CS 152: Function Parameters

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 14 (zybooks) – you should have already done that ☺
- Lab 09
- Reading 15 (zybooks) – you should have already done that ☺
- Lab 10
- Reading 16 (zybooks) – you should have already done that ☺

Recall Activity

- Write the function calls for the following code:

```
def reverseStr(str):
    if(len(str) == 0):
        return ''
    return str[-1] + reverseStr(str[:-1])
```

Write your answer in our today's attendance assignment.
☺

Keyword Arguments

- Functions that have many arguments could be difficult to read
- Keyword arguments allow arguments to map to parameters by name, instead of implicitly by position in the argument list.
- When using keyword arguments, the argument list does not need to follow a specific ordering.
- Keyword arguments can be mixed with positional arguments, provided that the keyword arguments come last.

```
def split_check(amount, num_people, tax_percentage, tip_percentage):
    # ...

split_check(125.00, tip_percentage=0.15, num_people=2, tax_percentage=0.095)
```

Default Parameters Value

- Function can have a default parameter value for one or more parameters
  - meaning that a function call can optionally omit an argument, and the default parameter value will be substituted for the corresponding omitted argument.

```
def print_date(day, month, year, style=0):
    if style == 0:  # American
        print(month, '/', day, '/', year)
    elif style == 1:  # European
        print(day, '/', month, '/', year)
    else:
        print('Invalid Style')

print_date(30, 7, 2012, 0)
print_date(30, 7, 2012, 1)
print_date(30, 7, 2012)  # style argument not provided! Default value of 0 used.
```

Arbitrary Arguments

- A function definition can include a *args parameter that collects optional positional parameters into an arbitrary argument list tuple.

```
def print_sandwich(bread, meat, *args):
    print(f'{meat} on {bread}', end=' ')
    if len(args) > 0:
        print('with', end=' ')
    for extra in args:
        print(extra, end=' ')
    print('')

print_sandwich('sourdough', 'turkey', 'mayo')
print_sandwich('wheat', 'ham', 'mustard', 'tomato', 'lettuce')
```

Arbitrary Arguments

- Adding a final function parameter of **kwargs creates a dictionary containing "extra" arguments not defined in the function definition
  - kwargs is short for keyword arguments.

```
def print_sandwich(meat, bread, **kwargs):
    print(f'{meat} on {bread}')
    for category, extra in kwargs.items():
        print(f'   {category}: {extra}')
    print()

print_sandwich('turkey', 'sourdough', sauce='mayo')
print_sandwich('ham', 'wheat', sauce1='mustard', veggie1='tomato', veggie2='lettuce')
```

Coding Along

- Download the file.csv
- Download the file_list_list.py
- We will go through the code so you can understand what is happening and think about the output before you run the program.