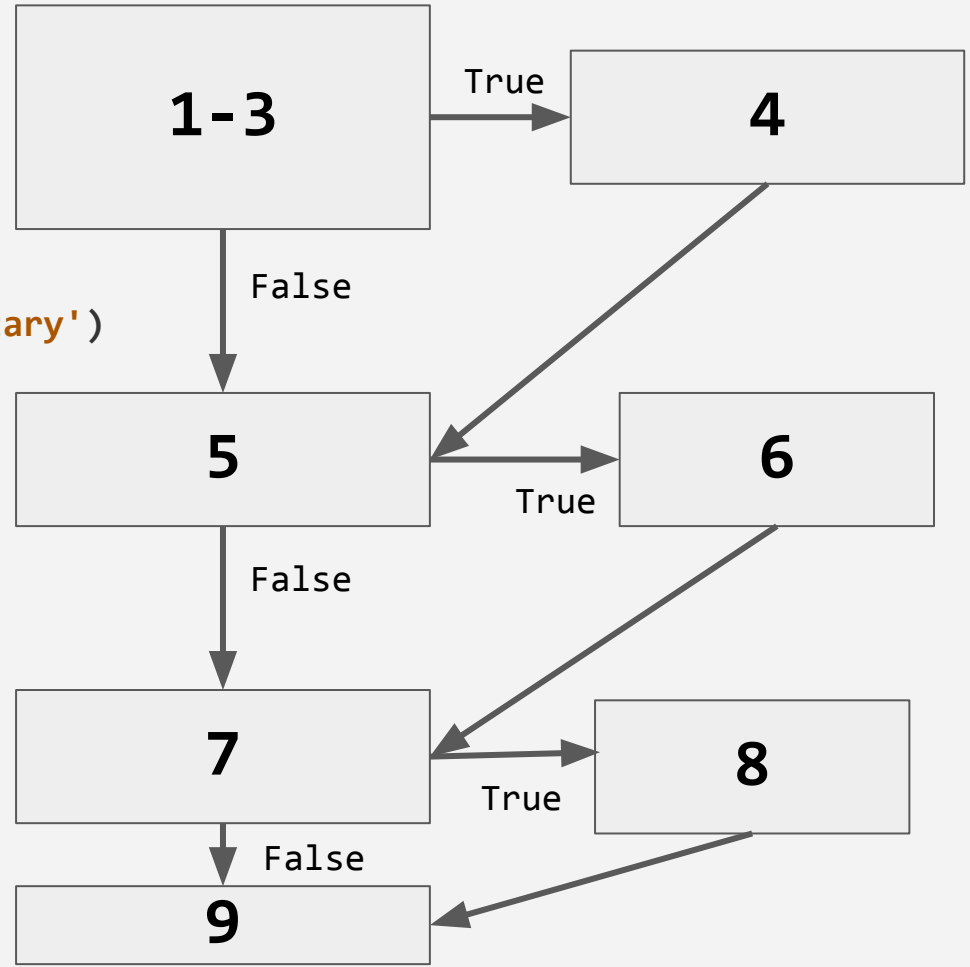
CS 110
If-statements

Adriana Picoral (she, her, hers)
adrianaps@arizona.edu
Gould-Simpson 829

Announcements

- No class Monday (labor day) (no office hours either)
- AP 2 (Star Wars) due on Tuesday
- Videos (content from videos is included in exams)

```
1 age = int(input('How old are you? \n'))
2
3 if age >= 18:
4    print('You may apply to join the military')
5 if age >= 21:
6    print('You may drink')
7 if age > 35:
8    print('You may run for president')
9
```



Checking for numbers

- You can use the function **isnumeric()** to determine if a string represents a number
- Makes sure a string contains only digits

Checking for numbers

- You can use the function **isnumeric()** to determine if a string represents a number
- Makes sure a string contains only digits
- For example:

```
name = 'Jimmy'
name.isnumeric()
```

```
age = 37
age.isnumeric()
```

Ifs and Prints

What, if any, input values would cause this program to print out no text?

```
par = int(input('Golf hole par: '))
swings = int(input('Swings on hole: '))

if swings >= 7:
    print('Go get golf lessons')
if swings == par:
    print('On Par!')
if swings <= (par-1):
    print('Wow, under par!')
```

Exiting a Program

- Sometime, when you encounter a problem or a bad input value, you want your program to exit, without running any additional code
- How?

