CSE 503
Introduction to Computer Science for Non-Majors

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 06
Python Exercises

Announcements

- Labs meet tomorrow (9/13)
  - Practice Lab and Lab #1 posted to course website
    - Practice lab not worth credit, but will teach you how to submit labs
    - Lab #1 due 9/26, autograder will open next week
- Office hours cancelled today

Recap

- Boolean expressions: Expressions that evaluate to **`True`** or **`False`**
  - Operators: **`or`**, **`and`**, **`not`**
- Selection statements allow our programs to make decisions
  - Use boolean expressions to determine whether or not to execute a block of code
  - Keywords: **`if`**, **`elif`**, **`else`**
  - Indentation is now even more important!

Comments and Assertions

- As our programs become larger (and more complex) we need to be able to understand them (ourselves, and others reading them)
- Comments allow us to document what our program is doing
  - Start with '#' in Python
  - Python ignores everything after the '#'
  - Good comments don't just describe the *what*. The *how* and *why* is more important.

Comments and Assertions

- Assertions allow us to tell Python assumptions we have about how our program should work
- If these assumptions are not true, Python will let us know
- Useful for documentation, and testing
- General form:

```
assert <expression> [,<expression>]
```

Comments and Assertions

- Assertions allow us to tell Python assumptions we have about how our program should work
- If these assumptions are not true, Python will let us know (and halt)
- Useful for documentation, and testing
- General form:

```
assert <expression> [,<expression>]
```

keyword

boolean expression

Optional expression. Output if the boolean expression is **`False`**.

Examples

```
def read_file(file, size):
    assert size > 0, "Error: file size must be a positive number"
    # If size is larger than 64 bytes we need to allocate more
    # space first so that we don't overflow memory.
    if size > 64:
        allocate_space();
    open(file)
    …
```

Examples

```
def read_file(file, size):
    assert size > 0, "Error: file size must be a positive number"
    # If size is larger than 64 bytes we need to allocate more
    # space first so that we don't overflow memory.
    if size > 64:
        allocate_space();
    open(file)
    …
```

The assert statement checks our assumption that the file size is a positive number. If it isn't, then something has gone wrong.

Examples

```
def read_file(file, size):
    assert size > 0, "Error: file size must be a positive number"
    # If size is larger than 64 bytes we need to allocate more
    # space first so that we don't overflow memory.
    if size > 64:
        allocate_space();
    open(file)
    …
```

This comment explains to people reading our code why we need this if statement, and what it's purpose is.

Exercise #1

Assume we have a standard deck of playing cards.

Write a function named **`color`** that returns the color of a card based on the suit of the card.

Assume the function takes a single string, and that the string passed in corresponds to the suit of the card: "Clubs", "Diamonds", "Hearts" or "Spades"

For example, **`color("Clubs")`** should return **`"black"`**.

Answers in replit...

Exercise #2

Write a function named **`name`** that takes the numerical value of a card and returns a string corresponding to the name of the card.

For example, **`face(12)`** would return **`"Queen"`**.

If the card does not have a special name, the function should just return the number as a string.

For example, **`face(9)`** would return **`"9"`**. *Reminder: str(x) converts x to a string…*

Write some tests with **`assert`** to verify your assumptions.

Answers in replit…

Exercise #3

Time to put it all together!

Define a function named **`description`**, which takes a numerical value, and a suit, and returns a description of the card.

For example, **`description(12, "Clubs")`** should return:

```
"The Queen of Clubs is black!"
```

Exercise #3

Time to put it all together!

Define a function named **`description`**, which takes a numerical value, and a suit, and returns a description of the card.

For example, **`description(12, "Clubs")`** should return:

```
"The Queen of Clubs is black!"
```

Hint: Can you use the previous exercises to help?

Answers in replit...