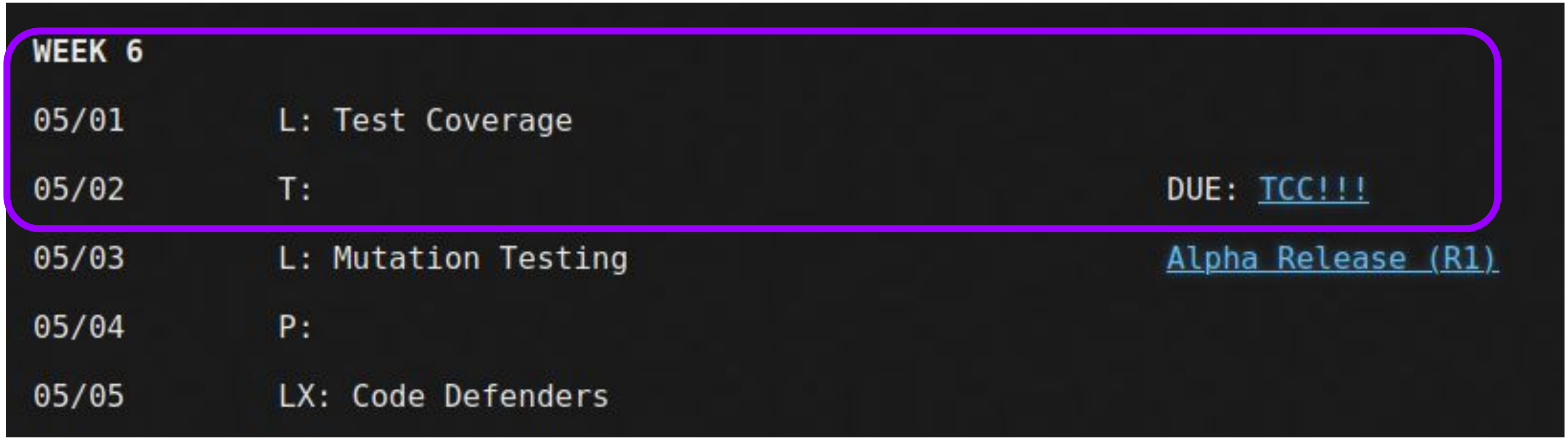
CSE 403

Software Engineering

Spring 2023

#16: Coverage-based Testing

Logistics



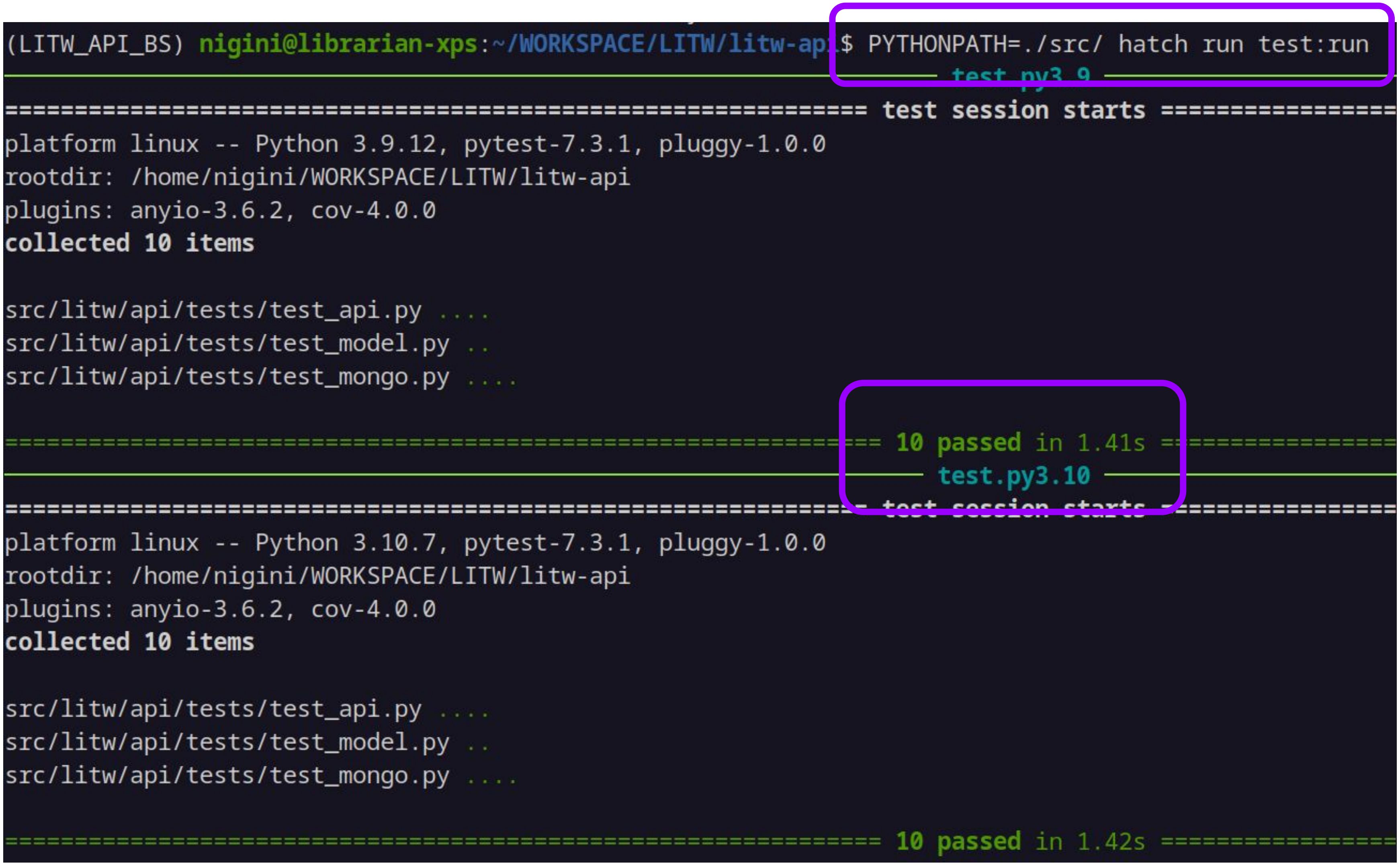
WEEK 6

05/01	L: Test Coverage	
05/02	T:	DUE: TCC!!!
05/03	L: Mutation Testing	Alpha Release (R1)
05/04	P:	
05/05	LX: Code Defenders	

This week: test efficacy and adequacy

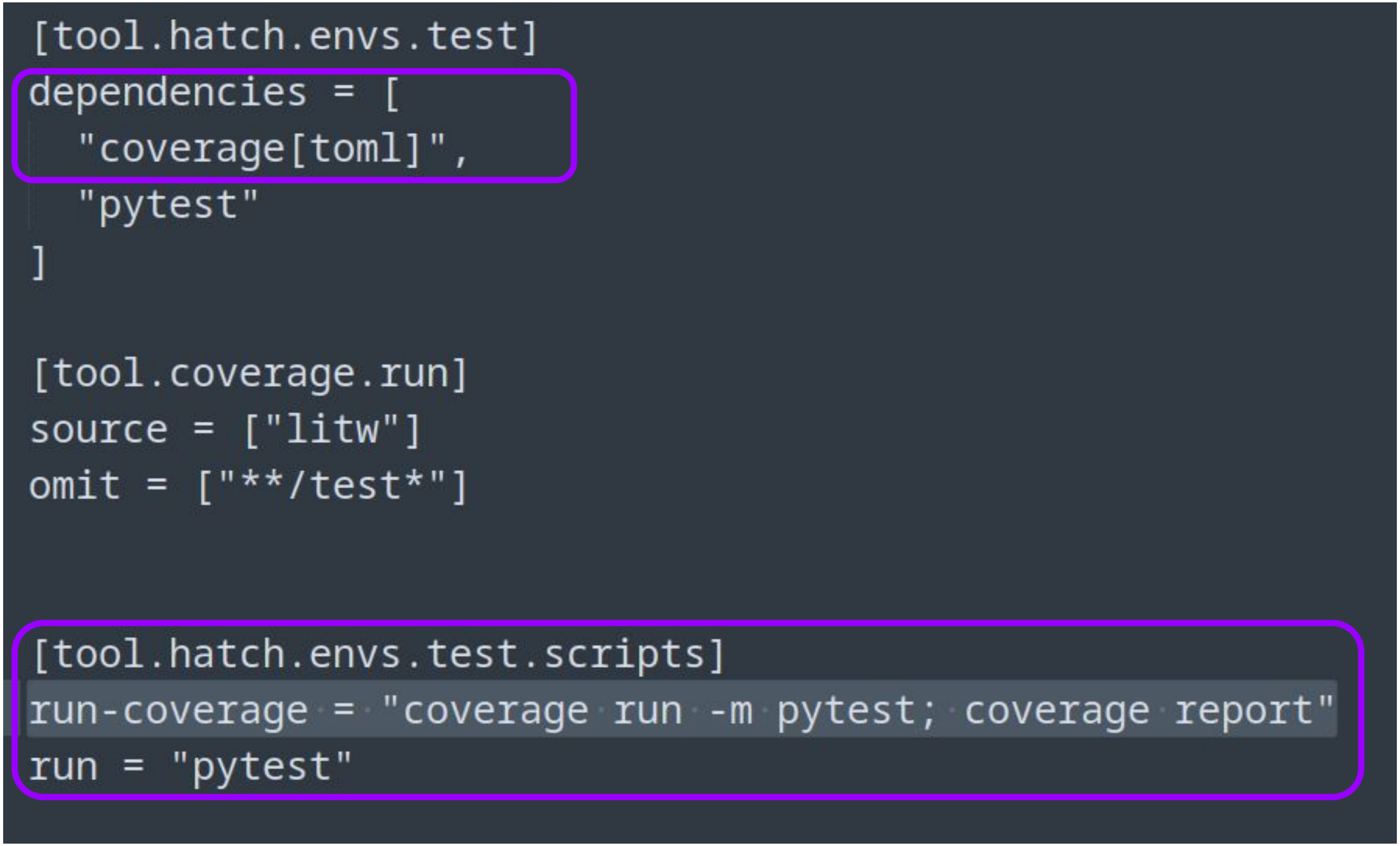
- Coverage-based testing
- Mutation-based testing
- In-class exercise

