Homework 8 is due Tuesday April 11
Homework 9 is due Tuesday April 18 (last homework)
Final Project (roughly 1.5 homeworks) due Tuesday May 2
Thursday office hours **10am-12pm (Ford 015)**
Liberal Arts Module on Friday
Outline: 4/5

- Recap Homework 4
- Continue recursion (Fibonacci)
- Introduce Lab 8 and Homework 8
- Friday: liberal arts module (maps)
Question 1: understand well
- Graphics
- For-loops, if-statements
- Nested structures

Question 2: need to spend more time
- While loops
- Dictionaries
- Different ways of reading files
Homework 4 Examples

(selected by Aditya)
def game(score_comp, score_user):
a = random.choice(['rock', 'paper', 'scissors'])
b = input('enter rock, paper, or scissors: ')

    # see if it's a draw(user's choice = computer's)
    if a == b:
        print(a, 'and', a, 'is a tie')

    else:
        
            # if not a draw, find both user's and computer's choices in one sentence
            # of the list, then see which one starts the sentence.
            # The one that starts the sentence is the winner.

        for i in range(3):
lst = ['rock smashes scissors', 'scissors cuts paper', 'paper covers rock']
string = lst[i].replace(a, '').replace(b, '')

            if string[0] == ' ' and string[-1] == ' ':
                if a[0] == lst[i][0]:
                    print(lst[i] + ', computer wins!')
score_comp = score_comp + 1
                else:
                    print(lst[i] + ', user wins!')
score_user = score_user + 1

        return score_comp, score_user
def main():
    score_comp = 0
    score_user = 0
    n = eval(input("Enter the number of rounds: "))

    # iteration of rounds
    for k in range(n):
        print("\nRound:",k+1)
        score_comp, score_user = game(score_comp, score_user)

        print("current score: user ",score_user," , computer ",score_comp, sep="")

    # Conclude final result
    print("\nThe final score is: user ",score_user," , computer ",score_comp, sep="")
    if score_user > score_comp:
        print("User wins")
    elif score_user < score_comp:
        print("Computer wins")
    else:
        print("Draw")
user.want_to_play = True  # A boolean variable to record if the user want to continue the game.

for i in range (rounds):
    if user.want_to_play == True:
        print ("round: ",i)
        user_choice = input("Enter rock, paper, or scissors: ")

        # The boolean variable changes into False when the user doesn't want to play any more.
        if user_choice == "quit":
            user.want_to_play = False
            print ("Game Over!\n")

    else:
        computer_choice = random.choice(game_choices)

        if computer_choice == user_choice:
            print (user_choice , "and", computer_choice + " is a tie.")
            user_score = user_score
            computer_score = computer_score

        else:
            message(computer_choice, user_choice)  # Call the second helper function
            result = compare(computer_choice, user_choice)  # To change scores after one round
            if result.split()[-2] == "computer":
                computer_score = computer_score + 1
            elif result.split()[-2] == "you":
                user_score = user_score + 1

            print ("current score: user", user_score, ", computer", computer_score, "\n")
switch = True # Boolean variable

for i in range(round_number):
    if switch == True:
        user = input("enter rock, paper, or scissors: ")
        computer = random.choice(choice)

    if user == "quit": # stop the loop if user types 'quit'
        switch = False

    if switch == True:
        winner = game(user, computer)

        if winner == user:
            user_score = user_score + 1 # calculating the score in a cycle

        elif winner == computer:
            computer_score = computer_score + 1

print("current score: user",user_score, "computer",computer_score)
print(" ")
# create a function that performs random pairs function for 10 times

def random_pairs_10(lst):

    total_lst = []  # a list to store all the pairs we had before
    for i in range(1,11):
        print("For week",str(i)+":")
        print()

        # call the random pairs function and assign the value of the partner list into the variable "partner_lst"
        lab_lst = random_pairs(lst)

        for item in lab_lst:

