http://www.cs.cornell.edu/courses/cs1110/2021sp

Lecture 8: Conditionals & Control Flow

(Sections 5.1-5.7)

CS 1110

Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Fan, D. Gries, L. Lee,
S. Marschner, C. Van Loan, W. White]

Announcements

- **Optional 1-on-1** with a staff member to help *just you* with course material. Sign up for a slot on CMS under "SPECIAL: one-on-ones".
- A1 part A first submission due Mar 5 Fri at 11:59pm
- A1 part B first submission due Mar 8 Mon at 11:59pm
- Conditionals—today's topic—not allowed in A1

Conditionals: If-Statements

Format

```
if <boolean-expression>:
        <statement>
        …
        <statement>
```

Example

```
# is there a new high score?
if curr_score > high_score:
        high_score  = curr_score
        print("New high score!")
```

Execution:

if ⟨*boolean-expression*⟩ is true, then execute all of the statements indented directly underneath (until first non-indented statement)

What are Boolean expressions?

Expressions that evaluate to a Boolean value.

```
is_student = True
is_senior = False
num_credits  = 25
```

Boolean operations:

```
if is_student and is_senior:
   print(“Hi senior student!”)
```

Boolean variables:

```
if is_student:
   print(“Hi student!”)
```

Comparison operations:

```
if num_credits > 24:
   print(“Are you serious?”)
```

What gets printed, Round 1

```
a = 0
print(a)
```

```
a = 0
a = a + 1
print(a)
```

```
a = 0
if a == 0:
    a = a + 1
print(a)
```

```
a = 0
if a == 1:
    a = a + 1
print(a)
```

```
a = 0
if a == 0:
    a = a + 1
a = a + 1
print(a)
```

What gets printed? (Question)

```
a = 0
if a == 0:
    a = a + 1
if a == 0:
    a = a + 2
a = a + 1

print(a)
```

A: 0
B: 1
C: 2
D: 3
E: I do not know

Conditionals: If-Else-Statements

Format

```
if <boolean-expression>:
        <statement>
        …
else:
        <statement>
        …
```

Example

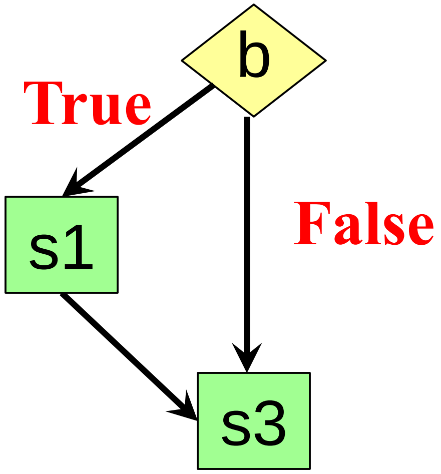
```
# new record?
if curr_score > high_score:
    print("New record!")
else:
    print("Try again next time")
```

Execution:

if ⟨*boolean-expression*⟩ is true, then execute statements indented under **if**; otherwise execute the statements indented under **else**

Conditionals: "Control Flow" Statements

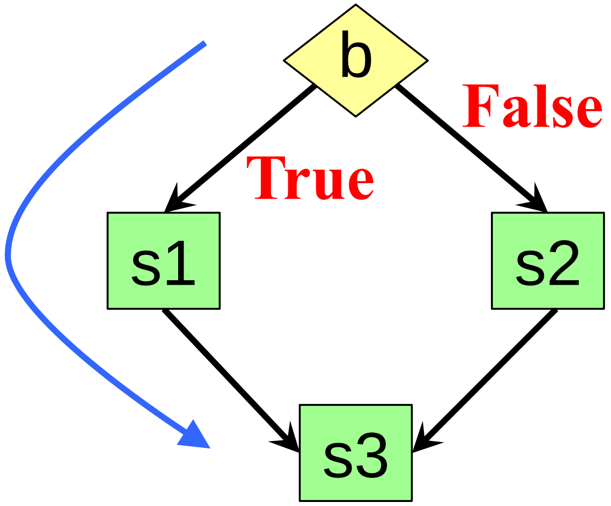
```
if b :
    s1   # statement
s3       # statement
```



Branch Point: Evaluate & Choose ◇

Statements: Execute 🟩

```
if b :
    s1
else:
    s2
s3
```



Flow

Program only takes one path during an execution (something will **not** be executed!)

