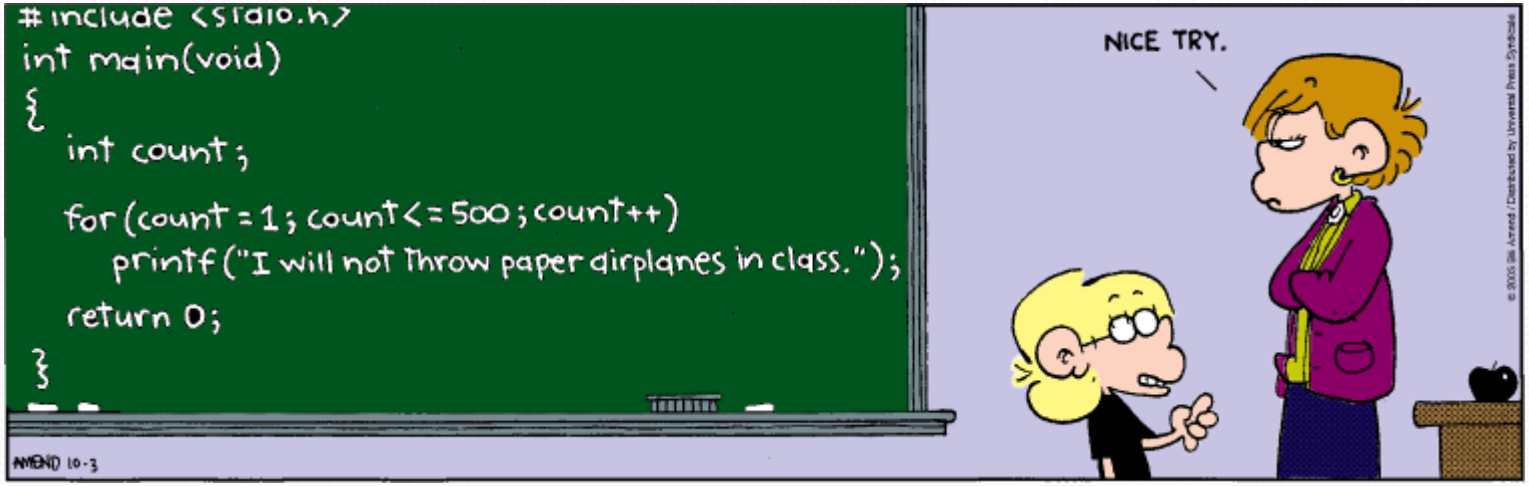
CS 115, Autumn 2021

Lecture 19: `while` loops; `for` loops

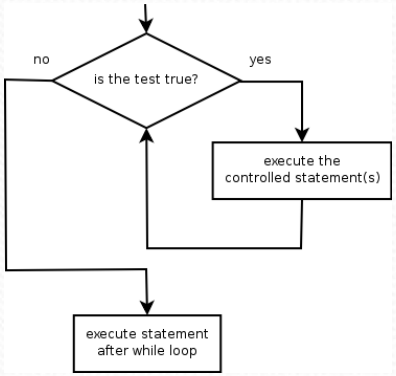


Thanks to Marty Stepp and Stuart Reges for parts of these slides

The `while` loop

- **`while` loop**: Repeatedly executes its body as long as a logical test is true.

```
while test:
    statement(s)
```



- Example:

```
num = 1                                  # initialization
while num <= 200:                        # test
    print(str(num) + " ", end='')
    num = num * 2                        # update

# output:  1 2 4 8 16 32 64 128
```

Exercise: speed up

- Alter the car animation to make it slowly accelerate.
  - Can we do this with `panel.animate(`**ms**, **function**`)`?
    - No – we set the speed once.
  - Instead, use `panel.sleep(`**ms**`)` to make the animation pause for the passed in time.
    - How can we make this repeat?
      - A `while` loop!

Exercise: roll dice

- Write a program that simulates rolling two dice until the numbers shown on them sum to 7. Print out each roll and print out the number of rolls it took.

Example:

```
1 + 4 = 5
3 + 5 = 8
5 + 6 = 11
3 + 3 = 9
1 + 5 = 6
4 + 3 = 7
You got a 7 in 6 rolls.
```

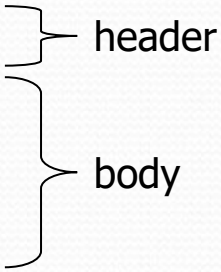
Categories of loops

- **indefinite loop**: One where the number of times its body repeats is not known in advance.
  - The `while` loops we have seen are indefinite loops.
    - Prompt the user until they type a non-negative number.
    - Print random numbers until a prime number is printed.
    - Repeat until the user has typed "q" to quit.

- **definite loop**: Executes a known number of times.
  - The `for` loops we will see next are definite loops.
    - Print "hello" 10 times.
    - Find all the prime numbers up to an integer *n*.
    - Print each odd number between 5 and 127.

`for` loop syntax

```
for variable in range (start, stop):
    statement
    statement
    ...
    statement
```



- Set the variable equal to the start value
- Repeat the following:
  - Check if the **variable** is less than the stop. If not, stop.
  - Execute the **statement**s.
  - Increase the variable's value by 1.

Repetition over a range

```
print("1 squared = " + str(1 * 1))
print("2 squared = " + str(2 * 2))
print("3 squared = " + str(3 * 3))
print("4 squared = " + str(4 * 4))
print("5 squared = " + str(5 * 5))
print("6 squared = " + str(6 * 6))
```

- Intuition: "I want to print a line for each number from 1 to 6"

- The `for` loop does exactly that!

```
for i in range(1, 7):
      print(str(i) "squared = " str(i * i))
```

- "For each integer **i** from 1 through 6, print ..."

Loop walkthrough

```
for i in range(1, 5):
    print(str(i) + " squared = " + str(i * i))

print("Whoo!")
```

Output:

```
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
Whoo!
```

Expressions for counter

```
high_temp = 5
for i in range(-3, high_temp // 2 + 1):
    print(i * 1.8 + 32)
```

- Output:

```
26.6
28.4
30.2
32.0
33.8
35.6
```

Rocket Exercise

- Go backwards by adding a negative increment amount:

```
T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
blastoff!
The end.
```

Counting down

```
for variable in range (start, stop, increment):
        statement
        statement
        …
        statement
```

- The default increment amount is 1

`print('', end='')`

- Adding `,end=''` allows you to print without moving to the next line
  - allows you to print partial messages on the same line

```
high_temp = 5
for i in range(-3, high_temp // 2 + 1):
    print(i * 1.8 + 32, end=' ')
```

  - Output:

```
26.6  28.4  30.2  32.0  33.8  35.6
```

    - Either concatenate `'  '` to separate the numbers or set `end='  '`