Structural code coverage: motivating example



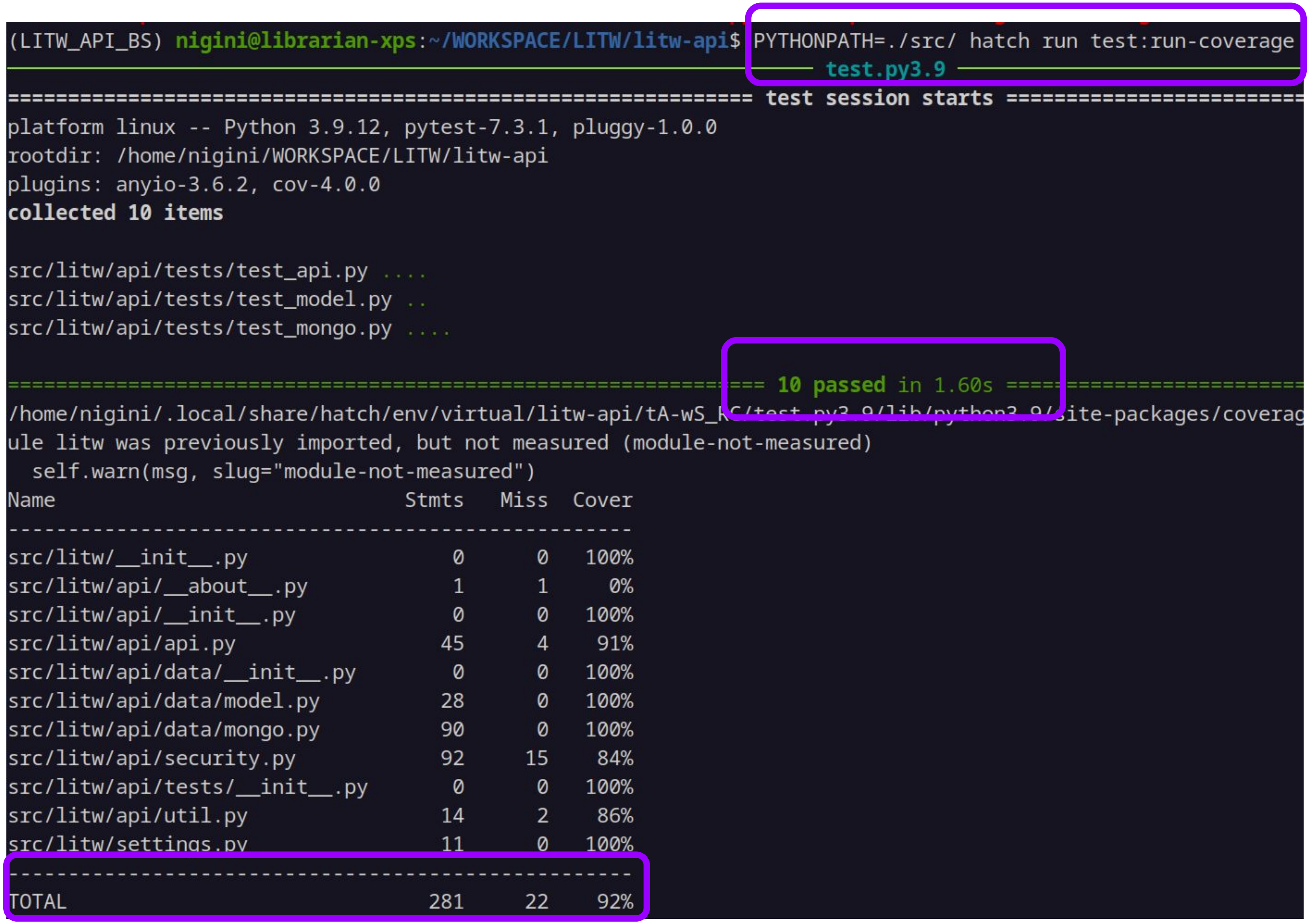
```
(LITW_API_BS) nigini@librarian-xps:~/WORKSPACE/LITW/litw-api$ PYTHONPATH=./src/ hatch run test:run
──────────────────────────────────────────────────────────────── test.py3.9 ────────────────────────
============================================================== test session starts ================
platform linux -- Python 3.9.12, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigini/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2, cov-4.0.0
collected 10 items

src/litw/api/tests/test_api.py ....
src/litw/api/tests/test_model.py ..
src/litw/api/tests/test_mongo.py ....

=============================================================== 10 passed in 1.41s ================
─────────────────────────────────────────────────────────────── test.py3.10 ────────────────────────
============================================================== test session starts ================
platform linux -- Python 3.10.7, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigini/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2, cov-4.0.0
collected 10 items

src/litw/api/tests/test_api.py ....
src/litw/api/tests/test_model.py ..
src/litw/api/tests/test_mongo.py ....

=============================================================== 10 passed in 1.42s ================
```