What gets printed, Round 2

```
a = 0
if a == 0:
    a = a + 1
else:
    a = a + 2
print(a)
```

```
a = 0
if a == 1:
    a = a + 1
else:
    a = a + 2
print(a)
```

```
a = 0
if a == 1:
    a = a + 1
else:
    a = a + 2
a = a + 1
print(a)
```

```
a = 0
if a == 1:
    a = a + 1
else:
    a = a + 1
    a = a + 1
a = a + 1
print(a)
```

Program Flow (car locked, 1)

if determines which statement is executed next

```
   def get_in_car(car_locked):
1      if car_locked:
2          print(“Unlock car!”)
3      print(“Open the door.”)

   car_locked = True
   get_in_car(car_locked)
```

Global Space

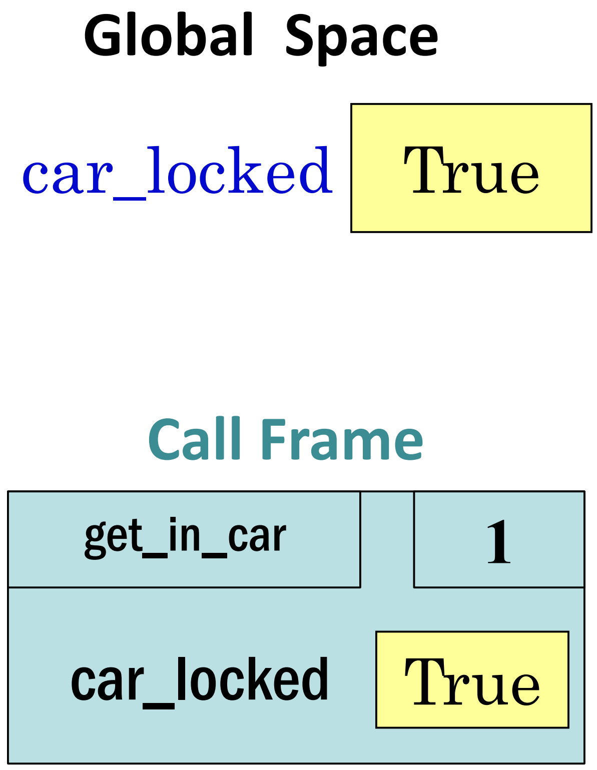
car_locked	True

Program Flow (car locked, 2)

if determines which statement is executed next

```
  def get_in_car(car_locked):
1     if car_locked:
2         print("Unlock car!")
3     print("Open the door.")

  car_locked = True
  get_in_car(car_locked)
```

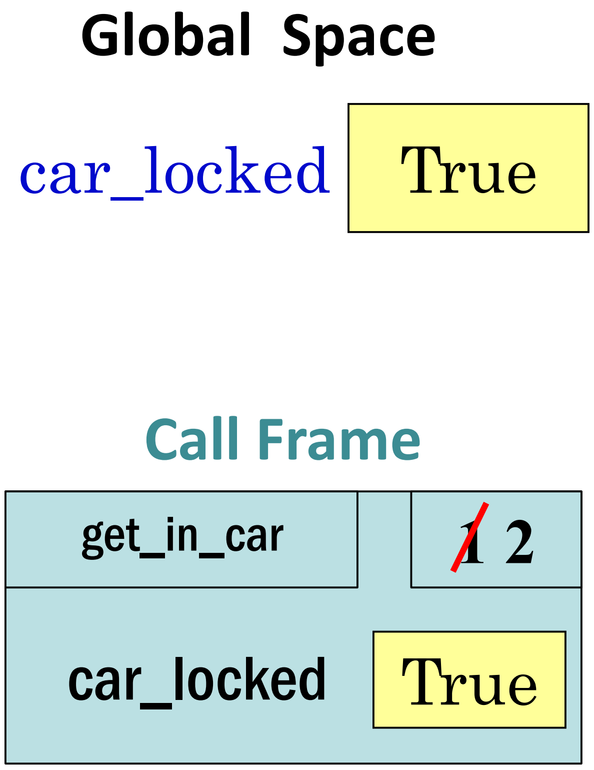


Program Flow (car locked, 3)

if determines which statement is executed next

```
  def get_in_car(car_locked):
1    if car_locked:
2       print(“Unlock car!”)
3    print(“Open the door.”)

  car_locked = True
  get_in_car(car_locked)
```



Program Flow (car locked, 4)

if determines which statement is executed next

```
  def get_in_car(car_locked):
1     if car_locked:
2         print("Unlock car!")
3     print("Open the door.")

  car_locked = True
  get_in_car(car_locked)
```

