Control flow: if statements

Andrew S. Fitz Gibbon
UW CSE 160
Winter 2022
Making decisions

• How do we compute absolute value?

\begin{align*}
\text{Absolute value of } & \ 5 \ \text{is} \\
\text{Absolute value of } & \ 0 \ \text{is} \\
\text{Absolute value of } & \ -22 \ \text{is}
\end{align*}

\textbf{If the value is negative}, negate it. \textbf{Otherwise}, use the original value.
Absolute value solution

**If** the value is negative, negate it. **Otherwise**, use the original value.

```python
val = -10

# calculate absolute value of val
if val < 0:
    result = -val
else:
    result = val

print(result)
```

In this example, `result` will always be assigned a value.
Absolute value solution

If the value is negative, negate it. Otherwise, use the original value.

```python
val = -10

# calculate absolute value of val
if val < 0:
    result = -val
else:
    result = val

print(result)
```

In this example, `result` will always be assigned a value.

Another approach that does the same thing without using `result`:

```python
val = -10

if val < 0:
    print(-val)
else:
    print(val)
```
Another Absolute value solution

As with loops, a sequence of statements could be used in place of a single statement:

```python
val = -10

# calculate absolute value of val
if val < 0:
    result = -val
    print("val is negative!")
    print("I had to do extra work!")
else:
    result = val
    print("val is positive")
print(result)
```

See in python tutor
Nested if statements

```python
val = 0

# calculate absolute value of val
if val < 0:
    print("val is negative")
    print(val)
    result = -val
else:
    if val == 0:
        print("val is zero")
        print(val)
        result = val
    else:
        print("val is positive")
        print(val)
        result = val

print(result)
```
Use `elif` instead of nested `ifs`

```python
val = 0

# calculate absolute value of val
if val < 0:
    print("val is negative")
    print(val)
    result = -val
elif val == 0:
    print("val is zero")
    print(val)
    result = val
else:
    print("val is positive")
    print(val)
    result = val

print(result)
```

Easier to read than nested if. Equivalent to code on previous slide.
Only if

It is **not required** that anything happens...

```python
val = -10

if val < 0:
    print("negative value!")
```

What happens when `val = 5`?
Another absolute value solution

What happens here?

```python
val = 5

# calculate absolute value of val
if val < 0:
    result = -val
    print("val is negative!")
else:
    for i in range(val):
        print("val is positive!")
    result = val

print(result)
```

See in python tutor
The if body can be any statements

```python
# height is in km
if height > 100:
    print("space")
else:
    if height > 50:
        print("mesosphere")
    else:
        if height > 20:
            print("stratosphere")
        else:
            print("troposphere")
```

Execution gets here only if "height > 100" is false AND "height > 50" is true.
Version 1

```python
# height is in km
if height > 100:
    print("space")
else:
    if height > 50:
        print("mesosphere")
    else:
        if height > 20:
            print("stratosphere")
        else:
            print("troposphere")
```

Execution gets here only if “height <= 100” is true AND “height > 50” is true.

Execution gets here only if “height <= 100” is true.

km above earth: troposphere | stratosphere | mesosphere | space
# height is in km
if height > 100:
    print("space")
else:
    if height > 50:
        print("mesosphere")
    else:
        if height > 20:
            print("stratosphere")
        else:
            print("troposphere")
if height > 50:
    if height > 100:
        print("space")
    else:
        print("mesosphere")
else:
    if height > 20:
        print("stratosphere")
    else:
        print("troposphere")
Version 3 (Best)

```python
if height > 100:
    print("space")
elif height > 50:
    print("mesosphere")
elif height > 20:
    print("stratosphere")
else:
    print("troposphere")
```

ONE of the print statements is guaranteed to execute: whichever condition it encounters **first** that is true
Order Matters

# version 3
if height > 100:
    print("space")
elif height > 50:
    print("mesosphere")
elif height > 20:
    print("stratosphere")
else:
    print("troposphere")

# broken version 3
if height > 20:
    print("stratosphere")
elif height > 50:
    print("mesosphere")
elif height > 100:
    print("space")
else:
    print("troposphere")

Try height = 72 on both versions, what happens?
# incomplete version 3

```python
if height > 100:
    print("space")
elif height > 50:
    print("mesosphere")
elif height > 20:
    print("stratosphere")
```

In this case it is possible that nothing is printed at all, when?
What Happens Here?

# height is in km
if height > 100:
    print("space")
if height > 50:
    print("mesosphere")
if height > 20:
    print("stratosphere")
else:
    print("troposphere")

Try height = 72
The then clause or the else clause is executed

```python
speed = 54
limit = 55
if speed <= limit:
    print("Good job!")
else:
    print("You owe $", speed/fine)
```

What if we change speed to 64?