Lab 3 Feedback

- Continuing to get tougher in grading
  - Paying more attention to style (e.g., variable names), efficiency, readability, good output
  - High-level descriptions
  - More strict on adhering to problem specification
  - Constants
  - Demonstrate program **more than once** if gets input from user or outcome changes when run again
    - Find errors before I do!

Lab 3: Feedback

- Careful not to do too much in one statement

```
print("Your hydrocarbon's molecular mass is",
round(C*C_WEIGHT+H*H_WEIGHT+O*O_WEIGHT,3),
"g/mol")
```

Instead: 1st: Compute the weight
2nd: Round
3rd: Display

Why do I care?
→ Error prone; More difficult to debug, read

Lab 3: Feedback

- Comments, good variable names on the OO graphics programming
- Compare:

```
# Draw the snow person's body
body2.draw(win)
```

vs

```
# Draw the second circle
circle2.draw(win)
```


Lab Reminders

- Student assistants and professor are here to help BUT you should know how to think/learn
  - Review the slides and examples
  - Narrow down the issues so they're not too broad
  - Find a clear symptom of the error before asking how to fix
- You should NOT simply look at your neighbor's code and write down the answer
  - You can ask your neighbor questions and discuss solutions but copying answers is an honor code violation
- Grappling with a problem is part of the process
  - We will give advice on the process and how to approach problems

Lab Learning Process

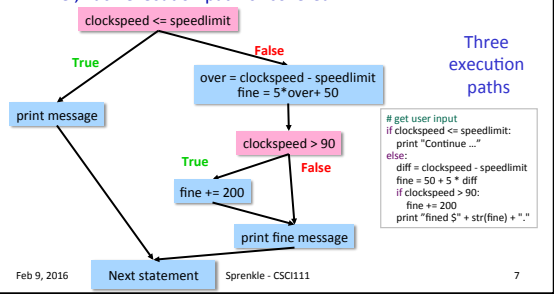
- More struggle does not mean higher grades
- BUT struggle does help you improve your process to arrive at solutions faster
  - What didn't work before? Don't do that again. Try something different
  - What did work before? Repeat that
- Approaches that work
  - Break problem into smaller pieces
    - don't try to solve whole problem at once
  - Good test cases
  - Asking questions: why was that output generated? Why did that behave like that, …

Review

- How do we tell Python to make decisions?
- What is the syntax for such statements?
  - What are the alternatives and their meaning?
- How do you write a condition that is true only if two conditions are true?
- How do you write a condition that is true if at least one of two conditions is true?
- How do you write a condition that is true if a condition is not true?

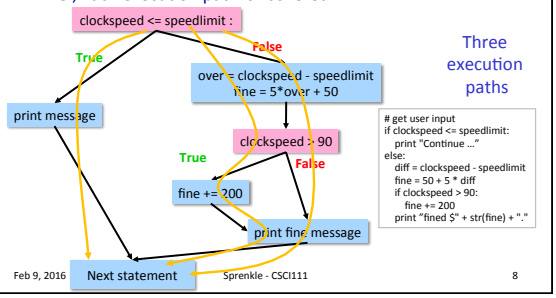
Review: Testing with `if` Statements

- Make sure *at least* have test cases that execute each branch in control flow diagram
  - i.e., Each execution path is "covered"



Review: Testing with `if` Statements

- Make sure have test cases that execute each branch in control flow diagram
  - i.e., Each execution path is "covered"



Review: Efficiency of `if` statements

- Which is more efficient?

```
if x < 0:
   print(x, "is negative")
if x >= 0:
   print(x, "is 0 or positive")
```

```
if x < 0:
   print(x, "is negative")
else:
   print(x, "is 0 or positive")
```


Review: Efficiency of `if` statements

- Which is more efficient?

```
if x < 0:
   print(x, "is negative")
if x >= 0:
   print(x, "is 0 or positive")
```

Additional computation

```
if x < 0:
   print(x, "is negative")
else:
   print(x, "is 0 or positive")
```

More efficient

Extra Credit Opportunity

SYS MODULE

`sys` module

- Has useful ***system*** functions
- Use the `exit([status])` function
  - **Exits the whole program**
  - If status is empty, defaults to 0
  - Status of 0 means success
  - Other values are various failures
- *Another example of changing control flow*

Instead of

```
print("This program determines your birth year")
print("given your age and current year")
print()
age = eval(input("Enter your age: "))

if age > 120:
    print("Don't be ridiculous, you can't be that old.")
else:
    currentYear = eval(input("Enter the current year: "))
    birthyear = currentYear – age
    print()
    print("You were either born in", birthyear, end=‘’)
    print("or", birthyear-1)
```


Example Use of **sys** module

```
import sys

print("This program determines your birth year")
print("given your age and current year")
print()
age = eval(input("Enter your age: "))

if age > 120:
   print("Don't be ridiculous, you can't be that old.")
   sys.exit(1)

# input is reasonable …
currentYear = eval(input("Enter the current year: "))
birthyear = currentYear – age
print()
print("You were either born in", birthyear, end=‘’)
print("or", birthyear-1)
```

Ejector seat

Example Use of **sys** module

```
import sys

print("This program det
print("given your age a
print()
age = eval(input("Enter

if age > 120:
   print("Don't be ridi
   sys.exit(1)

# input is reasonable …
currentYear = eval(input("Enter the current year: "))
birthyear = currentYear – age
print()
print("You were either born in", birthyear, end=‘’)
print("or", birthyear-1)
```

Ejector seat

Why exiting the program on error is considered better:

- Take care of the error cases and exit
- Then, only need to handle correct behavior
- Easier to read if don't have extra column of indentation → matters more as program size increases

Lab 4 Overview

- Conditional problems
- Extra Credit
  - Using sys module