Global Space

car_locked	True

Call Frame

get_in_car	~~1~~ ~~2~~ 3
car_locked	True

```
Unlock car!
```

Program Flow (car locked, 5)

if determines which statement is executed next

```
  def get_in_car(car_locked):
1     if car_locked:
2         print("Unlock car!")
3     print("Open the door.")

  car_locked = True
  get_in_car(car_locked)
```

Global Space

car_locked	True

Call Frame

get_in_car	~~1~~ ~~2~~ ~~3~~
car_locked	True
RETURN	None

```
Unlock car!
Open the door.
```

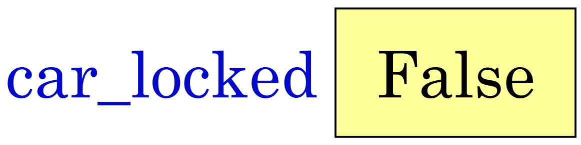
Program Flow (car not locked, 1)

if determines which statement is executed next

```
   def get_in_car(car_locked):
1      if car_locked:
2          print("Unlock car!")
3      print("Open the door.")

   car_locked = False
   get_in_car(car_locked)
```

Global Space

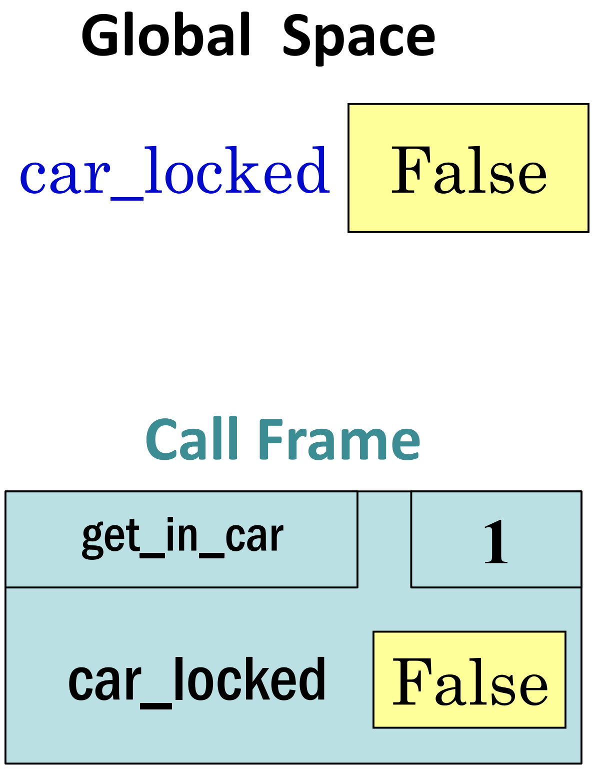


Program Flow (car not locked, 2)

if determines which statement is executed next

```
  def get_in_car(car_locked):
1     if car_locked:
2         print("Unlock car!")
3     print("Open the door.")

  car_locked = False
  get_in_car(car_locked)
```

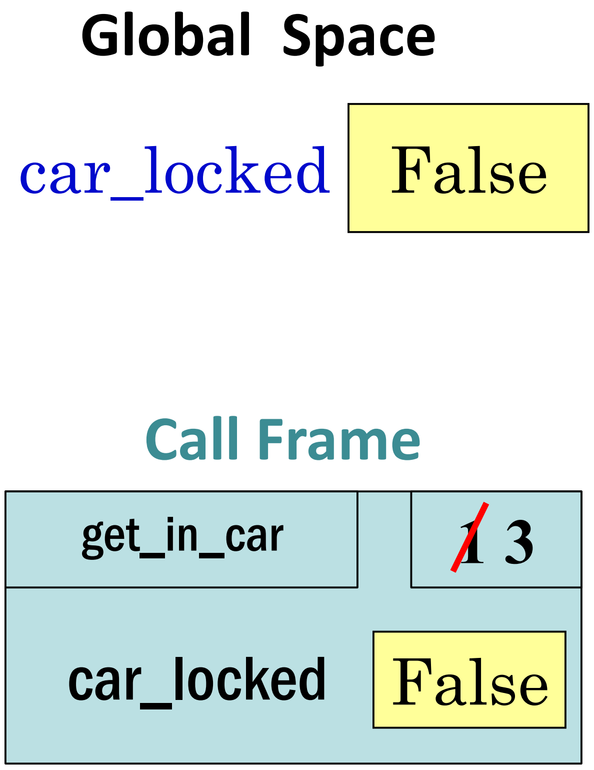


Program Flow (car not locked, 3)

if determines which statement is executed next

```
  def get_in_car(car_locked):
1     if car_locked:
2         print(“Unlock car!”)
3     print(“Open the door.”)

  car_locked = False
  get_in_car(car_locked)
```

