Admin

- No Lab 4 due to Rally Day (optional exercises online)
- Office hours tomorrow 11am-1pm (Ford 015)
- Notes about Chap 6 reading: skip graphics (Chap 4) examples
- Notes about homeworks and labs
Week 4: all about FUNCTIONS

- Homework 1 examples
- Recap last time
- Return statements (vs. print statements)
- Modify parameters vs. return
Extra:
Directory structure suggestions
Homework 1 examples
# CSC 111, Homework 1
# Part A: Computing Taxes
# Julia Kim

def main():
    # Info Gathering from user
    print("This program will compute your taxes for 2016.")
    first_name= (input("Enter your first name: "))
    last_name= (input("Enter you last name: "))
    salary=eval (input("Enter your salary for 2016: "))
    deductions= eval (input("Enter your deductions: "))

    # Computing Tax values
    federal_Taxes=round(((salary-deductions)*0.28)
    state_Taxes=round ((((salary-deductions)*0.05)

    # Values to return to user
    # Blank print used below to provide a line gap between then entries given,
    # and the data returned
    print (""
    print ("Dear "+first_name+last_name+":")
    print ("Your federal taxes for 2016 are ", federal_Taxes)
    print ("Your state taxes for 2016 are ", state_Taxes)
    print ("You will have to pay a total of ", (federal_Taxes+state_Taxes))

main()
# Define function main

def main():
    print('This program will compute your taxes for 2016')
    first = input('Enter your first name: ')  
    last = input('Enter your last name: ') 
    salary = eval(input('Enter your salary for 2016: '))
    deductions = eval(input('Enter your deductions: '))

    # Compute taxes
    fedtax = round((salary - deductions) * 0.28)
    statetax = round((salary - deductions) * 0.05)

    # Experiment with displaying $ 0 when deductions are greater than salary
    if salary > deductions:  
        #this section runs like the code asked for

        # Display info for user
        print('')
        print('Dear', first, last + ':')
        print('Your federal taxes for 2016 are $', fedtax)
        print('Your state taxes for 2016 are $', statetax)
        print('You will have to pay a total of $', fedtax + statetax, 'in taxes.

    if salary == deductions:
        # Display info for user
        print('')
        print('Dear', first, last + ':')
        print('Your federal taxes for 2016 are $', fedtax)
        print('Your state taxes for 2016 are $', statetax)
        print('You will have to pay a total of $', fedtax + statetax, 'in taxes.

    if salary < deductions:
        # Display info for user
        print('')
        print('Dear', first, last + ':')
        print('Your federal taxes for 2016 are $ 0')
        print('Your state taxes for 2016 are $ 0')
        print('You will have to pay a total of $ 0 in taxes.

#Invoke Main
main()
# CSC 111 HW1 B
# Heloise Cheruvalath
# Program to find how many hours it would take to travel distances between 100 and 1000 kms
# (intervals of 50kms) based on input speed in miles/hr

def main():
    # Person inputs speed in miles/hr
    mph = eval(input("Enter your speed in miles/hour:"))

    # function to print out values
    for kms in range(100,1050,50):
        miles = round(kms * 0.62137119)
        hours = miles/mph
        print(kms,"kms is", miles,"miles.","At",mph,"miles/hr, this would take",hours,"hrs.")

main()
# CSC HW Part B: Distance Conversions
# Author: Danica Miguel
# This program will convert distance

def main():
    speed = eval(input("Enter your speed in miles/hr: "))
    for i in range(100, 1001, 50):

        # convert kms to miles (conversion factor is kms * 0.62)
        miles = round(i * 0.62)

        # formula for hours
        hours = (miles / speed)

        # give this information back to the user
        print(i, "kms is", miles, "miles.", "At", speed, "miles/hr,"", "this would take", hours, "hrs.")

main()
Recap
Informal quiz (discuss with a partner)

1) Which are the “helper functions” in this code?

2) Which parameters are formal vs. actual?

3) How many time is the function `happy()` called?

4) True or false: `return` is kind of like `print`
Informal quiz (discuss with a partner)

1) Which are the “helper functions” in this code?
   happy() and sing(person)

2) Which parameters are formal vs. actual?

3) How many times is the function happy() called?

4) True or false: return is kind of like print
Informal quiz (discuss with a partner)

1) Which are the “helper functions” in this code?
   happy() and sing(person)

2) Which parameters are formal vs. actual?
   formal: person
   actual: “Shaneil”, “Yuhan”, “Sarah”, “Sophia”

3) How many time is the function happy() called?

4) True or false: return is kind of like print

```python
# Happy Birthday
# Zelle, Section 6.2

def happy():
    print("Happy Birthday to you!")

def sing(person):
    happy()
    happy()
    print("Happy Birthday, dear", person + ",
         person + ".")
    happy()

def main():
    sing("Shaneil")
    print()
    sing("Yuhan")
    print()
    sing("Sarah")
    print()
    sing("Sophia")

main()
```
Informal quiz (discuss with a partner)

1) Which are the “helper functions” in this code?
   - happy() and sing(person)

2) Which parameters are formal vs. actual?
   - formal: person

3) How many time is the function happy() called?
   - 12 times

4) True or false: return is kind of like print
Informal quiz (discuss with a partner)

1) Which are the “helper functions” in this code?
   happy() and sing(person)

2) Which parameters are formal vs. actual?
   formal: person
   actual: “Shaneil”,”Yuhan”,”Sarah”, “Sophia”

3) How many time is the function happy() called?
   12 times

4) True or false: return is kind of like print
   FALSE!
Return statements
Return vs. print

- **Print** produces output on the screen (in the shell), it goes not actually give back anything.
- **Return** gives back the actual value of a variable, which can be used later on.
- **Print** is like “showing”, **return** is like “giving”.
- Functions should almost always either **return something** or **modify something**, although the birthday example only prints.
Got to this point last time:

```
# CSC 111, Day 11
# Author: Sara Mathieson and CSC 111 class
# Sea-level (HW2) revisited using functions and return

# Compute the rate for a specific year
def compute_rate(orig_rate, accel, year):
    rate = orig_rate + accel*year
    return rate  # return ("give back") rate

# Compute the sea level for a specific year
def compute_level(orig_rate, accel, year):
    # start off with a sea level of 0
    level = 0

    # loop over each year, adding on the current rate
    for i in range(year):
        # use our "helper function" to compute the rate
        curr_rate = compute_rate(orig_rate, accel, i)
        level = level + curr_rate

    # here we printed level, but if we wanted to use it
    # later on, we would need to return it
    print(level)
```

Get back to original output:

```
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
```
Return vs. Modify
(live coding demo)
Right after `addInterest` is called, `balance` and `amount` both point to 1000.
After `addInterest` is finished, `balance` is 1050, but `amount` has never been reassigned.
Functions that modify parameters

Balances and amounts still point to the same structure, but the elements have been changed.

```python
def test():
    amounts = [1000, 2150, 800, 3275]
    rate = 0.05
    addInterest(amounts, rate)
    print(amounts)

def addInterest(balances, rate):
    for i in range(len(balances)):
        balances[i] = balances[i] * (1+rate)
```

Takeaway: lists are mutable, but most primitive types (int, str, float, bool) are not.