Structural code coverage: motivating example



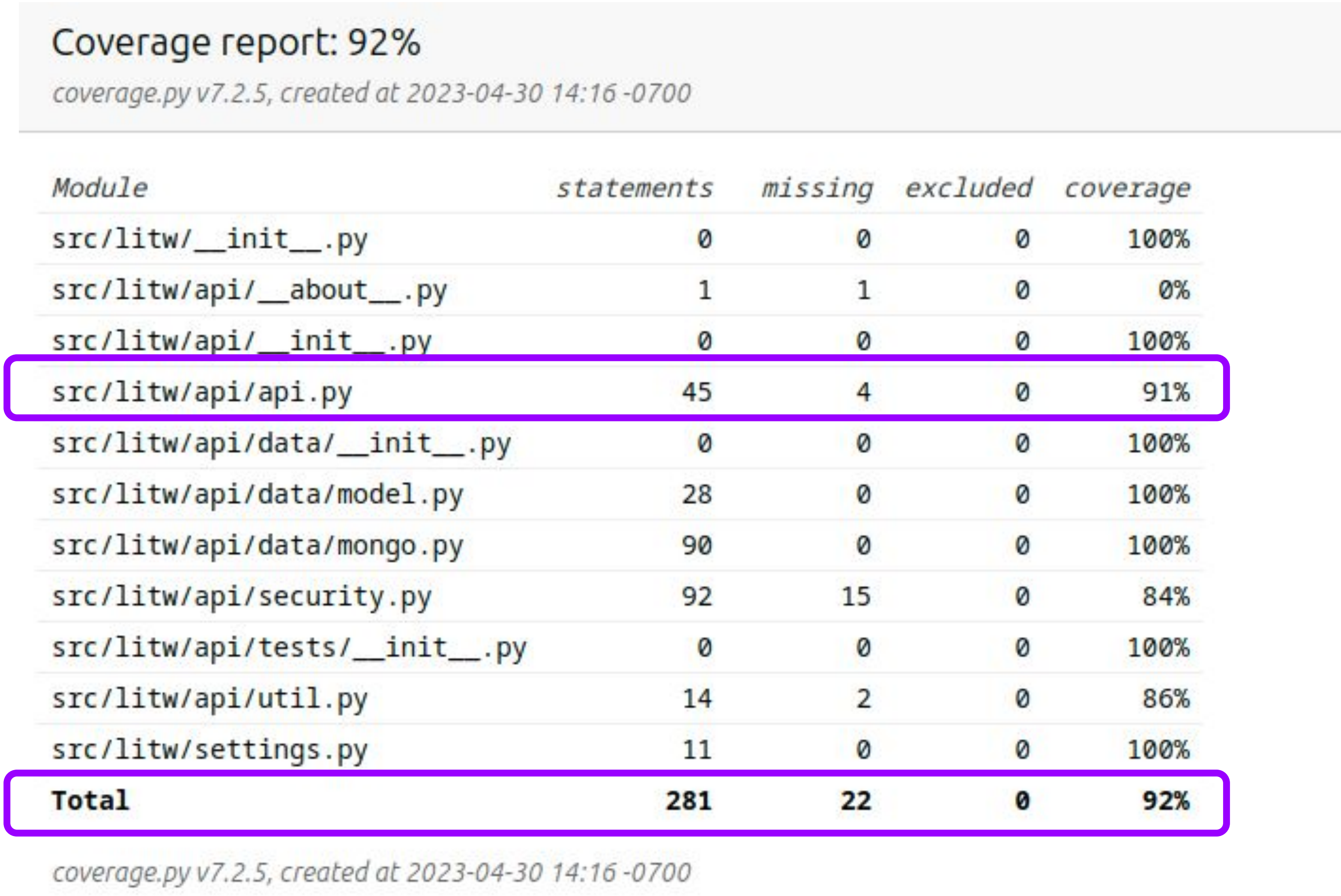
```
[tool.hatch.envs.test]
dependencies = [
  "coverage[toml]",
  "pytest"
]

[tool.coverage.run]
source = ["litw"]
omit = ["**/test*"]


[tool.hatch.envs.test.scripts]
run-coverage = "coverage run -m pytest; coverage report"
run = "pytest"
```

Structural code coverage: motivating example



```
(LITW_API_BS) nigini@librarian-xps:~/WORKSPACE/LITW/litw-api$ PYTHONPATH=./src/ hatch run test:run-coverage
──────────────────────────────────────────────────────────────── test.py3.9 ─────────────────────────────
============================================================== test session starts =========================
platform linux -- Python 3.9.12, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigini/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2, cov-4.0.0
collected 10 items

src/litw/api/tests/test_api.py ....
src/litw/api/tests/test_model.py ..
src/litw/api/tests/test_mongo.py ....

================================================================ 10 passed in 1.60s ========================
/home/nigini/.local/share/hatch/env/virtual/litw-api/tA-wS_KC/test.py3.9/lib/python3.9/site-packages/coverag
ule litw was previously imported, but not measured (module-not-measured)
  self.warn(msg, slug="module-not-measured")
Name                                 Stmts   Miss  Cover
--------------------------------------------------------
src/litw/__init__.py                     0      0   100%
src/litw/api/__about__.py                1      1     0%
src/litw/api/__init__.py                 0      0   100%
src/litw/api/api.py                     45      4    91%
src/litw/api/data/__init__.py            0      0   100%
src/litw/api/data/model.py              28      0   100%
src/litw/api/data/mongo.py              90      0   100%
src/litw/api/security.py                92     15    84%
src/litw/api/tests/__init__.py           0      0   100%
src/litw/api/util.py                    14      2    86%
src/litw/settings.py                    11      0   100%
--------------------------------------------------------
TOTAL                                  281     22    92%
```

Structural code coverage: motivating example



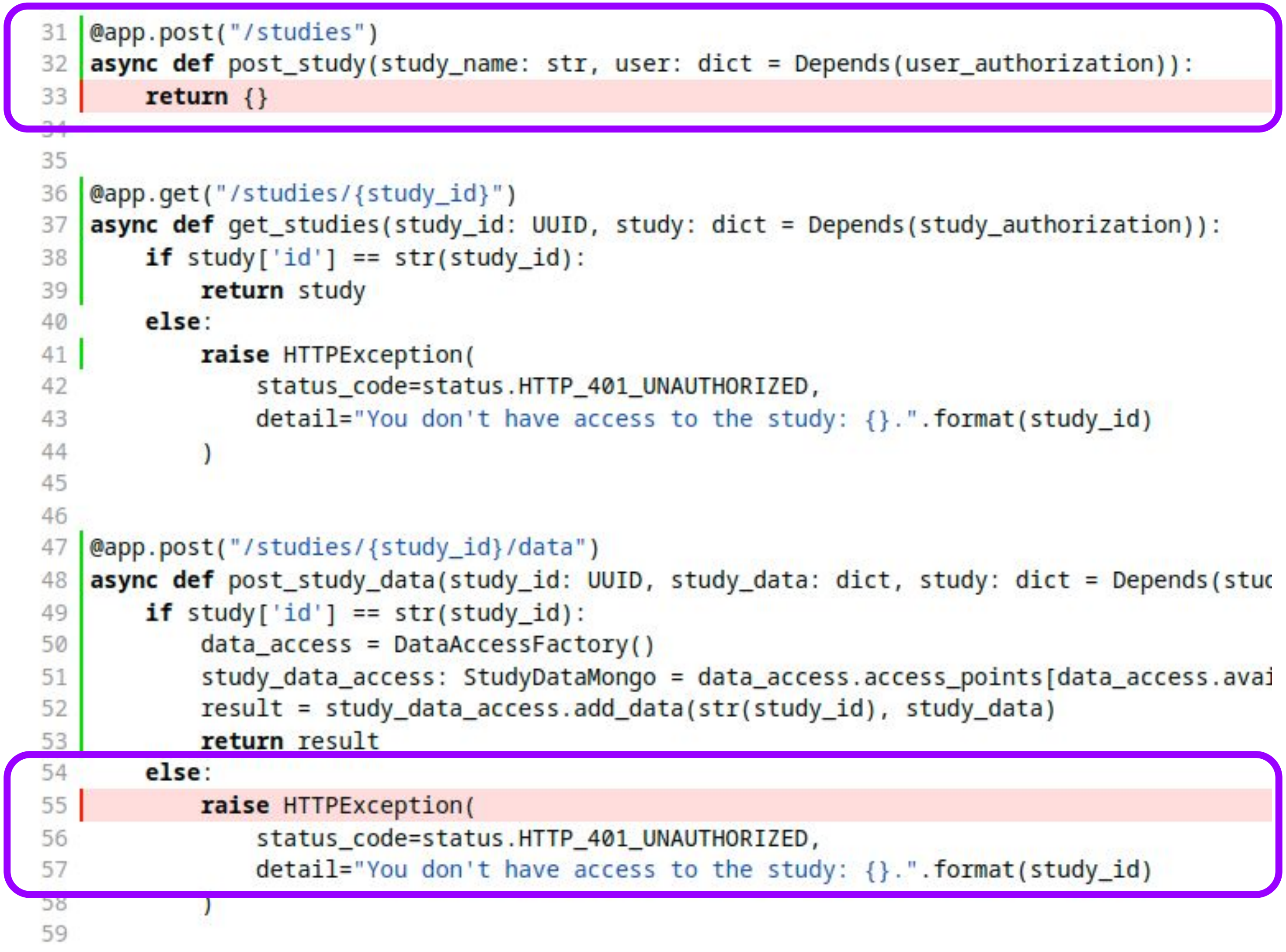
Coverage report: 92%

coverage.py v7.2.5, created at 2023-04-30 14:16 -0700

Module	*statements*	*missing*	*excluded*	*coverage*
src/litw/__init__.py	0	0	0	100%
src/litw/api/__about__.py	1	1	0	0%
src/litw/api/__init__.py	0	0	0	100%
src/litw/api/api.py	45	4	0	91%
src/litw/api/data/__init__.py	0	0	0	100%
src/litw/api/data/model.py	28	0	0	100%
src/litw/api/data/mongo.py	90	0	0	100%
src/litw/api/security.py	92	15	0	84%
src/litw/api/tests/__init__.py	0	0	0	100%
src/litw/api/util.py	14	2	0	86%
src/litw/settings.py	11	0	0	100%
Total	**281**	**22**	**0**	**92%**

coverage.py v7.2.5, created at 2023-04-30 14:16 -0700

Structural code coverage: motivating example



```
@app.post("/studies")
async def post_study(study_name: str, user: dict = Depends(user_authorization)):
    return {}


@app.get("/studies/{study_id}")
async def get_studies(study_id: UUID, study: dict = Depends(study_authorization)):
    if study['id'] == str(study_id):
        return study
    else:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="You don't have access to the study: {}.".format(study_id)
        )


@app.post("/studies/{study_id}/data")
async def post_study_data(study_id: UUID, study_data: dict, study: dict = Depends(stuc
    if study['id'] == str(study_id):
        data_access = DataAccessFactory()
        study_data_access: StudyDataMongo = data_access.access_points[data_access.avai
        result = study_data_access.add_data(str(study_id), study_data)
        return result
    else:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="You don't have access to the study: {}.".format(study_id)
        )

