# Quiz Section Week 6
# May 3, 2018

HMMs

For loop clarifications

File I/O

## My favorite HMM example (and apparently Wikipedia’s as well)

## My favorite HMM example (and apparently Wikipedia’s as well)

You have a friend named Stan who lives across the country, and who only likes to do three things:

1. Go for a walk
2. Shop
3. Clean the house

## My favorite HMM example (and apparently Wikipedia’s as well)

You have a friend named Stan who lives across the country, and who only likes to do three things:

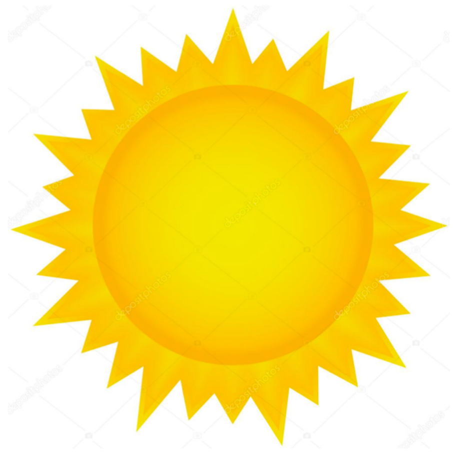
1. Go for a walk
2. Shop
3. Clean the house

He decides what to do each day depending on the weather

## My favorite HMM example (and apparently Wikipedia’s as well)

If it’s sunny:

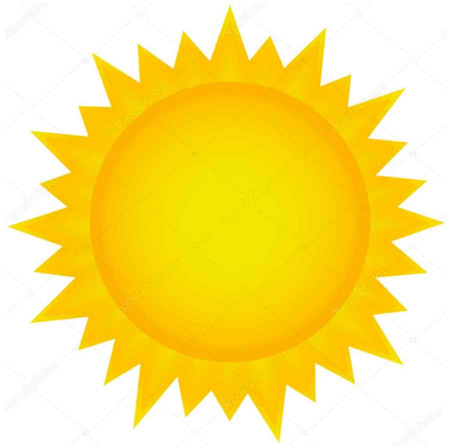
1. 60% of the time he goes for a walk
2. 30% of the time he goes shopping
3. 10% of the time he cleans his house



# My favorite HMM example (and apparently Wikipedia's as well)

If it's sunny:

1. 60% of the time he goes for a walk
2. 30% of the time he goes shopping
3. 10% of the time he cleans his house



If it's rainy:

1. 10% of the time he goes for a walk
2. 40% of the time he goes shopping
3. 50% of the time he cleans his house

