CS 152: Conditionals

CS 152: Python for STEM

Colorado State University
Computer Science Department
Slides Originally Created by Albert Lionelle and Updated by Marcia Moraes

Colorado State University

Weekly Announcements!

TODO Reminders:

- Reading 3 (zyBooks) – you already should have done that for Monday's class ☺
- Lab 01 – Warm Up
- Reading 4 (zyBooks) – you already should have done that for today's class ☺
- Lab 02 - Application
- Reading 5 (zyBooks)
- RPA 2

Recall Activity

- Identify how many functions are in the code below
- List their names and parameters
- Identify where the functions are being called

```
def name_last_name(name, last_name):
    return name + " " + last_name

def greetings(msg, name, last_name):
    print(msg + " " + name_last_name(name, last_name) + "!")

msg = input("Enter the greetings message: ")
name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
greetings(msg, name, last_name)
```

Recall Activity - Solution

- Identify how many functions are in the code below: 2 functions
- List their names and parameters: name_last_name has name and last_name as parameters; greetings has msg, name, and last_name as parameters
- Identify where the functions are being called

```
def name_last_name(name, last_name):
    return name + " " + last_name

def greetings(msg, name, last_name):
    print(msg + " " + name_last_name(name, last_name) + "!")

msg = input("Enter the greetings message: ")
name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
greetings(msg, name, last_name)
```

Basic Conditionals

- Logic that evaluates as
  - Yes or No
  - True or False (called a Boolean)

- Essential in all programming languages
  - You mentally do this all the time
  - 100 pennies greater than $1?

- Common logic operators
  - == Equals
  - < Less than (is left less than right)
  - > Greater than
  - <= Less than OR equal
  - >= Greater than OR equal
  - != not equal (! is your NOT character)

Structure of if statements

- if without else

```
def get_happy(puppies):
    happy = False
    if puppies >= 100:
        happy = True
    return happy
```

- if with else

```
def get_happy2(puppies):
    if puppies >= 100:
        happy = True
    else:
        happy = False
    return happy
```

```
# conditions are operations, so you can return the result
def get_happy3(puppies):
    return puppies >= 100
```

Coding Practice

- Complete the following code

```
def age_check(age):
        #TODO - add the code necessary to test and return True or
        # False, you  do not print here!


print(age_check(21)) # prints True
print(age_check(20)) # prints False
print(age_check(22)) # prints True
print(age_check(18)) # prints False
```

Elif – Part 1

- Used for chaining if statements
- Let's analyze the following code

```
def verify_number(number):
  if number > 0:
        print(f'Positive number: {number:d}')
  if number < 0:
        print(f'Negative number: {number:d}')
  if number == 0:
        print("Number 0")

verify_number(10)
verify_number(-1)
verify_number(0)
```

How many tests are done each time we call verify_number?

Elif – Part 2

```
def verify_number(number):
    if number > 0:
        print(f'Positive number: {number:d}')
    elif number < 0:
        print(f'Negative number: {number:d}')
    else:print("Number 0")

verify_number(10)
verify_number(-1)
verify_number(0)
```

How many tests are done each time we call verify_number now?

Elif – Coding Practice 1

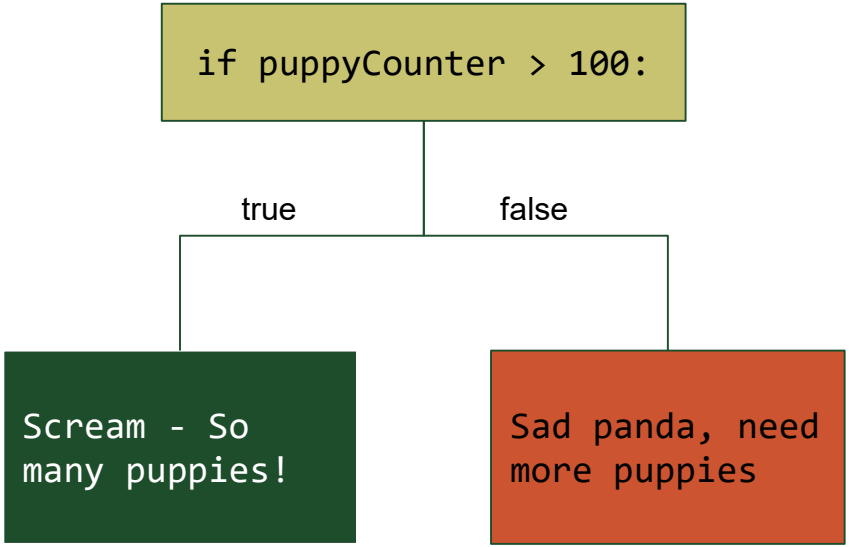
```
def broken_rogue(dice_roll):
    if dice_roll >= 10:
        if dice_roll > 15:
            print("Trap Disarmed!")
        else:
            print("Get the 10-foot pole...")
    elif dice_roll >= 5:
        print("As far as I am aware, no traps.")
    else:
        print("Found the trap!")

broken_rogue(10) # prints Get the 10-foot pole...
broken_rogue(16) # Trap Disarmed!
broken_rogue(5)  # As far as I am aware, no traps.
broken_rogue(1)  # Found the trap!
```

What is printed in each call?

Secret Ninja Logic Trick

- Work it out!
- Draw it out - flow chart!
- Really - just that
  - Often we over think it

Can you code based on the tree?



Coding Practice 2

```
def age_check_by_region(age, region):
   #TODO - add  multiple lines of code here


age_check_by_region(21,  "USA") # prints "OK to buy"
age_check_by_region(20, "USA") # prints Not OK
age_check_by_region(20, "EURO") # prints OK to buy
age_check_by_region(18, "EURO") # prints OK to buy
age_check_by_region(17, "EURO") # prints Not OK
### The  following is true for anything else you put in besides USA or EURO
age_check_by_region(25, "YOLO") #  prints Not  OK
```