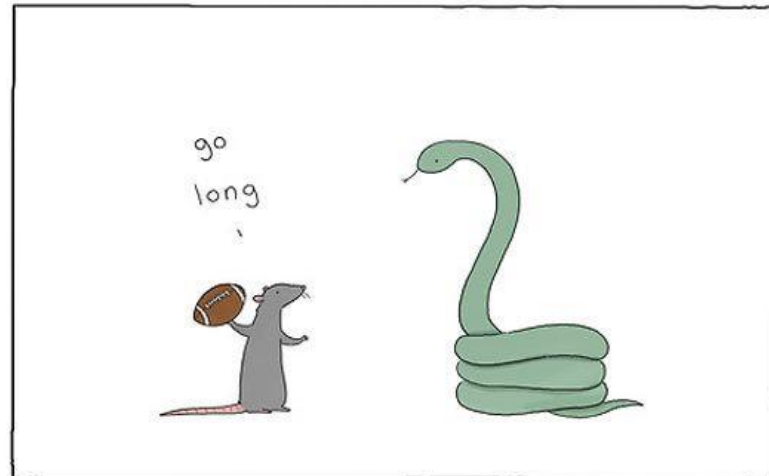# CSc 110, Autumn 2017
Lecture 26: Assertions

Adapted from slides by Marty Stepp and Stuart Reges

lizclimo.tumblr.com

Section attendance question

- Read a file of section attendance (*see next slide*):

```
yynyyynayayynyyyayanyyyaynayyayyanayyyanyayna
ayyanyyyayanaayyanayyyananayayaynyayayynynya
yyayaynyyayyanynnyyyayyanayaynannnyyayyayayny
```

- And produce the following output:

```
Section 1
Student points: [20, 16, 17, 14, 11]
Student grades: [100.0, 80.0, 85.0, 70.0, 55.0]

Section 2
Student points: [16, 19, 14, 14, 8]
Student grades: [80.0, 95.0, 70.0, 70.0, 40.0]

Section 3
Student points: [16, 15, 16, 18, 14]
Student grades: [80.0, 75.0, 80.0, 90.0, 70.0]
```

- Students earn 3 points for each section attended up to 20.

Section input file

student		123451234512345123451234512345123451234512345
week		1 2 3 4 5 6 7 8 9
section	1	yynyyynayayynyyyayanyyyaynayyayyanayyyanyayna
section	2	ayyanyyyyayanaayyanayyyananayayaynyayayynynya
section	3	yyayaynyyayyanynnyyyayyanayaynannnyyayyayayny

- Each line represents a section.
- A line consists of 9 weeks' worth of data.
    - Each week has 5 characters because there are 5 students.
- Within each week, each character represents one student.
    - `a` means the student was absent (+0 points)
    - `n` means they attended but didn't do the problems (+1 points)
    - `y` means they attended and did the problems (+3 points)

Logical assertions

- **assertion**: A statement that is either true or false.

  Examples:
  - Python was created in 1995.
  - The sky is purple.
  - 23 is a prime number.
  - 10 is greater than 20.
  - x divided by 2 equals 7. *(depends on the value of x)*

- An assertion might be false ("The sky is purple" above), but it is still an assertion because it is a true/false statement.

Reasoning about assertions

- Suppose you have the following code:

```
if x >= 3:
    # Point A
    x -= 1
else:
    # Point B
    x += 1
    # Point C
# Point D
```

- What do you know about `x`'s value at the three points?
  - Is `x > 3`? Always? Sometimes? Never?

Assertions in code

- We can make assertions about our code and ask whether they are true at various points in the code.
  - Valid answers are ALWAYS, NEVER, or SOMETIMES.

```
number = input("Type a nonnegative number: ")
# Point A: is number < 0.0 here?                    (SOMETIMES)

while number < 0.0:                                 (ALWAYS)
    # Point B: is number < 0.0 here?
    number = input("Negative; try again: ")


    # Point C: is number < 0.0 here?                (SOMETIMES)


# Point D: is number < 0.0 here?                    (NEVER)
```

Reasoning about assertions

- Right after a variable is initialized, its value is known:
  ```
  x = 3
  # is x > 0?  ALWAYS
  ```

- In general you know nothing about parameters' values:
  ```
  def mystery(a, b):
  # is a == 10?  SOMETIMES
  ```

- But inside an `if`, `while`, etc., you may know something:
  ```
  def mystery(a, b):
      if a < 0:
          # is a == 10?  NEVER
          ...
  ```

Assertions and loops

- At the start of a loop's body, the loop's test must be `True`:
  ```
  while y < 10:
      # is y < 10?  ALWAYS
      ...
  ```

- After a loop, the loop's test must be `False`:
  ```
  while y < 10:
      ...

  # is y < 10?  NEVER
  ```

- Inside a loop's body, the loop's test may become `False`:
  ```
  while y < 10:
      y += 1
      # is y < 10?  SOMETIMES
  ```

"Sometimes"

- Things that cause a variable's value to be unknown
  (often leads to "sometimes" answers):

  - reading from `input`
  - reading a number from a `random` object
  - a parameter's initial value to a function

- If you can reach a part of the program both with the answer being "yes" and the answer being "no", then the correct answer is "sometimes".

  - If you're unsure, "Sometimes" is a good guess.

Assertion example 1

```
def mystery(x, y):
    z = 0
    # Point A

    while x >= y:
        # Point B
        x = x - y
        z += 1

        if x != y:
            # Point C
            z = z * 2
        # Point D
    # Point E
    print(z)
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

	x < y	x == y	z == 0
Point A	SOMETIMES	SOMETIMES	ALWAYS
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	SOMETIMES	NEVER	NEVER
Point D	SOMETIMES	SOMETIMES	NEVER
Point E	ALWAYS	NEVER	SOMETIMES

Assertion example 2

```
def mystery():
    prev = 0
    count = 0
    next = input()
    # Point A

    while next != 0:
        # Point B

        if next == prev:
            # Point C
            count += 1

        prev = next
        next = input()
        # Point D
    # Point E
    return count
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

	next == 0	prev == 0	next == prev
Point A	SOMETIMES	ALWAYS	SOMETIMES
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	NEVER	NEVER	ALWAYS
Point D	SOMETIMES	NEVER	SOMETIMES
Point E	ALWAYS	SOMETIMES	SOMETIMES

Assertion example 3

```
# Assumes y >= 0, and returns x^y
def pow(x, y):
    prod = 1

    # Point A
    while (y > 0):
        # Point B
        if (y % 2 == 0):
            # Point C
            x = x * x
            y = y // 2
            # Point D
        else:
            # Point E
            prod = prod * x
            y -= 1
            # Point F
    # Point G
    return prod
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

	y > 0	y % 2 == 0
Point A	SOMETIMES	SOMETIMES
Point B	ALWAYS	SOMETIMES
Point C	ALWAYS	ALWAYS
Point D	ALWAYS	SOMETIMES
Point E	ALWAYS	NEVER
Point F	SOMETIMES	ALWAYS
Point G	NEVER	ALWAYS