```

Code coverage metrics

- Statement coverage
- Branch coverage
  - Condition coverage
  - Decision coverage
  - Modified Condition/Decision coverage

Structural code coverage: the basics

Average of the absolute values of an array of doubles

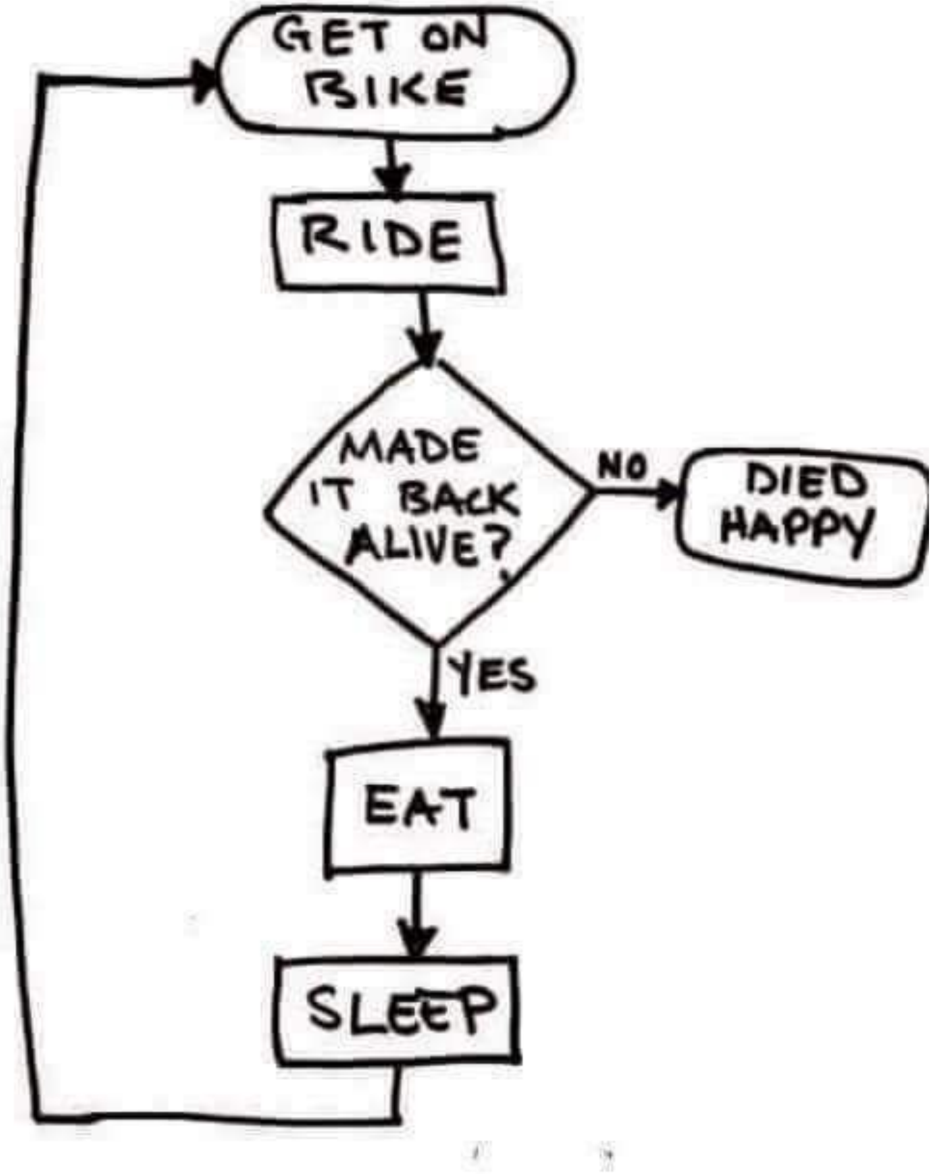
```
public double avgAbs(double ... numbers) {

  // We expect the array to be non-null and non-empty
  if (numbers == null || numbers.length == 0) {
    throw new IllegalArgumentException("Array numbers must not be null or empty!");
  }

  double sum = 0;
  for (int i=0; i<numbers.length; ++i) {
    double d = numbers[i];
    if (d < 0) {
      sum -= d;
    } else {
      sum += d;
    }
  }

  return sum/numbers.length;
}
```

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

Structural code coverag

Average of the absolute val

```
public double avgAbs(double ... numbers)

  // We expect the array to be non-null
  if (numbers == null || numbers.length
    throw new IllegalArgumentException('
  }

  double sum = 0;
  for (int i=0; i<numbers.length; ++i) {
    double d = numbers[i];
    if (d < 0) {
      sum -= d;
    } else {
      sum += d;
    }
  }

  return sum/numbers.length;
}
```



What’s the **control flow graph (CFG)** for this method?

