Presentation 11

Asserts and Error Handling

Announcements for Today

Lessons

- **Lesson 13, 14** for today
- **Videos 15.1-15.7** next time

- **Prelim, Oct 18th 7:30-9:00**
  - Material up October 13th
  - Study guide **TOMORROW**
- **Conflict with Prelim time?**
  - Submit to Prelim 1 Conflict assignment on CMS
  - Do not submit if no conflict

Assignments

- Finishing **Assignment 1**
  - We are going to score it
  - Get **one more** chance Sun.
- **Assignment 2** in progress
  - Will grade it by Saturday
  - Solutions posted by Saturday
- **Assignment 3** due next week
  - Just before the exam
  - Same "length" as A1

Lab Today

- There is **no lab** next Tuesday/Wednesday
  - Technically Wednesday is break
  - We will honor this – no class
  - Since no Wednesday lab, not Tuesday either
- But will hold Zoom session Tuesday!
- Today's lab is due Thursday/Friday
  - But get it out of the way soon!
  - Want to focus your time on A3

Assert Statements

```
>>> s = 'l'
>>> assert type(s) == int, str(s)+' is not an int'
```

What is the error message?

A: "'l' is not an int"
B: "l is not an int"
C: 'l' is not an int
D: The error has no message
E: There is no error

Assert Statements

```
>>> s = 'l'
>>> assert type(s) == int, str(s)+' is not an int'
```

What is the error message?

A: "'l' is not an int"
B: "l is not an int" **CORRECT**
C: 'l' is not an int
D: The error has no message
E: There is no error

Assert Statements

```
>>> s = 'l'
>>> assert type(s) == str, repr(s)+' is not a string'
>>> assert ' ' in s, repr(s)+' has no space'
```

What is the error message?

A: "'l' is not a string"
B: "'l' has no space"
C: "l has no space"
D: The error has no message
E: There is no error

Assert Statements

```
>>> s = 'l'
>>> assert type(s) == str, repr(s)+' is not a string'
>>> assert ' ' in s, repr(s)+' has no space'
```

What is the error message?

A: "'l' is not a string"
B: "'l' has no space" **CORRECT**
C: "l has no space"
D: The error has no message
E: There is no error

Brainstorm: Enforcing Preconditions

```
def lookup_netid(nid):
    """Returns: name of student with netid nid.

    Precondition: nid is a string, which consists of
    2 or 3 lowercase letters followed by a sequence
    of digits """
    assert ?????
```

Assert uses expressions only.
Cannot use if-statements.
Each one must fit on one line.

Try-Except: Tracing Control Flow

```
def first(x):
    print('Start first.')
    try:
        second(x)
        print('No errors first')
    except:
        print('Caught first')
    print('Done first')
```

```
def second(n):
    print('Start second.')
    try:
        assert n <= 0, str(n)+' is not <= 0'
        print('No errors second')
    except:
        print('Caught second')
    assert n >= 0, str(n)+' is not >= 0'
    print('Ending second')
```

What is the output of first(1)?

Which One is Closest to Your Answer?

A:
```
'Start first'
'Start second'
AssertionError: '-1 is not <= 0'
```

B:
```
'Start first'
'Start second'
'1 is not <= 0'
'Caught second'
'Done second'
'No errors first'
'Done first'
```

C:
```
'Start first'
'Start second'
'Caught second'
'Done second'
'No errors first'
'Done first'
```

D:
```
'Start first'
'Start second'
'No errors second'
'No errors first'
'Done first'
```

Which One is Closest to Your Answer?

A:
```
'Start first'
'Start second'
AssertionError: '-1 is not <= 0'
```

B:
```
'Start first'
'Start second'
'1 is not <= 0'
```

E:

¯_(ツ)_/¯

C:
```
'Start first'
'Start second'
'Caught second'
'Done second'
'No errors first'
'Done first'
```

Which One is Closest to Your Answer?

