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

```python
if <boolean-expression>:
    <statement>
    ...
    <statement>
```

### Example

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

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

Comparison operations:
if num_credits > 24:
    print(“Are you serious?”)
What gets printed, Round 1

\[
\begin{align*}
\text{a} &= 0 \\
\text{print(}a\text{)} \\
\text{a} &= a + 1 \\
\text{print(}a\text{)} \\
\text{if } a == 0: \\
\text{a} &= a + 1 \\
\text{print(}a\text{)} \\
\text{if } a == 1: \\
\text{a} &= a + 1 \\
\text{print(}a\text{)} \\
\text{if } a == 0: \\
\text{a} &= a + 1 \\
\text{print(}a\text{)}
\end{align*}
\]
What gets printed? (Question)

```python
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 |

print(a)
Conditionals: If-Else-Statements

**Format**

```python
if <boolean-expression>:
    <statement>
    ...
else:
    <statement>
    ...
```

**Example**

```python
# 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

if b:
    True
    s1
    False
    s3

else:
    True
    s1
    False
    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 + 1
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):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")

car_locked = True
get_in_car(car_locked)
Program Flow (car locked, 2)

if determines which statement is executed next

def get_in_car(car_locked):
    if car_locked:
        print(“Unlock car!”)
        print(“Open the door.”)
    car_locked = True

car_locked = True
get_in_car(car_locked)
Program Flow (car locked, 3)

if determines which statement is executed next

```python
def get_in_car(car_locked):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")
    car_locked = True
get_in_car(car_locked)
```

Global Space
- car_locked: True

Call Frame
- get_in_car: 1, 2
- car_locked: True
def get_in_car(car_locked):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")
    car_locked = True
get_in_car(car_locked)
Program Flow (car locked, 5)

if determines which statement is executed next

```python
def get_in_car(car_locked):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")
        car_locked = True
    get_in_car(car_locked)
```

Unlock car!
Open the door.
**Program Flow (car not locked, 1)**

If determines which statement is executed next.

```python
def get_in_car(car_locked):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")
    car_locked = False

global_space
```

Global Space

<table>
<thead>
<tr>
<th>car_locked</th>
<th>False</th>
</tr>
</thead>
</table>

car_locked = False

global_space

23
Program Flow (car not locked, 2)

if determines which statement is executed next

def get_in_car(car_locked):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")
    car_locked = False
get_in_car(car_locked)
if determines which statement is executed next

def get_in_car(car_locked):
    if car_locked:
        print("Unlock car!")
        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):
    if car_locked:
        print("Unlock car!")
        print("Open the door.")
        car_locked = False
    get_in_car(car_locked)

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

```python
def max(x,y):
    if x > y:
        return x
    return y
```

`max(0,3)`

Current call frame:

<table>
<thead>
<tr>
<th>max</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
</tbody>
</table>
Variables created inside if continue to exist past if:

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

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

```python
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
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

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

When run, this code will print 4, as it first sets `a` to 2, then changes it to 3 when `a == 2` is true, and finally changes it to 4 when `a == 3` is true. The print statement at the end then outputs the final value of `a`, which is 4.
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 $0 \leq d < 90$
  - 2 if $90 \leq d < 180$
  - 3 if $180 \leq d < 270$
  - 4 if $270 \leq d < 360$

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

WARNING
Robot Operating in Quadrant 1
Nesting Conditionals

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

```python
if <above x-axis>:
    if <left of y-axis>:
        # Code
    else:
        # Code
else:
    if <left of y-axis>:
        # Code
    else:
        # Code
See quadrants.py
```

- 1 if $0 \leq d < 90$
- 2 if $90 \leq d < 180$
- 3 if $180 \leq d < 270$
- 4 if $270 \leq d < 360$
Can use print statements to examine program flow

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

if x > y:
    z = x
else:
    z = y
```
Can use print statements to examine program flow

`before if`

`inside if x>y`

`after if`

# 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')

x must have been greater than y

“traces” or “breadcrumbs”
# 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')