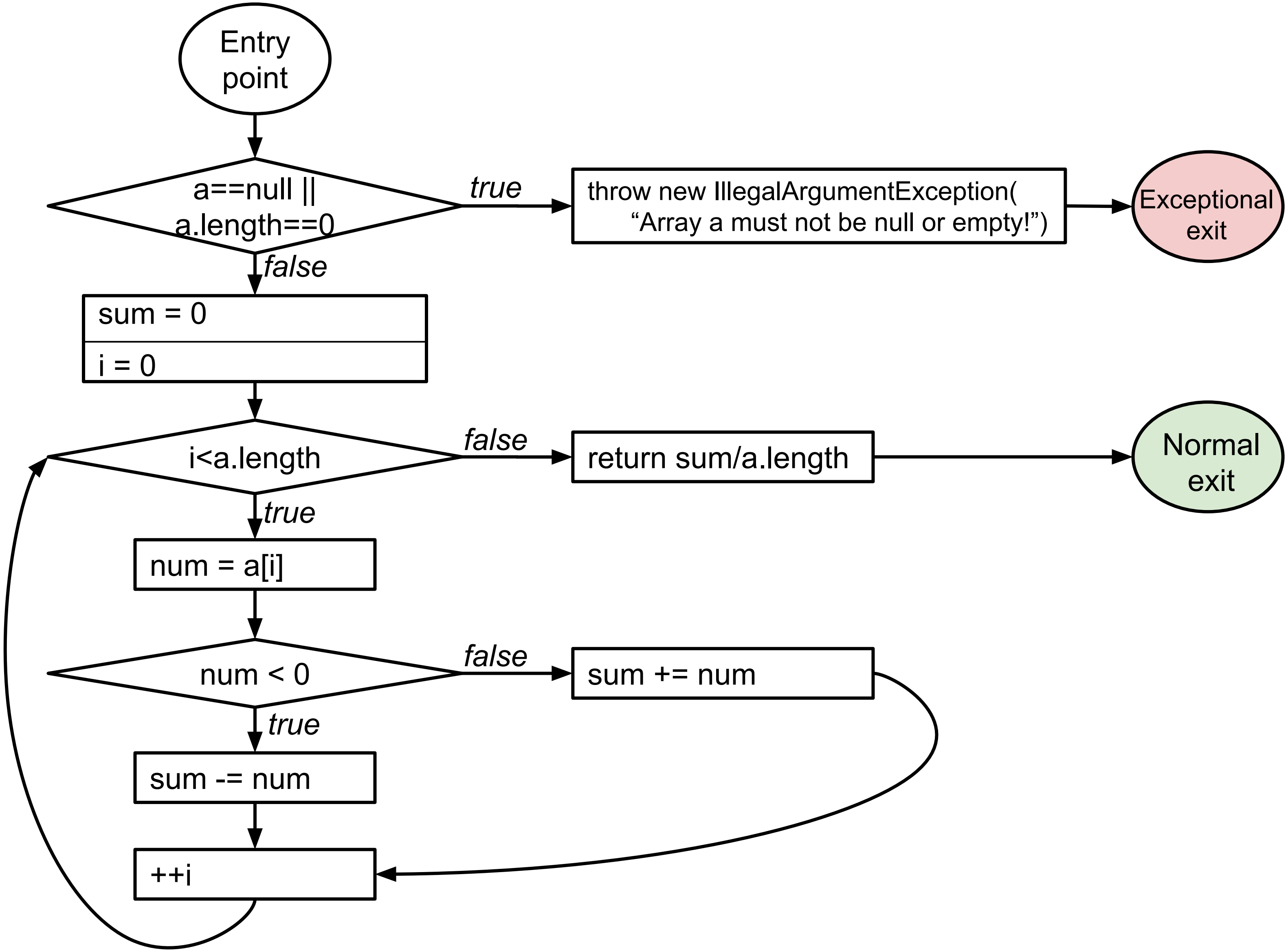
Structural code coverage: the basics

Average of the absolute values of an array of doubles

```
public double avgAbs(double ... numbers) {

  // We expect the array to be non-null and non-empty
  if (numbers == null || numbers.length == 0) {
    throw new IllegalArgumentException("Array numbers must not be null or empty!");
  }

  double sum = 0;
  for (int i=0; i<numbers.length; ++i) {
    double d = numbers[i];
    if (d < 0) {
      sum -= d;
    } else {
      sum += d;
    }
  }

  return sum/numbers.length;
}
```

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

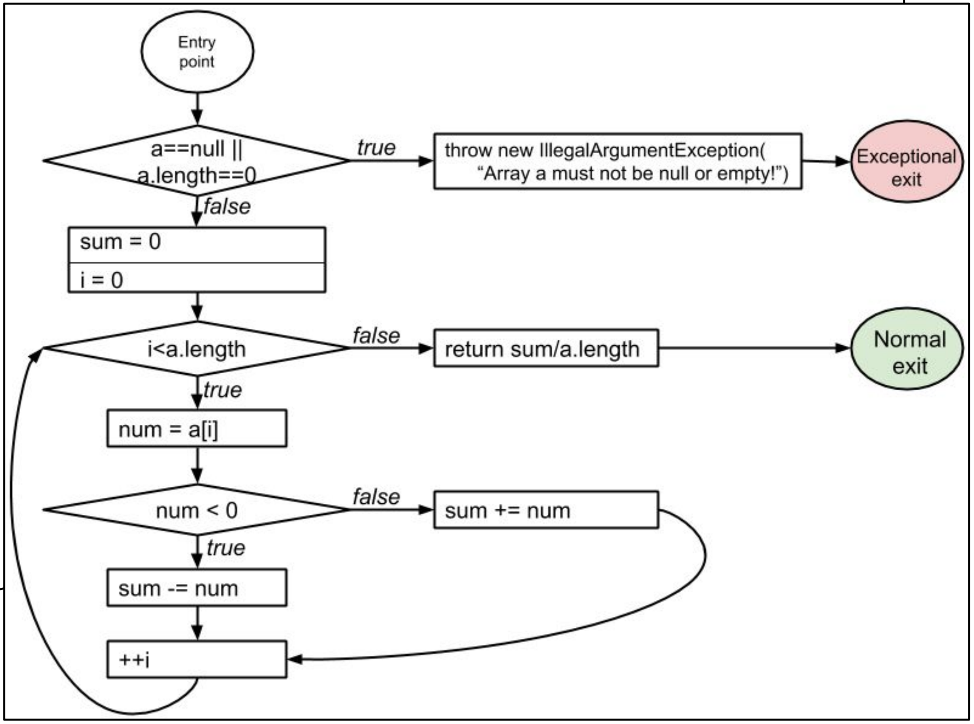
Structural code coverage: the basics



Structural code coverage: the basics

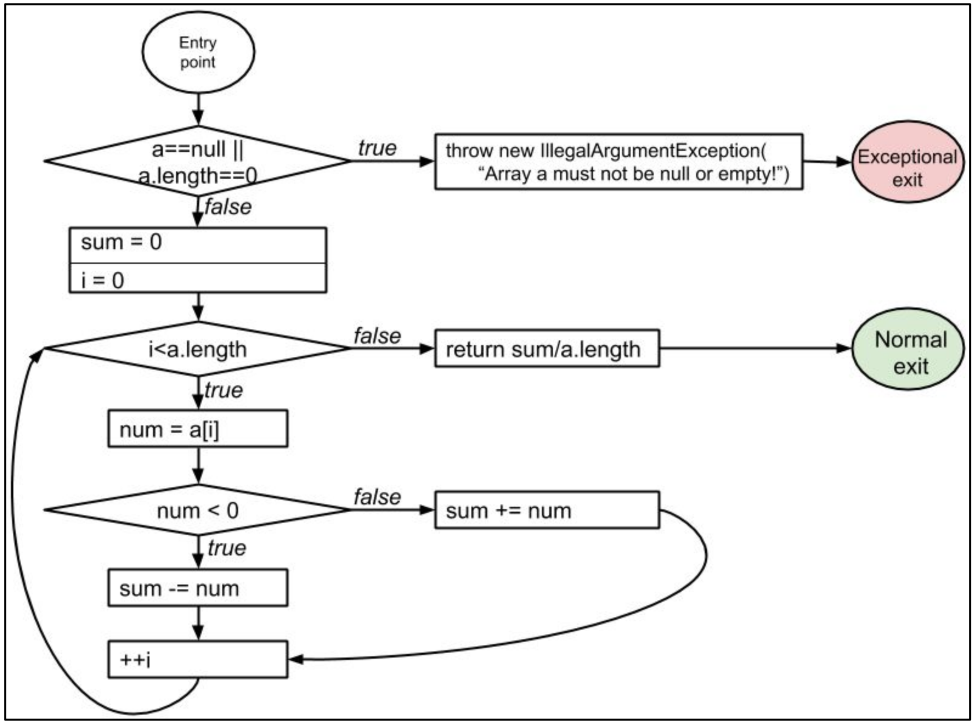
Average of the absolute values of an array of doubles

```
public double avgAbs(double ... numbers) {

  // We expect the array to be non-null and non-empty
  if (numbers == null || numbers.length == 0) {
    throw new IllegalArgumentException("Array numbers must not be null or empty!");
  }

  double sum = 0;
  for (int i=0; i<numbers.length; ++i) {
    double d = numbers[i];
    if (d < 0) {
      sum -= d;
    } else {
      sum += d;
    }
  }

  return sum/numbers.length;
}
```