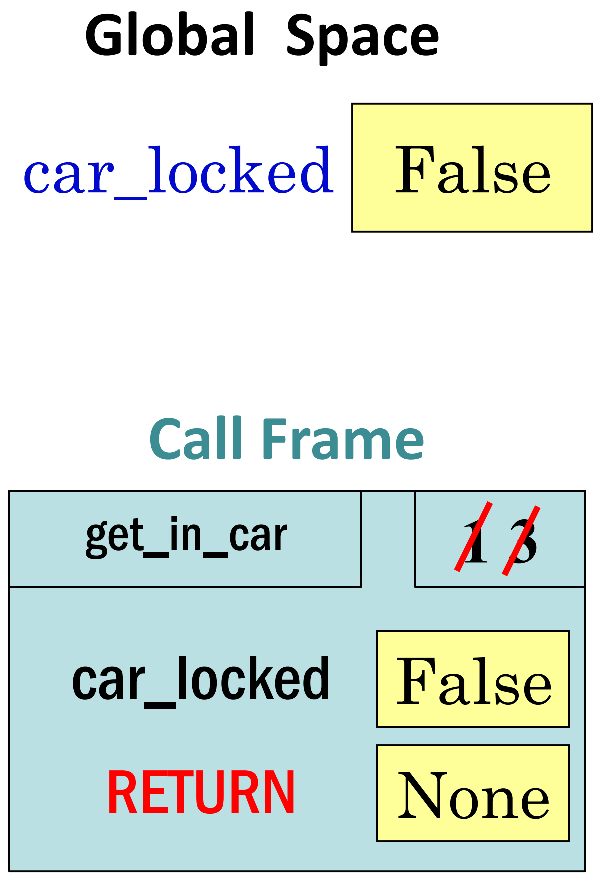


Program Flow (car not locked, 4)

if determines which statement is executed next

```
  def get_in_car(car_locked):
1     if car_locked:
2         print(“Unlock car!”)
3     print(“Open the door.”)

  car_locked = False
  get_in_car(car_locked)
```



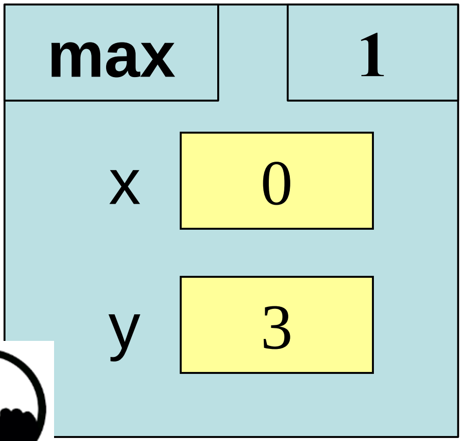
```
Open the door.
```

What does the call frame look like next? (Q)

```
  def max(x,y):
1     if x > y:
2         return x
3     return y

  max(0,3)
```

Current call frame:



Program Flow and Variables

Variables created inside **if** continue to exist past **if**:

```
a = 0
if a == 0:
    b = a + 1
print(b)
```

…but are only created if the program actually executes that line of code

Control Flow and Variables (Q1)

```
def max(x,y):
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger

maximum = max(3,0)
```

Value of maximum?

A: 3
B: 0
C: **Error!**
D: I do not know

Control Flow and Variables (Q2)

```
def max(x,y):
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger

maximum = max(0,3)
```

Value of maximum?

A: 3
B: 0
C: **Error!**
D: I do not know

Program Flow and Variables

```
def zero_or_one(a):
    if a == 1:
        b = 1
    else:
        b = 0
    print(b)
```

make sure that ALL **if** branches create the variable

Conditionals: If-Elif-Else-Statements

Format

```
if <Boolean expression>:
    <statement>
    ...
elif <Boolean expression>:
    <statement>
    ...
...
else:
    <statement>
    ...
```

Example

```
# Find the winner
if score1 > score2:
    winner  = "Player 1"
elif score2 > score1:
    winner  = "Player 2"
else:
    winner = "Players 1 and 2"
```

Conditionals: If-Elif-Else-Statements

Format

```
if <Boolean expression>:
    <statement>
    …
elif <Boolean expression>:
    <statement>
    …
…
else:
    <statement>
    …
```

Notes on Use

- No limit on number of **elif**
  - Must be between **if**, **else**
- **else** is optional
  - if-elif by itself is fine
- Booleans checked in order
  - Once Python finds a true *<Boolean-expression>*, skips over all the others
  - **else** means **all** *<Boolean-expression>* are false

If-Elif-Else (Question)

```
a = 2

if a == 2:
    a = 3
elif a == 3:
    a = 4
print(a)
```

What gets printed?