## My favorite HMM example (and apparently Wikipedia’s as well)

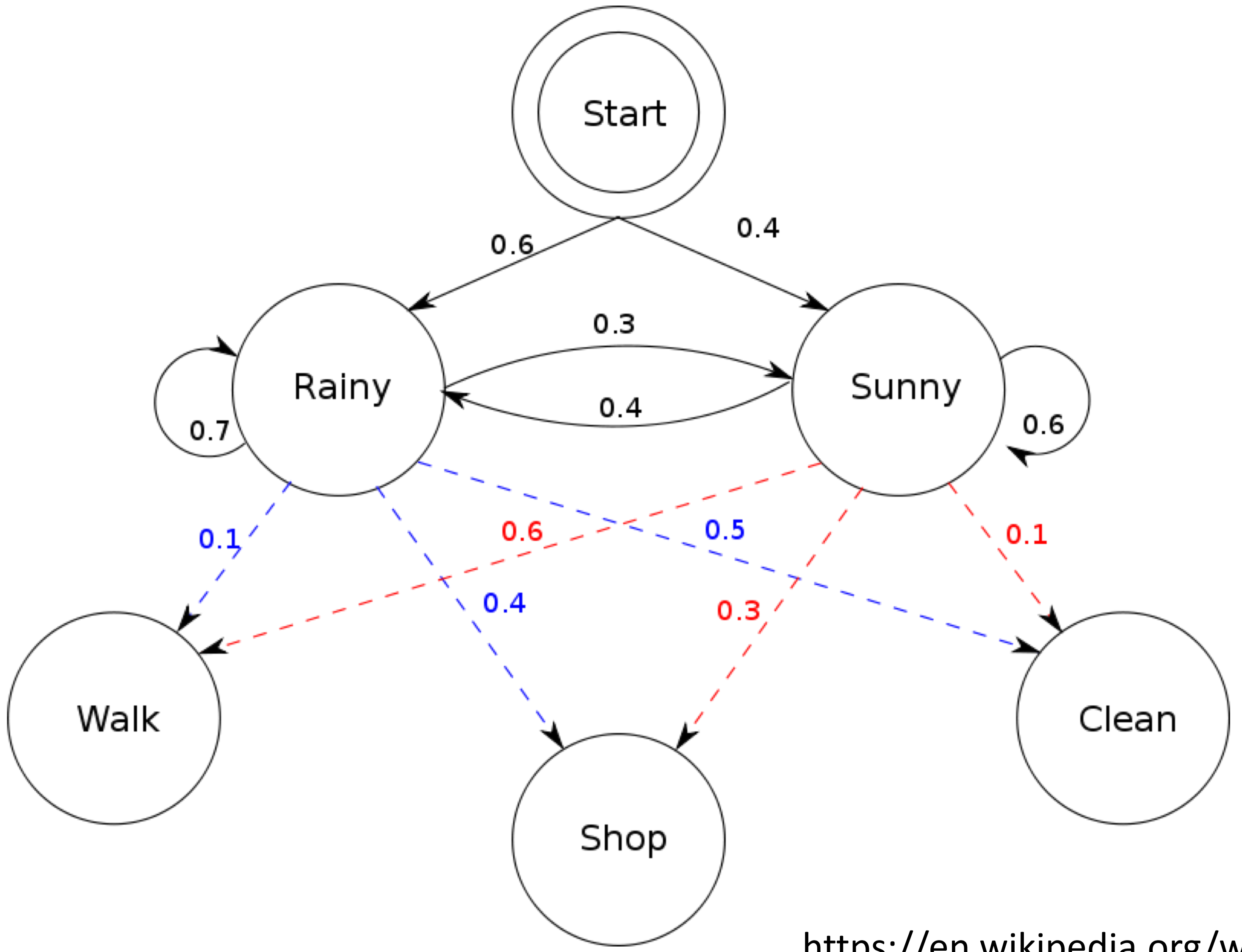
The challenge:

Stan calls you every day and tells you what he did that day, but does not tell you the weather. Can you predict what the weather is based on what he tells you?