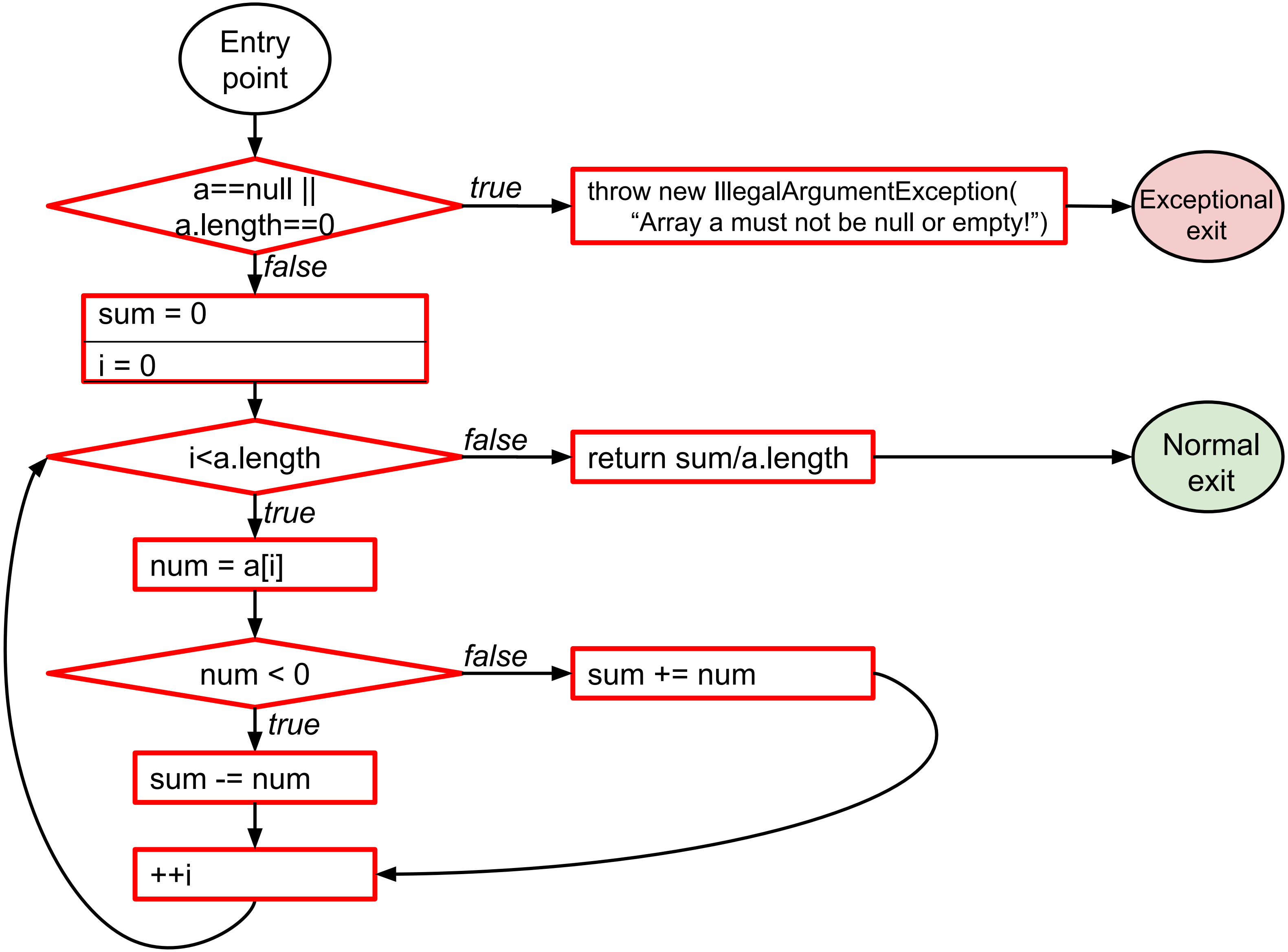


Statement coverage

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



Statement coverage


Entry point

a==null \|\| a.length==0

true

throw new IllegalArgumentException(“Array a must not be null or empty!”)

Exceptional exit

false

sum = 0

i = 0

i<a.length

false

return sum/a.length

Normal exit

true

num = a[i]

num < 0

false

sum += num

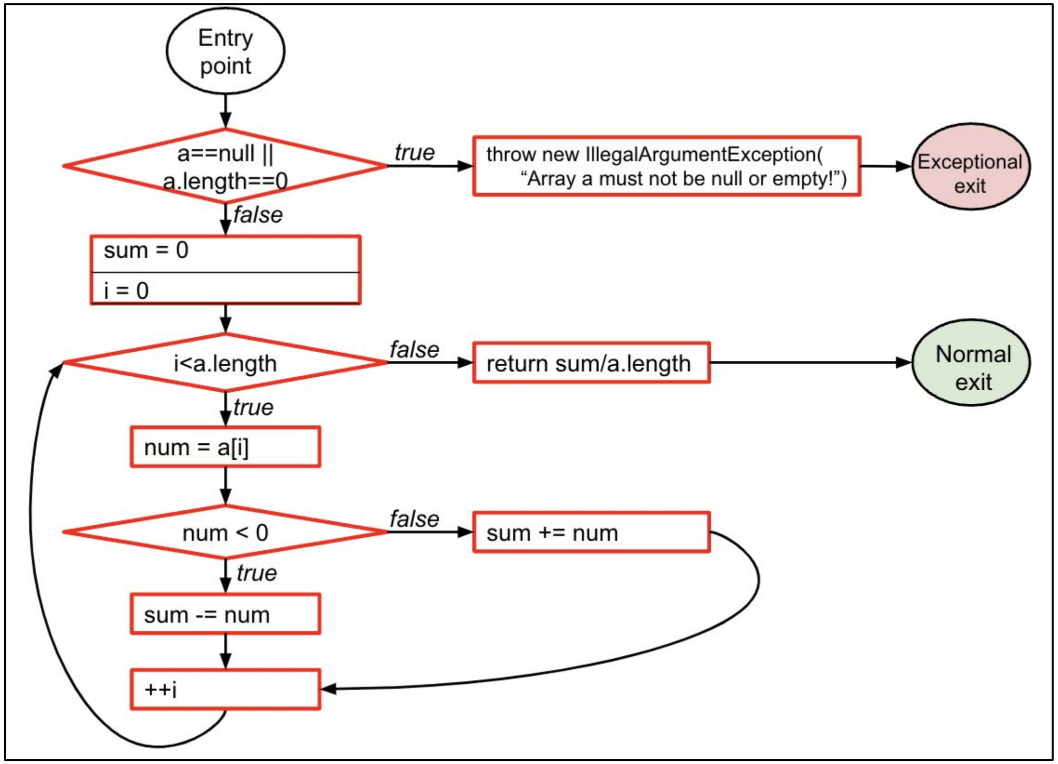
true

sum -= num

++i

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.



Branch coverage: Condition vs. Decision

Branch coverage: Condition vs. Decision

Terminology

- **Condition**: a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).
- **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* | *b*) { … }
    - *a* and *b* are ***conditions.***
    - The boolean expression *a* | *b* is a *decision.*

Branch coverage: Condition vs. Decision

Terminology

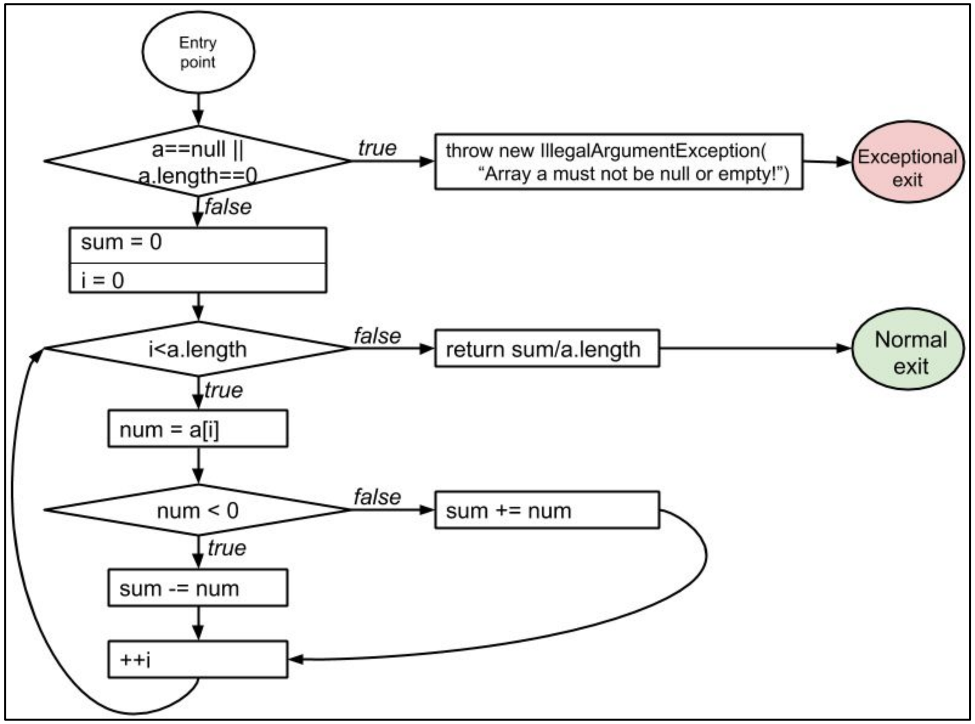
- **Condition**: a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).

- **Decision**: a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).