A: 2
B: 3
C: 4
D: I do not know

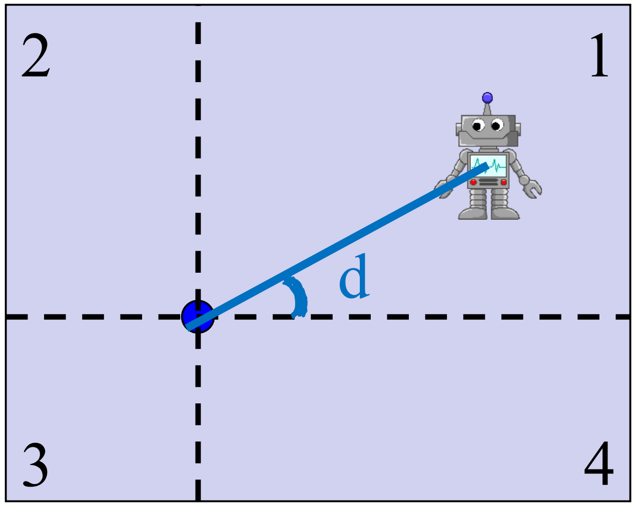
What gets printed, Round 3

```
a = 2
if a == 2:
    a = 3
elif a == 3:
    a = 4
print(a)
```

```
a = 2
if a == 2:
    a = 3
if a == 3:
    a = 4
print(a)
```

Where is the robot?

- Angle of the robot relative to the sensor is d degrees, where d is non-negative
- Robot is in which quadrant?
- To avoid ambiguity, use this convention:
  - 1 if $0 \le d < 90$
  - 2 if $90 \le d < 180$
  - 3 if $180 \le d < 270$
  - 4 if $270 \le d < 360$



WARNING
Robot Operating in Quadrant **1**

Can solve using if-elif-elif… Other options?

Nesting Conditionals

- Separate choices into 2 general categories
- Subdivide each category into subcategories
- Subdivide each subcategory further…

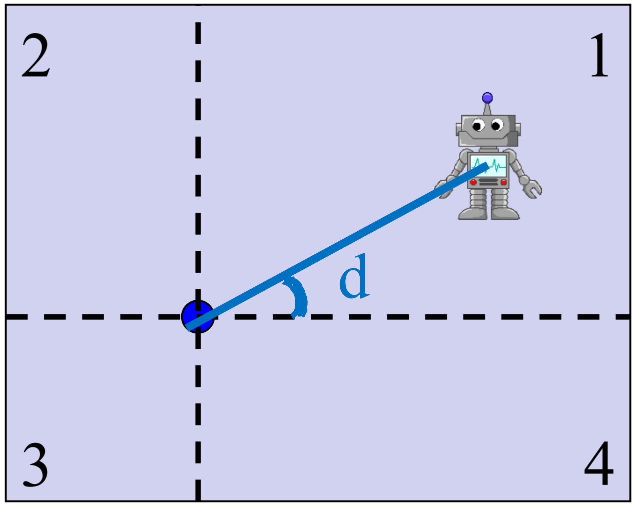
```
if <above x-axis> :
    if <left of y-axis> :

    else:

else:
    if <left of y-axis> :

    else:

```



1	if	$0 \leq d < 90$
2	if	$90 \leq d < 180$
3	if	$180 \leq d < 270$
4	if	$270 \leq d < 360$

See **quadrants.py**

Program Flow and Testing

Can use print statements to examine program flow

```
# Put max of x, y in z

if x > y:
    z = x
else:
    z = y
```

Program Flow and Testing

Can use print statements to examine program flow

```
'before if'
'inside if x>y'
'after if'
```

x must have been greater than y

```
# Put max of x, y in z
print('before if')
if x > y:
    print('inside if x>y')
    z = x
else:
    print('inside else (x<=y)')
    z = y
print('after if')
```

"traces" or "breadcrumbs"

Traces (control) and Watches (data)

```
# Put max of x, y in z
print('before if')
if x > y:
    print('inside if x>y')
    z = x
    print('z = '+str(z))
else:
    print('inside else (x<=y)')
    z = y
    print('z = '+str(z))
print('after if')
```

← TRACES

Trace **program flow**

What code is being executed?

Place them at the beginning of a block of code that might be skipped.

← WATCHES

Watch **data** values

What is the value of a variable?

Place them after assignment statements.