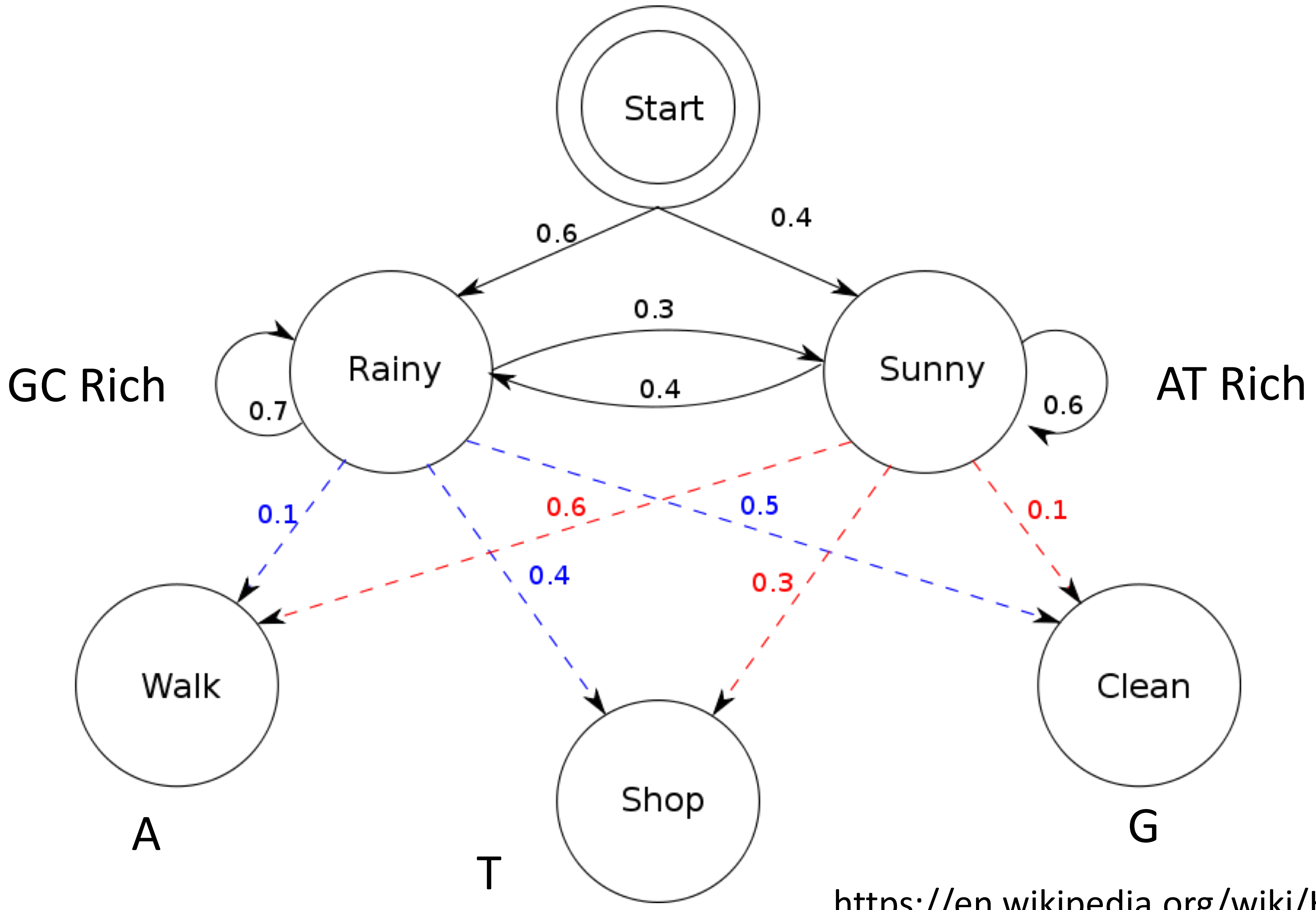
What are the emissions?

What are the hidden states?

Any other info you need that I haven’t given yet?



https://en.wikipedia.org/wiki/Hidden_Markov_model



https://en.wikipedia.org/wiki/Hidden_Markov_model

## Chain rule of probability

- Probability that A and B occur is the probability that A occurs * the probability that B occurs

## Programming

# For loops follow up

When should you use the format `for i in list1:` and when should you use the format `for i in range(len(list1)):` ?

```
list1 = [0,1,2,3]
list2 = [3,4,5,6]
```

Summing all of the items in list1?

Adding the items from list1 and list2 pairwise?

Adding some of the items from list1 to a new list?

## What’s wrong here?

```
for item in h_list:
    print h_list[item]
```

## Opening files

- The open() command returns a file object.

```
<filehandle> = open(<filename>, <access type>)
```

- Python can read, write or append to a file:
  - 'r' = read
  - 'w' = write
  - 'a' = append
- Create a file called “hello.txt” containing one line: “Hello, world!”

```
>>> my_file = open("hello.txt", "r")
```

# Reading the whole file

- You can read the contents of the file into a single string.

```
>>> my_string = my_file.read()
>>> print my_string
Hello, world!


>>>
```

Why is there a blank line here?

