Key course information

Quiz #2 will be available on Monday!

- Available on MyCourses (multiple choice questions)
- Quiz #2 closes at 11:59:59 pm on Monday, September 25th
- Questions cover topics from the last two weeks
- The total number of quizzes will change from 5 to 10
  - Still contribute to 5% of your final grade
  - Less weight per quiz
Position weight matrices (PWM) is a common way of representing motifs (or patterns) in sequences.

A sequence motif is a nucleotide or amino-acid sequence pattern that is widespread and has, or is conjectured to have, a biological significance.

In HW1, we’re attempting to understand where a protein of interest (’protein_X’) binds:

- To start, you’re given a set of sequences where protein_X is believed to bind.
- Then you’ll determine it’s sequence motif of binding.
- Finally, we’ll learn how to scan a DNA sequence for potential binding sites of protein_X.
Python programming visualizer

Link to Python Tutor’s programming visualizer: http://www.pythontutor.com/visualize.html#mode=edit

Visually represent Python code execution
  ▶ Basic Python programming examples provided
    ▶ Other languages available as well
  ▶ Great resource for understanding a program’s control flow

Try out the live programming mode
  ▶ Continually runs and visualizes your code as you type
  ▶ Experimental (may break from time to time)
Nested loops

You can use one or more loops inside any another

nested for loops

```python
for item_A in sequence_A:
    for item_B in sequence_B:
        statements(s)
        statements(s)
```

nested while loops

```python
while condition_A:
    while condition_B:
        statement(s)
        statement(s)
```
Nested loops #2

You can mix different types of loops nested for and while loops

```
# for loop encapsulating a while loop
for item in sequence:
    while condition:
        statements(s)
        statements(s)

# while loop encapsulating a for loop
while condition:
    for item in sequence:
        statement(s)
        statement(s)
```
Mixing loops example

Chris’ talented biochemist friend, David, loves Subway™

Let’s represent his order to the Subway clerk in Python

- Using nested **for**/**while** loops
- Each time David chooses a vegetable his excitement increases
- If his excitement becomes too high, he’ll pass out

```python
index = 0
vegetables = []
print("""David: Subway is the best!
Clerk: Ummm, ok.
What would you like to have on your sub?"")
Davids_excitement = 1
# insert for/while loops
for ...
```

1
```python
index = 0
vegetables = []
print("""David: Subway is the best!
Clerk: Ummm, ok.
What would you like to have on your sub?"")
Davids_excitement = 1
# insert for/while loops
for ...
```
Loops and conditional statements

Conditional statements can be used within loops to control flow

```python
animals = ["dog","cAt"]
for animal in animals:
    capital = False
    for letter in animal:
        if letter.isupper():
            capital = True
    if capital:
        print(animal,"has capital letter(s)")
    else:
        print(animal,"has no capital letters")
```

# Ouput:
# dog has no capital letters
# cAt has capital letter(s)
```
Loop control statements

A situation may arise where you will need to exit a loop completely
- When an external condition is triggered or
- To skip a part of the **loop body** and start the next iteration

Python provides **`break`** and **`continue`** statements to handle such situations
- Allows for more control of loop flow
- Can be used in both **`for`** and **`while`** loops
The break statement

**break** in Python terminates the current loop

- Execution resumes at the next statement after the loop

Most common use is to trigger a hasty exit from a loop under some condition

```python
for letter in "Python":
    if letter == "t":
        break
    print(letter)
print(letter)
```

# Output:

# P
# P
# y
# t
continue statement

**continue** returns the control to the beginning of the loop

- Skips all the remaining statements in the current iteration

```python
for letter in "Python":
    if letter in ["y","o"] :
        continue
    print(letter)
```

# Output:
# P
# t
# h
# n
The pass statement

pass is used when a statement is required syntactically but there is no command or code to execute

- Is a null operation; nothing happens when it execute
- Useful where code will eventually go, but nothing written yet

```python
for letter in "Python":
    if letter == "P":
        pass  # insert code here
    print(letter)
```

# Output:
# P
# y
# t
# h
# o
# n
List comprehensions

Python supports a concept called **list comprehensions**

- Can be used to construct lists in a very natural way
- Like in Mathematics:

\[
S = \{ x^2 : x \in \{0 \ldots 9\} \}
\]
\[
V = (1, 2, 4, 8, \ldots, 2^{12})
\]

We can represent list \( S \) and tuple \( V \) in Python as:

```python
1 S = [x**2 for x in range(10)]
2 V = tuple([2**i for i in range(13)])
```

** is the power operator in Python
Let's convert the following `for` loop to a list comprehension:

```python
student_names = ["Linda","Denny","Daniella",
                 "Kris","Yujing","Grace"]

# for loop
phrases = []
for name in student_names:
    phrases.append(name + " is a student in COMP 364")
print("\n".join(phrases))

# list comprehension
print("\n".join([name + " is a student in COMP 364" for name in student_names]))
```
List comprehensions with conditional statements

Conditional statements can also be used in list comprehensions

```python
# for loop
phrases = []
for name in student_names:
    if not name == "Grace":
        phrases.append(name + " is a student in COMP 364")
print("\n".join(phrases))

# list comprehension
print("\n".join([name+" is a student in COMP 364" for name in student_names if not name== "Grace"]))
```
Rick and Morty example

Sorry, no Jerrys...but plenty of Ricks and Mortys
Rick and Morty example #2

**Rick** is a man of unquestionable intelligence

▶ **Morty** is his less bright grandson

We observe a subset of the many **Ricks** and **Mortys** across all the infinite possible dimensions

▶ In some dimensions, **Rick** is a clown, drunk, or pickle
  ▶ [https://www.youtube.com/watch?v=4P50K7c9xiM](https://www.youtube.com/watch?v=4P50K7c9xiM)
▶ While **Morty** can be lost, super-powered, or British

```python
obs = [('drunk_rick', 'British_morty'),
       ('pickle_rick', 'British_morty'),
       ('pickle_rick', 'lost_morty'),
       ('drunk_rick', 'lost_morty'),
       ('pickle_rick', 'lost_morty'),...]
```
Download the code template from MyCourses

- Content → Code → rick_morty

Address the following questions using Python:

1. How many lost Mortys are seen with drunk Rick?
2. What is the count of each Morty found with a drunk Rick?
3. Create a tuple describing the count of each Morty for drunk Rick
4. What’s the probability of finding a drunk Rick in our observations?