            # if in the list we have the same pair as in the total list, we will do random pair function again
            # and create a new list until there is no pair that is same as those in the total list.
            if item in total_lst:
                lab_lst = random_pairs(lst)

        total_lst.append(lab_lst)  # update the total list to include week i's partner list
Homework 7 Extensions
Flappy Pinwheel by Mai and Butterfly Catcher by Isabelle
Continue Recursion
Reversing a string

reverse("summer")

reverse("ummer") + "s"

(reverse("mer") + "m") + "u") + "s"

(((reverse("er") + "m") + "m") + "u") + "s"

((((reverse("r") + "e") + "m") + "m") + "u") + "s"

((((reverse("" ) + "r") + "e") + "m") + "m") + "u") + "s"
Fibonacci Example
Fibonacci numbers

Each Fibonacci number is the sum of the previous two Fibonacci numbers

Recursion: \( F_n = F_{n-1} + F_{n-2} \)

Base cases: \( F_0 = 1 \) and \( F_1 = 1 \)
Fibonacci Function Stack

fib(5)

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

Line A
Line B
Line C

current line
Function Stack
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

A
fib(4)

A
fib(5)

Function Stack

current line
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

- Line A
- Line B
- Line C

Function Stack

- A: fib(3)
- A: fib(4)
- A: fib(5)
Fibonacci Function Stack

```python
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack:

- A: `fib(5)`
- A: `fib(4)`
- A: `fib(3)`
- A: `fib(2)`

Current line:

- Line A: `fib(5)
- Line B: `fib(4)
- Line C: `fib(3)

Line A: `fib(2)`
Fibonacci Function Stack

```python
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack:
- fib(1)
- fib(2)
- fib(3)
- fib(4)
- fib(5)

return 1
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

defib(0):
  out1=1

fib(2):

fib(3):

fib(4):

fib(5):
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack

- Line A: fib(2)
- Line A: fib(3)
- Line A: fib(4)
- Line A: fib(5)

Current Line: out1=1, out2=1
Fibonacci Function Stack

```python
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack

```
<table>
<thead>
<tr>
<th></th>
<th>fib(2)</th>
<th>return 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>fib(3)</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>fib(4)</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>fib(5)</td>
<td></td>
</tr>
</tbody>
</table>
```
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

```
<table>
<thead>
<tr>
<th>Line</th>
<th>Function Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>fib(5)</td>
</tr>
<tr>
<td>B</td>
<td>fib(4)</td>
</tr>
<tr>
<td>C</td>
<td>fib(3)</td>
</tr>
<tr>
<td></td>
<td>out1=2</td>
</tr>
</tbody>
</table>
```
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack:

- A: `fib(5)`
- B: `fib(4)`
- C: `fib(3)`
- 1
- 2
- 1

Current line: `return 1`
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack:

<table>
<thead>
<tr>
<th>Line A</th>
<th>Line B</th>
<th>Line C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>fib(4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fib(5)</td>
</tr>
</tbody>
</table>

Current line:

- out1 = 2
- out2 = 1
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

fib(5)
```

Function Stack

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fib(3)</td>
<td>fib(4)</td>
<td>fib(5)</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|       |       | current line
```

Line A

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

fib(5)
fib(4)

out1=3

A

B

C

1 1 1
2 1
3

current line

Function Stack
Fibonacci Function Stack

```python
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack

- Line A
- Line B
- Line C

Current line: out1 = 3
Fibonacci Function Stack

```python
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Stack:
- fib(5)
- fib(4)
- fib(2)
- fib(1)

Function Stack:

<table>
<thead>
<tr>
<th>Line</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>fib(5)</td>
</tr>
<tr>
<td>B</td>
<td>fib(4)</td>
</tr>
<tr>
<td>C</td>
<td>fib(2)</td>
</tr>
<tr>
<td></td>
<td>fib(1)</td>
</tr>
</tbody>
</table>

Current line: `return out1 + out2`

Output: `out1 = 3`
Fibonacci Function Stack

fib(5)

fib(4)

fib(2)

3

2 1

1 1

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

A

B

C

out1 = 1

out1 = 3

Function Stack

Line A

Line B

Line C
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

fib(5)  # Stack function calls

fib(4)

3  fib(2)
2  1
1

fib(2)
1
1

fib(0)
1
1

Line A
Line B
Line C

Function Stack

fib(0)  return 1
fib(2)  out1=1
fib(4)  out1=3
fib(5)  

current line