## Reading the whole file

- Now add a second line to your file (“How ya doin’?”) and try again.

```
>>> my_file = open("hello.txt", "r")
>>> my_string = my_file.read()
>>> print my_string
Hello, world!
How ya doin'?

>>>
```

## Reading the whole file

- Alternatively, you can read the file into a list of strings.

```
>>> my_file = open("hello.txt", "r")
>>> my_string_list = my_file.readlines()
>>> print my_string_list
['Hello, world!\n', "How ya doin'?\n"]
>>> print my_string_list[1]
How ya doin'?
```

# Reading one line at a time

- The `readlines()` command puts all the lines into a list of strings.
- The `readline()` command returns the next line.

```
>>> my_file = open("hello.txt", "r")
>>> my_string = my_file.readline()
>>> print my_string
Hello, world!

>>> my_string = my_file.readline()
>>> print my_string
How ya doin'?

>>>
```

# Writing to a file

- Open the file for writing or appending.

```
>>> my_file = open("new.txt", "w")
```

- Use the `<file>.write()` method.

```
>>> my_file.write("This is a new file\n")
>>> my_file.close()
>>> ^D
> cat new.txt
This is a new file
```

Always close a file after you are finished reading from or writing to it.

# Write

- `<file>.write()` does not automatically append an end-of-line character.
- `<file>.write()` requires a string as input

```
>>> new_file.write("foo")
>>> new_file.write(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: argument 1 must be string or read-only character buffer,
 not int
```

- `<file> = open(<filename>, r|w|a)`
- `<string> = <file>.read()`
- `<string> = <file>.readline()`
- `<string list> = <file>.readlines()`
- `<file>.write(<string>)`
- `<file>.close()`

## Sample problem #1

- Write a program `read-first-line.py` that takes a file name from the command line, opens the file, reads the first line, and prints the result to the screen.

```
> python read-first-line.py hello.txt
Hello, world!

>
```

## You can use a for loop to iterate through lines of a file

```
fin = open('qs5.txt', 'r')
all_lines = []
for line in fin: # In a for loop, fin acts
like a list of strings
    all_lines.append(line)
fin.close() # Lets the computer know it can
free up resources used to read the file
print all_lines
```

## Often, you don't want the newline at the end of the line

```
my_open_file = open(sys.argv[1])
s1 = my_open_file.readline().strip()
s2 = my_open_file.readline().strip()
```

Note: if in a different directory, have to supply **file path, e.g.**:
python myScript.py /Users/cecilia/genome373/dataFile.txt

## Example program structure with input/output

**python analyze_sequence_pairs.py inputfile.txt outputfile.txt**

Setup

```
#import needed modules and functions
#
```

Input

```
#Read in data from file
```

```
#Do a calculation
```

Output

```
#Write output to file
```

# Example program structure with input/output

python analyze_sequence_pairs.py inputfile.txt outputfile.txt

Setup

```
import sys
from qs6 import * #import the definition of calculate_jukes_cantor
```

Input

```
fin = open(sys.argv[1],'r')
seqs = []
for line in fin:
        seqs.append(line.rstrip()) # gets rid of \n at the end of the
line
print seqs
fin.close()
```

Output

```
answer = calculate_jukes_cantor(seqs[0], seqs[1])
fout = open(sys.argv[2],'w')
fout.write( seqs[0] + ' ' + seqs[1] + ' ')
fout.write( str(answer) + '\n')
fout.close()
```

## Exercise: write a program to calculate and write to a file the # of times a start codon occurs in each sequence

Use this function:

```
def count_start_codons(seq):
    num_starts = seq.count("ATG")
    return num_starts
```