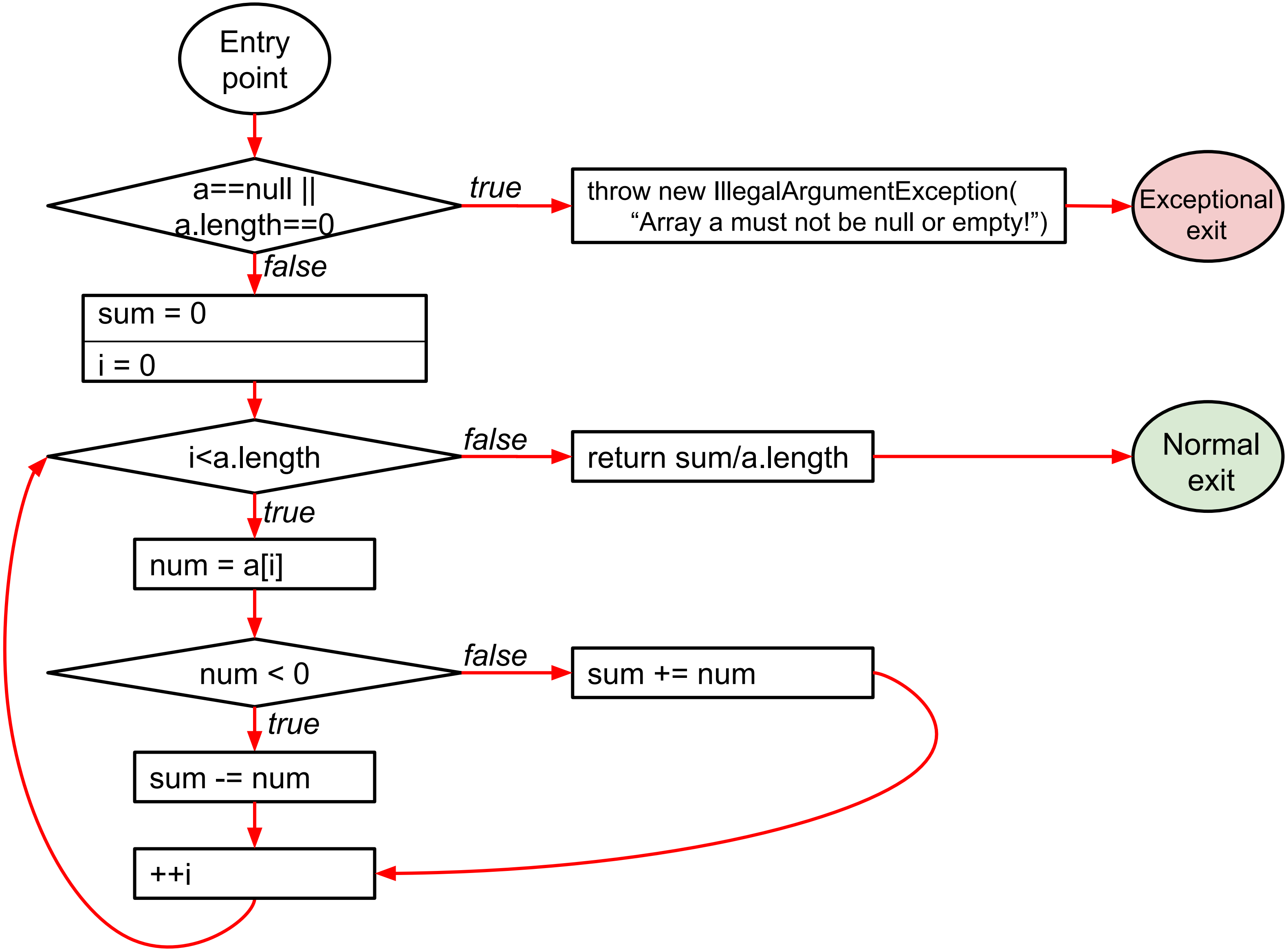
- **Example:** if (*a* | *b*) { … }
    - *a* and *b* are *conditions.*
    - The boolean expression (*a* | *b*) is a ***decision****.*

Decision coverage

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



Decision coverage



Branch coverage: **Condition** vs. Decision

Terminology

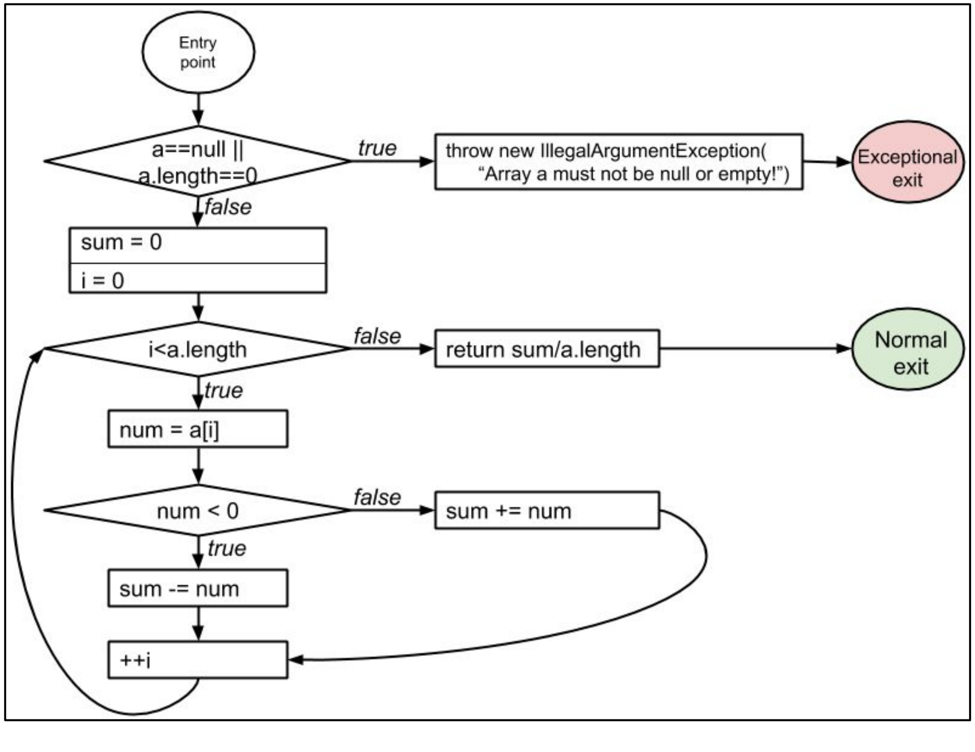
- **Condition**: a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).

- **Decision**: a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).

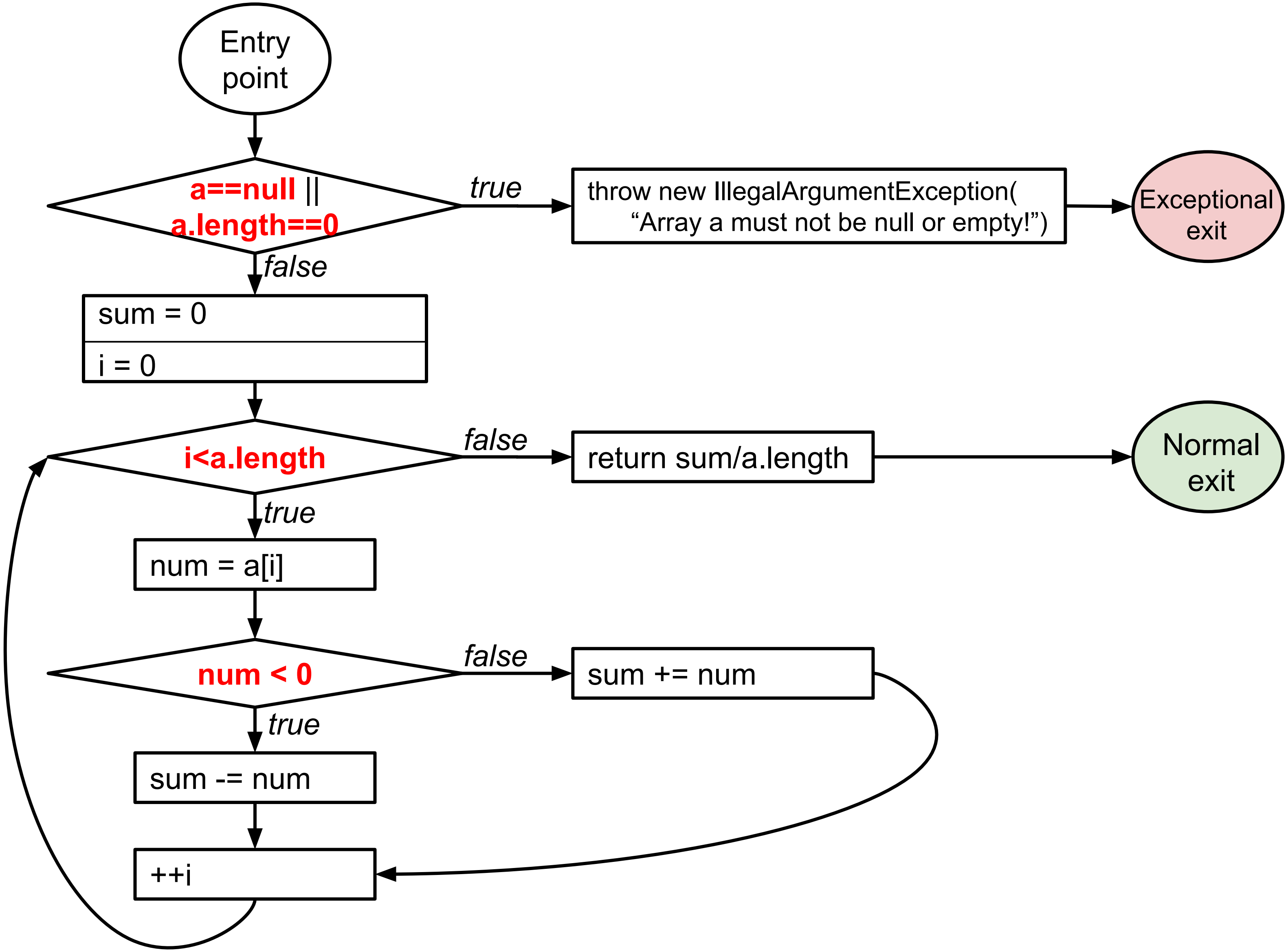
- **Example:** if (*a* | *b*) { … }
  - *a* and *b* are ***conditions.***
  - The boolean expression *a* | *b* is a *decision.*