Exiting a Program

- Sometime, when you encounter a problem or a bad input value, you want your program to exit, without running any additional code
- How?

```
exit()
```

Stops the program

Exiting a Program

- Sometime, when you encounter a problem or a bad input value, you want your program to exit, without running any additional code
- How?

```
1 print('antelope')
2 exit()
3 print('artichoke')
```

Stop the program,

Ifs and Prints

Modify so that if the input values are not numeric, the program will print a message and then exit

```
par = int(input('Golf hole par: '))
swings = int(input('Swings on hole: '))

if swings >= 7:
    print('Go get golf lessons')
if swings == par:
    print('On Par!')
if swings <= (par-1):
    print('Wow, under par!')
```

```
par = input('Golf hole par: ')
if par.isnumeric() != True:
    print('Invalid par')
    exit()
par = int(par)

swings = input('Swings on hole: ')
if swings.isnumeric() != True:
    print('Invalid swing amount')
    exit()
swings = int(swings)
```

```
if swings >= 7:
    print('Go get golf lessons')
if swings == par:
    print('On Par!')
if swings <= (par-1):
    print('Wow, under par!')
```

```
par = input('Golf hole par: ')
if par.isnumeric() != True:
    print('Invalid par')
    exit()
par = int(par)
swings = input('Swings on hole: ')
if swings.isnumeric() != True:
    print('Invalid swing amount')
    exit()
swings = int(swings)

if swings >= 7:
    print('Go get golf lessons')
if swings == par:
    print('On Par!')
if swings <= (par-1):
    print('Wow, under par!')
```

What if we want to add an additional restriction?

Perhaps, the par must less than 7 and greater than 1.

Try it!

```
par = input('Golf hole par: ')
if par.isnumeric() != True:
    print('Invalid par')
    exit()
par = int(par)
if par < 1:
    print('Enter a realistic par.')
    exit()
if par > 6:
    print('Enter a realistic par.')
    exit()

# (also get swing count)

if swings >= 7:
    print('Go get golf lessons')
if swings == par:
    print('On Par!')
if swings <= (par-1):
    print('Wow, under par!')
```

What if we want to add an additional restriction?

Perhaps, the par must less than 7 and greater than 1.

Try it!

Relational Operators

What are they?

Relational Operators

- Can be used to compare two variables or expressions
- Compare the **left-hand side** with the **right-hand side**, and then evaluate to either **True** or **False**

Relational Operators

- Can be used to compare two variables or expressions
- Compare the **left-hand side** with the **right-hand side**, and then evaluate to either **True** or **False**

`==` `!=` `>=` `>` `<=` `<`

Combining conditions with **and**

- You can place the **and** keyword in-between two expressions that evaluate to a boolean (True or False)
- The **and** will combine the two sides, and will result in **True** only if both sides are also **True**.
  - Otherwise **False**

A **and** B

		B	
		True	False
A	True	True	False
	False	False	False

What would it print?

```
a = (4 > 4) and (3 <= 7)
b = a and ((5 - 12) != 4)
print(a and b)
```

Combining conditions with **or**

- You can place the **or** keyword in-between two expressions that evaluate to a boolean (True or False)
- The **or** will combine the two sides, and will result in **False** only if both sides are also **False**.
    - Otherwise **True**

A **or** B

		B	
		True	False
A	True	True	True
	False	True	False

What would it print?

```
a = (4 == 4) or (3 > 7)
b = a and False or True
print(a or b)
```

Duplication

Notice the duplication

Can this code be made more compact?

```
if par < 1:
    print('Enter a realistic par.')
    exit()
if par > 6:
    print('Enter a realistic par.')
    exit()
```

Activity

```
par = input('Golf hole par: ')
if par.isnumeric() != True:
    print('Invalid par')
    exit()
par = int(par)
# . . .
if par < 1:
    print('Enter a realistic par.')
    exit()
if par > 6:
    print('Enter a realistic par.')
    exit()

if swings >= 7:
    print('Go get golf lessons')
if swings == par:
    print('On Par!')
if swings <= (par-1):
    print('Wow, under par!')
```

How could this code be made more compact using the and / or boolean operators?

Activity

```
par = input('Golf hole par: ')
if par.isnumeric() != True:
    print('Invalid par')
    exit()
par = int(par)
# . . .
if par > 6 or par < 1:
    print('Enter a realistic par.')
    exit()

if swings >= 7:
    print('Go get golf lessons')
if swings == par:
    print('On Par!')
if swings <= (par-1):
    print('Wow, under par!')
```

if-else

Use the `if` statement to execute code in the ***True*** case

Can add an `else` to specify code to run in the ***False*** case

```
statements . . .

if condition:
    statement 1a
    statement 2a
    . . .
    statement Na
else:
    statement 1b
    statement 2b
    . . .
    statement Nb

statements . . .
```

if-else

What pair of inputs would cause *only* "**Workload OK**" to print ?

```
title = input('Job title: ')
hours = int(input('Hours worked per week: '))

if title != 'manager' and hours > 20:
    if title == 'waiter':
        print('Workload too high')
    else:
        print('Workload OK')
if hours > 40:
    print('Reduce hours')
else:
    print('Acceptable hours')
```