```
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Function Stack:

- **A**: fib(5)
- **B**: fib(4)
- **C**: fib(2)

Current line:

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

- **A**: fib(5) -> out1 = 3
- **B**: fib(4) -> out1 = 1
- **C**: fib(2) -> out1 = 1, out2 = 1
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

fib(5)
  fib(4)
    3
    fib(2)
      2
      1
      1
      1
      1
    1
    1
  return 2

out1 = 3
Fibonacci Function Stack

fib(5)

fib(4)

3

2

2

1

1

1

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

Line A
Line B
Line C

C

fib(4)  out1=3
out2=2

A

fib(5)
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

fib(4)

3

2

1

1

1

2

1

1

1
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

```
<table>
<thead>
<tr>
<th>Line A</th>
<th>Line B</th>
<th>Line C</th>
</tr>
</thead>
<tbody>
<tr>
<td>if n == 0 or n == 1:</td>
<td>out1 = fib(n-1)</td>
<td>return out1 + out2</td>
</tr>
<tr>
<td>return 1</td>
<td>out2 = fib(n-2)</td>
<td></td>
</tr>
</tbody>
</table>
```

fib(5)

5

3

2

2 1 1

1 1

1

Line A

Line B

Line C

out1 = 5

current line

Function Stack
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

- **Line A**: `fib(3)
- **Line B**: `fib(5)`, `out1=5`
- **Line C**: `fib(3)`
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

```
A
| fib(2) |
A
| fib(3) |
B
| fib(5) |
```

out1=5
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fib(5)</td>
<td>fib(3)</td>
<td>fib(1)</td>
<td>out1=5</td>
</tr>
<tr>
<td>fib(3)</td>
<td>fib(2)</td>
<td>return 1</td>
<td></td>
</tr>
<tr>
<td>fib(2)</td>
<td>fib(1)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>fib(1)</td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

out1 = fib(5)

out1 = fib(3)

out1 = fib(2)

out1 = 5

out1 = 1

fib(2)
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

Fibonacci Function Stack

fib(5)

5

3

2

fib(3)

fib(2)

1

1

1

1

1

1

2

1

1

1

out1=5

return 2
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

out1=5

out1=2

out1=5
Fibonacci Function Stack

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

```
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>def fib(n):</td>
<td>fib(5)</td>
</tr>
<tr>
<td>B</td>
<td>if n == 0 or n == 1:</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>return 1</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>else:</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>out1 = fib(n-1)</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>out2 = fib(n-2)</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>return out1 + out2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>return 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>out1=2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>out1=5</td>
<td></td>
</tr>
</tbody>
</table>
```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

out1 = fib(5)
out2 = fib(3)

C
  | fib(3) |
  |---|---|
  |   | out1=2 out2=1 out1=5 |
B
  | fib(5) |
  |---|---|
  |   | out1=5 |

Function Stack
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

fib(5)
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
**Fibonacci Function Stack**

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

```
5
  3
  3
  2
  2
  1
  1
  1

fib(5)

1 1
```

Return 8
Fibonacci Function Stack

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

empty!
Fibonacci Tree with Function Calls

\[ \text{fib}(5) \]

\[ \text{fib}(4) \quad \text{fib}(3) \]

\[ \text{fib}(3) \quad \text{fib}(2) \]

\[ \text{fib}(2) \quad \text{fib}(1) \]

\[ \text{fib}(1) \quad \text{fib}(0) \]

\[ \text{fib}(1) \quad \text{fib}(0) \]

\[ \text{fib}(0) \]

---

def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2

Line A
Line B
Line C
Homework 8 and Lab 8
Homework 8: Maps
Homework 8: Maps
Homework 8: Maps
Lab 8: Fractal Trees using Recursion
Lab 8: Fractal Trees using Recursion