Condition coverage

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



Condition coverage



Code coverage metrics

- Statement coverage
- Branch coverage
  - Condition coverage
  - Decision coverage
  - Modified Condition/Decision coverage

Structural code coverage: subsumption

Given two coverage criteria A and B,
A subsumes B iff **satisfying A implies satisfying B**

- Subsumption relationships (True or False):
  1. Does **statement** coverage subsume **decision** coverage?
  2. Does **decision** coverage subsume **statement** coverage?
  3. Does **decision** coverage subsume **condition** coverage?
  4. Does **condition** coverage subsume **decision** coverage?

https://tinyurl.com/cse403-cov

Structural code coverage: subsumption

Given two coverage criteria A and B,
A subsumes B iff **satisfying A implies satisfying B**

- Subsumption relationships (True or False):
  1. Does **statement** coverage subsume **decision** coverage?
  2. Does **decision** coverage subsume **statement** coverage?
  3. Does **decision** coverage subsume **condition** coverage?
  4. Does **condition** coverage subsume **decision** coverage?

The only correct statement in #2!!!

Decision coverage vs. condition coverage

4 possible tests for the decision *a* | *b*:

1. *a* = 0, *b* = 0
2. *a* = 0, *b* = 1
3. *a* = 1, *b* = 0
4. *a* = 1, *b* = 1

a	*b*	*a* | *b*
0	0	0
0	**1**	**1**
1	**0**	**1**
1	1	1

Satisfies **condition coverage** but **not decision coverage**

a	*b*	*a* | *b*
0	**0**	**0**
0	**1**	**1**
1	0	1
1	1	1

Does **not** satisfy **condition coverage** but **decision coverage**

Neither coverage criterion subsumes the other!

Modified Condition/Decision Coverage (MC/DC)

Modified Condition/Decision Coverage (MC/DC)

Do not confuse… 🤘

MCDC: Modified condition and decision coverage

- **Every decision** in the program must take on all possible outcomes (true/false) **at least once**
- **Every condition** in the program must take on all possible outcomes (true/false) **at least once**
- **Each condition** in a decision has been shown to **independently affect that decision's outcome.**
  (A condition is shown to independently affect a decision's outcome by: varying just that condition while holding fixed all other possible conditions.)

Required for safety critical systems (DO-178B/C)

MC/DC: an example

```
if (a | b)
```

a	b	Outcome
0	0	0
0	1	1
1	0	1
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Which tests (combinations of a and b) satisfy MCDC?

MC/DC: an example

```
if (a | b)
```

a	b	Outcome
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

MCDC is still cheaper than testing all possible combinations.

MC/DC: another example

```
if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
1	0	1
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Why is this example different?

MC/DC: another example

```
if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
1	--	1
1	--	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Short-circuiting operators may not evaluate all conditions.

MC/DC: yet another example

```
if (!a) ... if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
1	0	1
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

What about this example?

MC/DC: another example

```
if (!a) ... if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
X	X	X
X	X	X

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Not all combinations of conditions may be possible.

MCDC: complex expressions

Provide an MCDC-adequate test suite for:

1. `a | b | c`
2. `a & b & c`

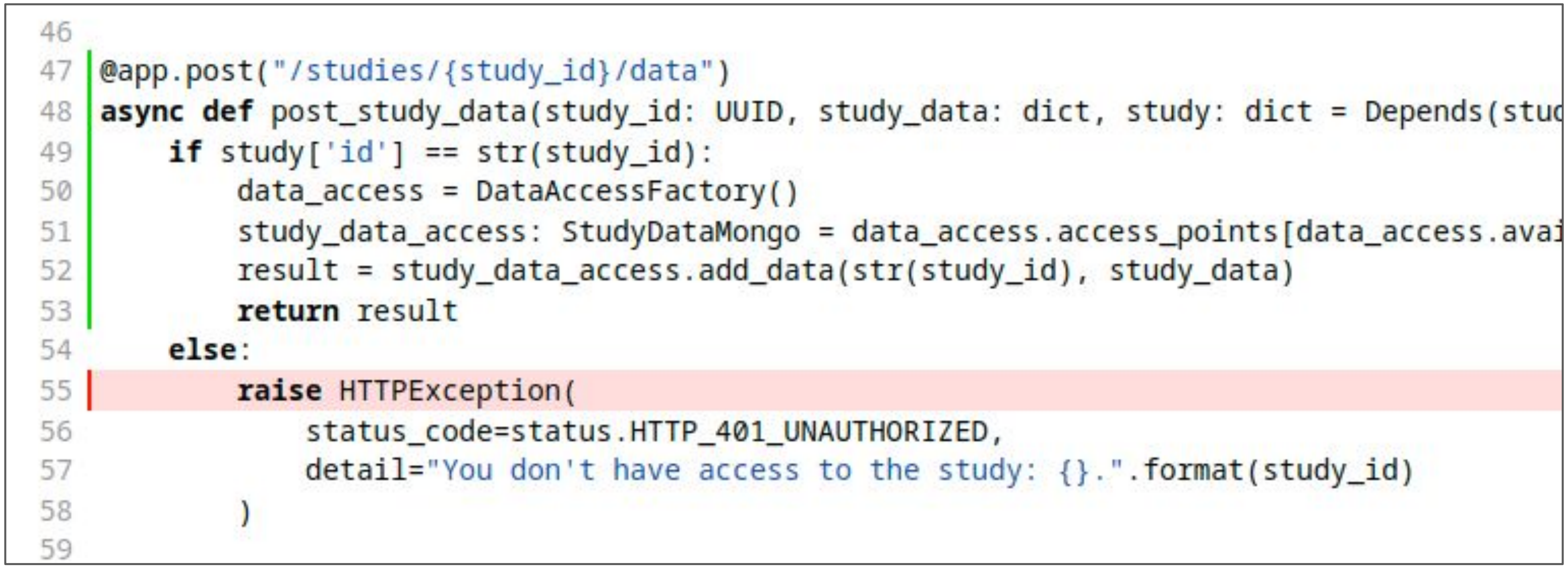
a | b | c

a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

a & b & c

a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Structural code coverage: summary



```
@app.post("/studies/{study_id}/data")
async def post_study_data(study_id: UUID, study_data: dict, study: dict = Depends(stud
    if study['id'] == str(study_id):
        data_access = DataAccessFactory()
        study_data_access: StudyDataMongo = data_access.access_points[data_access.avai
        result = study_data_access.add_data(str(study_id), study_data)
        return result
    else:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="You don't have access to the study: {}.".format(study_id)
        )
```

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