A:
```
'Start first'
'Start second'
AssertionError: '-1 is not <= 0'
```

B:
```
'Start first'
'Start second'
'1 is not <= 0'
'Caught second'
'Done second'
'No errors first'
'Done first'
```

C:
```
'Start first'
'Start second'
'Caught second'
'Done second'
'No errors first'
'Done first'
```
CORRECT

D:
```
'Start first'
'Start second'
'No errors second'
'No errors first'
'Done first'
```

Try-Except: Tracing Control Flow

```
def first(x):
    print('Start first.')
    try:
        second(x)
        print('No errors first')
    except:
        print('Caught first')
    print('Done first')
```

```
def second(n):
    print('Start second.')
    try:
        assert n <= 0, str(n)+' is not <= 0'
        print('No errors second')
    except:
        print('Caught second')
    assert n >= 0, str(n)+' is not >= 0'
    print('Ending second')
```

What is the output of first(-1)?

Which One is Closest to Your Answer?

A:
```
'Start first'
'Start second'
'No errors second'
'Caught first'
'Done first'
```

B:
```
'Start first'
'Start second'
'No errors second'
'-1 is not <= 0'
'Caught first'
'Done first'
```

C:
```
'Start first'
'Start second'
'No errors second'
AssertionError: '-1 is not <= 0'
```

D:
```
'Start first'
'Start second'
'No errors second'
'Done second'
'Caught first'
'Done first'
```

Which One is Closest to Your Answer?

A:
```
'Start first'
'Start second'
'No errors second'
'Caught first'
'Done first'
```
CORRECT

B:
```
'Start first'
'Start second'
'No errors second'
'-1 is not <= 0'
'Caught first'
'Done first'
```

C:
```
'Start first'
'Start second'
'No errors second'
AssertionError: '-1 is not <= 0'
```

D:
```
'Start first'
'Start second'
'No errors second'
'Done second'
'Caught first'
'Done first'
```

Try-Except: Tracing Control Flow

```
def first(x):
    print('Start first.')
    try:
        second(x)
        print('No errors first')
    except:
        print('Caught first')
    print('Done first')
```

```
def second(n):
    print('Start second.')
    try:
        assert n <= 0, str(n)+' is not <= 0'
        print('No errors second')
    except:
        print('Caught second')
    assert n >= 0, str(n)+' is not >= 0'
    print('Ending second')
```

What is the output of first(0)?

Which One is Closest to Your Answer?

A:
```
'Start first'
'Start second'
'No errors second'
'No errors first'
'Done first'
```

B:
```
'Start first'
'Start second'
'No errors second'
'Done second'
'No errors first'
'Done first'
```

C:
```
'Start first'
'Start second'
'Caught second'
'No errors first'
'Done first'
```

D:
```
'Start first'
'Start second'
'No errors second'
'Caught first'
'Done first'
```

Which One is Closest to Your Answer?

A:
```
'Start first'
'Start second'
'No errors second'
'No errors first'
'Done first'
```

B:
```
'Start first'
'Start second'
'No errors second'
'Done second'
'No errors first'
'Done first'
```
CORRECT

C:
```
'Start first'
'Start second'
'Caught second'
'No errors first'
'Done first'
```

D:
```
'Start first'
'Start second'
'No errors second'
'Caught first'
'Done first'
```

Brainstorm: Without Conditionals

```
def isnetid(s):
    """Returns: True if s is a potential netid.

    A potential netid is 2 or 3 lowercase letters followed
    by a sequence of digits

    Precondition: nid is a string"""
```

How can we implement this without if-statements?

Brainstorm: Without Conditionals

```
def isnetid(s):
    """Returns: True if s is a potential netid.

    A potential netid is 2 or 3 lowercase letters followed
    by a sequence of digits

    Precondition: nid is a string"""
```

Was this a good idea?

A: Yes
B: No
C: Unsure

Questions?