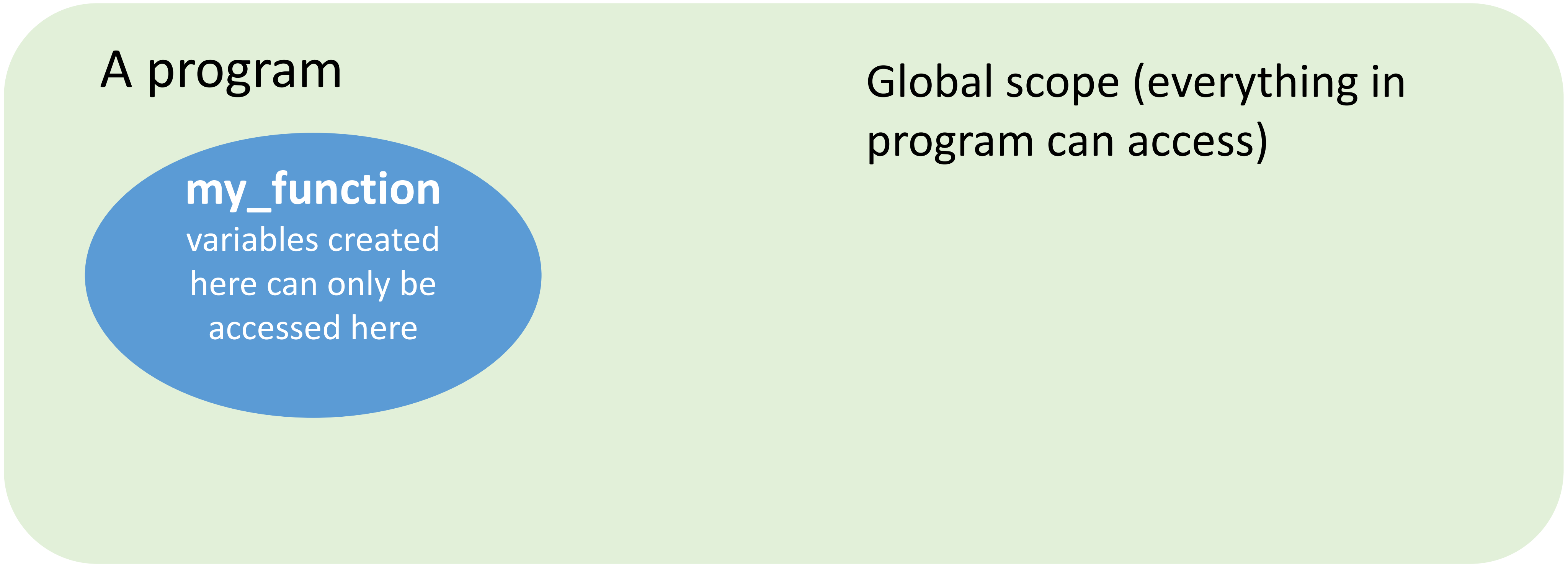
```
python count_starts.py sequences.txt output_file.txt
```

output file:

```
ATGGGGATG     2
CAGTTATGCCT   1
```

# Scope of a variable

- Variables created in the main part of your program can be accessed anywhere (**global** scope)
- Variables created within functions are only accessible within that function (**local** scope)


A program
Global scope (everything in program can access)
my_function
variables created here can only be accessed here

## Scope of a variable

```
new_list = [0,1,2]

def less_than(myList, num = 4):
     new_list = []
     for x in myList:
          if x < num:
               new_list.append(x)
     return new_list

print new_list
anotherList = [3,7,12]
print less_than(anotherList)
```

# Scope of a variable

```
new_list = [0,1,2]

def less_than(myList, num = 4):
     #new_list = []
     for x in myList:
          if x < num:
               new_list.append(x)
     return new_list

print new_list
anotherList = [3,7,12]
print less_than(anotherList)
```

Don't do this!! You'll confuse yourself

Define all your functions at the beginning of your program or in another file