CSC 111:
Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson

Admin

- No Lab 4 due to Rally Day
- Homework 3 due Tuesday (tomorrow)
- Office hours today 3-5pm (Ford 355 or across the hall)

Outline: 2/20

WEEK 4: all about FUNCTIONS

- Recap last time
- Formalize our discussion of functions
- Parameters
- Functions calling functions
- Return statements

Recap

Informal quiz (discuss with a partner)

1) Slicing can be done with:

   A) strings B) lists C) both

2) The product of slicing is called a _____, but its type is _____.

3) Which have more built-in methods, strings or lists?

4) Describe the difference between the elements of strings and lists.

5) Give the phrase below, how could you produce just "Rally" or just "Day" using string slicing?

```
phrase = "Rally Day"
```

Informal quiz (discuss with a partner)

1) Slicing can be done with:

   A) strings B) lists **C) both**

2) The product of slicing is called a _____, but its type is _____.

3) Which have more built-in methods, strings or lists?

4) Describe the difference between the elements of strings and lists.

5) Give the phrase below, how could you produce just "Rally" or just "Day" using string slicing?

```
phrase = "Rally Day"
```

Informal quiz (discuss with a partner)

1) Slicing can be done with:

   A) strings B) lists **C) both**

2) The product of slicing is called a **substring**, but its type is **str (string)**.

3) Which have more built-in methods, strings or lists?

4) Describe the difference between the elements of strings and lists.

5) Give the phrase below, how could you produce just "Rally" or just "Day" using string slicing?

```
phrase = "Rally Day"
```

Informal quiz (discuss with a partner)

1) Slicing can be done with:

   A) strings B) lists **C) both**

2) The product of slicing is called a **substring**, but its type is **str (string)**.

3) Which have more built-in methods, strings or lists?

Strings have more built-in methods (strings are more "special", lists are more "flexible").

4) Describe the difference between the elements of strings and lists.

5) Give the phrase below, how could you produce just "Rally" or just "Day" using string slicing?

```
phrase = "Rally Day"
```

Informal quiz (discuss with a partner)

1) Slicing can be done with:

   A) strings B) lists **C) both**

2) The product of slicing is called a **substring**, but its type is **str (string)**.

3) Which have more built-in methods, strings or lists?

Strings have more built-in methods (strings are more "special", lists are more "flexible").

4) Describe the difference between the elements of strings and lists.

The elements of strings are characters, the elements of lists can be (almost) anything.

5) Give the phrase below, how could you produce just "Rally" or just "Day" using string slicing?

```
phrase = "Rally Day"
```

Why use functions?

Several advantages to functions

- **Organizes code into logically connected sections**
  - Easier for a reader or user to understand
  - Easier for the programmer to debug and maintain

Credit: Joe O'Rourke

Several advantages to functions

- **Organizes code into logically connected sections**
  - Easier for a reader or user to understand
  - Easier for the programmer to debug and maintain
- **Avoids repetition, duplication**

Credit: Joe O'Rourke

Several advantages to functions

- **Organizes code into logically connected sections**
  - Easier for a reader or user to understand
  - Easier for the programmer to debug and maintain
- **Avoids repetition, duplication**
- **Isolates one task from another by passing to the function exactly the data needed**
  - Need-to-know basis

Credit: Joe O'Rourke

Several advantages to functions

- **Organizes code into logically connected sections**
  - Easier for a reader or user to understand
  - Easier for the programmer to debug and maintain
- **Avoids repetition, duplication**
- **Isolates one task from another by passing to the function exactly the data needed**
  - Need-to-know basis
- **To document separately with doc-strings (more later)**

Credit: Joe O'Rourke

Recipe Idea: Example

Here we have **4 "helper"** functions and 1 "main" function that call (invokes) the helper functions

Image: Public Radio International

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Recipe Idea: Example

Functions are defined with **"formal"** parameters. Think of these as variables without values yet, or "placeholders".

These formal parameters are only valid inside the function where they are used. They are therefore **"local"** variables and theirs **"scope"** is inside the function.

Zelle: Section 6.4

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Recipe Idea: Example

Before calling our helper functions, we need to assign the necessary variables some concrete, specific values.

Then we can call/invoke the helper functions with the **"actual"** parameters.

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Zelle: Section 6.4

Recipe Idea: Example

Idea: function -> minion (Dominique)

Image: Minions (2015) Illumination Entertainment

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Recipe Idea: Example

Idea: function -> minion (Dominique)

Image: Minions (2015) Illumination Entertainment

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Recipe Idea: Example

Idea: function -> minion (Dominique)

Image: Minions (2015) Illumination Entertainment

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Recipe Idea: Example

Idea: function -> minion (Dominique)

Image: Minions (2015) Illumination Entertainment

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Recipe Idea: Example

Idea: function -> minion (Dominique)

Image: Minions (2015) Illumination Entertainment

```
# Making cookies in Python (pseudo-code)
# Yield: 4 dozen cookies

def mix(item1, item2):
    # combine item1 and item2 in a bowl
    return mixture

def spoon(batter):
    # place a spoonful of batter on cookie sheet
    return cookie

def bake(cookie):
    # bake in oven at 375 deg

def ice(cookie):
    # spread icing on cookie with knife
    # decorate with sprinkles

def main():

    flour = "my_flour"
    sugar = "domino_sugar"
    butter = "land_o_lakes_butter"
    eggs = "free_range_eggs"

    dry_ingredients = mix(flour, sugar)
    wet_ingredients = mix(butter, eggs)

    batter = mix(dry_ingredients, wet_ingredients)

    for i in range(12*4):
        cookie = spoon(batter)
        bake(cookie)
        ice(cookie)

    print("finished making cookies")

main()
```

Verb/Noun idea

- Function names are often action oriented (i.e. verbs)
  - compute, count, mix, bake
- Variable names are often object oriented (i.e. nouns)
  - cookie, name, level, x

Back to count example (Lab 3)

Count: function with two parameters

Function definition with two formal parameters (placeholders for ***any*** string and ***any*** character)

```
def count(string, character):

    char_count = 0
    for c in string:
        if c == character:
            char_count = char_count + 1

    print(char_count)
```

We can call this function inside our .py file ***or*** in the shell

```
count("hello","l")
count("Mississippi","i")
```

Back to sea level (HW 2)

Separate this assignment into functions

Goal Before

```
Enter rate (mm/yr) = 3
Enter acceleration (mm/yr^2) = 0.1
Enter # of years = 8

year=0 rate=3.0 level=0.0
year=1 rate=3.1 level=3.0
year=2 rate=3.2 level=6.1
year=3 rate=3.3 level=9.3
year=4 rate=3.4 level=12.6
year=5 rate=3.5 level=16.0
year=6 rate=3.6 level=19.5
year=7 rate=3.7 level=23.1
year=8 rate=3.8 level=26.8
```

Goal After

```
>>> compute_level(3,0.1,8)
26.8
>>> compute_level(3,0.1,0)
0
>>> compute_level(3,0.1,1)
3.0
>>> compute_level(3,0.1,5)
16.0
>>> compute_rate(3,0.1,5)
3.5
```