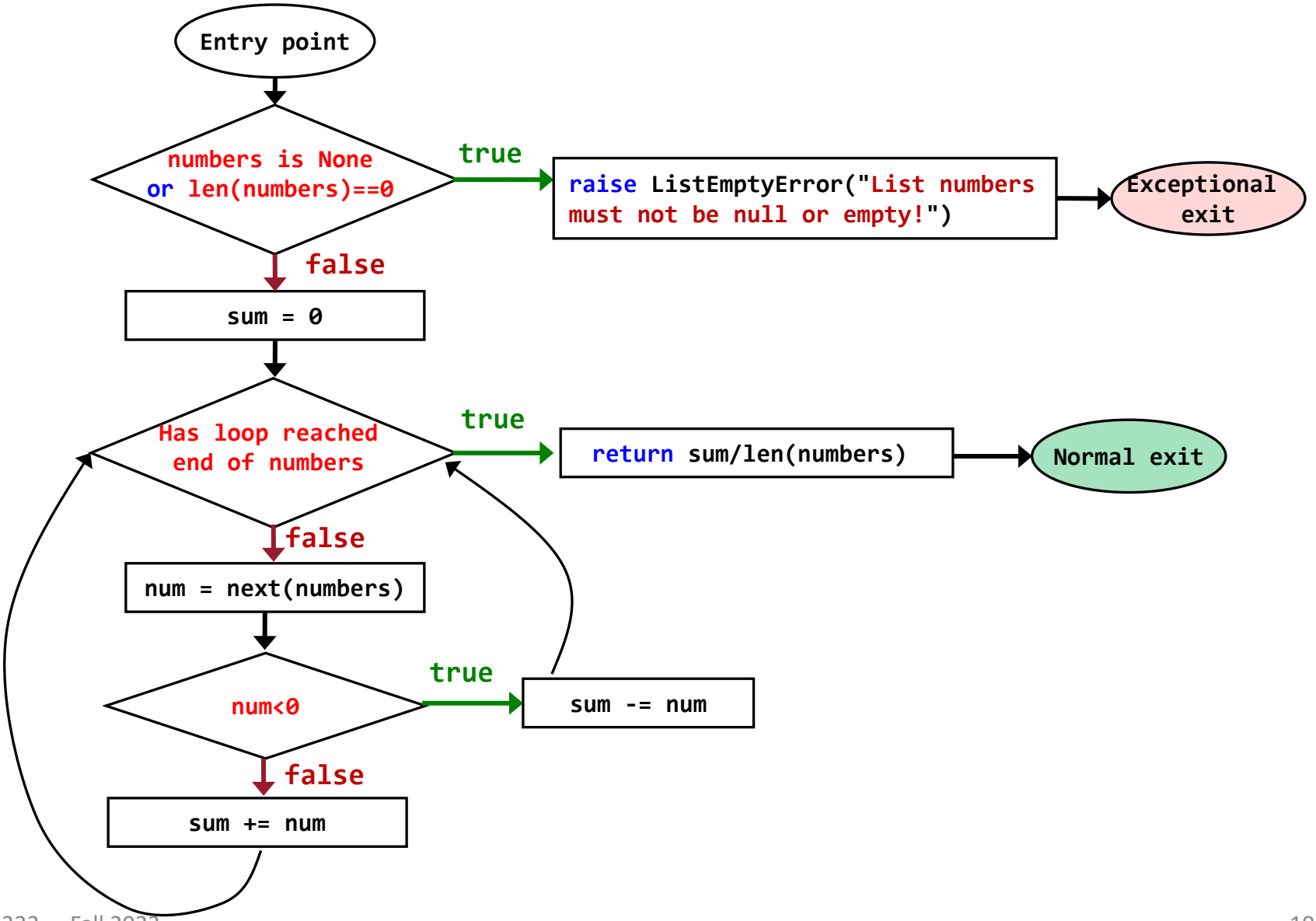
Decision Coverage (aka branch coverage)

- Every decision in the program must take on all possible outcomes (true/false) at least once
- Given the CFG, this is equivalent to **edge coverage**
- Example:
  - `(a>0 and b>0)`
    - `a=1, b=1`
    - `a=0, b=0`

Condition Coverage

- Every condition in the program must take on all possible outcomes (true/false) at least once
- Example:
  - `(a>0 and b>0)`
    - `a=1, b=0`
    - `a=0, b=1`

Structural Code Coverage: condition coverage



Structural code coverage: Tools

- `Coverage.py`
  - a tool for measuring code coverage of Python programs.
  - coverage measurement show which parts of your code are being exercised by tests, and which are not.

- How to use `Coverage.py`
  - Install coverage.py: `pip install coverage`
  - Run coverage analysis:
    - unittest: `coverage run -m unittest <testfile>`
    - pytest: `coverage run -m pytest <testfile>`
  - Use coverage report to report on the results:
    - `coverage report -m`
  - For a nicer presentation, use coverage html to get annotated HTML listings detailing missed lines:
    - `coverage html`

Structural code coverage: Summary

avg.py > Avg > avgAbs

```
1
2   class ListEmptyError(Exception):
3       """Raised when the input value is too small"""
4       pass
5
6   class Avg():
7       """Compute the average of the absolute values of an array of doubles """
8       def avgAbs(self,numbers):
9           # We expect the array to be non-null and non-empty
10          if (numbers is None or len(numbers) == 0):
11              raise ListEmptyError("List numbers must not be null or empty!");
12
13          sum = 0
14          for num in numbers:
15              if (num < 0):
16                  sum -= num
17              else:
18                  sum += num
19
20          return sum/len(numbers)
```

- Code coverage is easy to compute.
- Code coverage has an intuitive interpretation.
- Code coverage itself is not sufficient!
- Code coverage in industry: Code coverage at Google