Genome Informatics Quiz section week 4

April 19, 2018
Housekeeping

• Extra office hours at 4pm today, meet outside this room – if you’re late, email me

• Midterm is next Friday, next week’s quiz section will be review
Any questions on clustering?
A note on programming style

• You should aim for your programs to be both efficient (fewer lines, fewer operations)

• and easy to understand/follow (sometimes means more lines/functions)

• There are MANY ways to solve any programming problem, as you progress, try to be thoughtful about what are the better ways
Homework 2 programming assignment

How did you exclude 'bad' characters?
Homework 2 programming assignment

How did you exclude 'bad' characters?

```python
import sys

codon = sys.argv[1]
ok_nucs = 'augcAUGC'
if codon[0] not in ok_nucs or codon[1] not in ok_nucs or codon[2] not in ok_nucs or len(codon) != 3:
    print 'Error! Invalid input!'
elif codon.upper() == 'AUG':
    print 'Start'
elif codon.upper() == 'UAA' or codon.upper() == 'UAG' or codon.upper() == 'UGA':
    print 'Stop'
else:
    print 'Amino acid'
```
Homework 2 programming assignment

How did you exclude 'bad' characters?

```python
import sys
codon = sys.argv[1]
ok_nucs = 'augcAUGC'
if codon[0] not in ok_nucs or codon[1] not in ok_nucs or codon[2] not in ok_nucs or len(codon) != 3:
    print 'Error! Invalid input!'
elif codon.upper() == 'AUG':
    print 'Start'
elif codon.upper() == 'UAA' or codon.upper() == 'UAG' or codon.upper() == 'UGA':
    print 'Stop'
else:
    print 'Amino acid'
```

Now that we know for loops, is there a better way to write this?
A few more notes on for loops

L1 = [1,2,3,4,5]
L2 = [5,4,3,2,1]

# How do we add each pair of elements together? i.e.
# L1[0]+L2[0] etc.
A few more notes on for loops

L1 = [0, 1, 2, 3, 4]
L2 = [4, 3, 2, 1, 0]

newL = []
for i in range(len(L1)):
    newL.append(L1[i] + L2[i])
print(newL)

[5, 5, 5, 5, 5]
while loop

while (<test>):
    statement1
    statement2

• In English: While some logical test is true, continue executing the block of statements.
What does this program do?

```python
sum = 0
count = 1
while (count < 3):
    sum = sum + count
    count = count + 1
print count
print sum
```
What does this program do?

sum = 0
count = 1
while (count < 3):
    sum = sum + count
    count = count + 1
print count
print sum

sum = 0
count = 1
count < 3? TRUE
sum = 0 + 1 = 1
count = 1 + 1 = 2
count < 3? TRUE
sum = 1 + 2 = 3
count = 2 + 1 = 3
count < 3? FALSE
For versus while loops

• You will probably use for loops more.
• For is natural to loop through a list, characters in a string, etc. (anything of determinate size).
• While is natural to loop an indeterminate number of times until some condition is met.
Examples of for loops

for base in sequence:
    <do something with each base>

for sequence in database:
    <do something with each sequence>

for index in range(5, 200):
    <do something with each index>
Examples of \texttt{while} loops

\begin{verbatim}
while (error > 0.05):
    <do something that will reduce error>
\end{verbatim}

\begin{verbatim}
while (score > 0):
    <traceback through a DP matrix, each
time setting the current score>
\end{verbatim}

Warning: if you write a while loop and the conditional test stays True, the loop will run forever (infinite loop).
What's a function?

**Function:** reusable pieces of code, that take zero or more arguments, perform some actions, and return one or more values

E.g. the function **len**

- arguments: a string or list
- actions: count the number of characters or elements
- return: the integer length of the string or list

How about the function **range**?

- arguments:
- actions:
- return:

```python
>>> len("AGCAGTTTT")
9
```

```python
>>> range(1,4)
[1,2,3]
```
Writing your own functions

```python
def do_something(datapoint):
    # Whatever your calculation is
    result = datapoint * 100
    return result
```

- **argument(s)**: `datapoint`
- **output returned**: `result`
Why write our own functions?

• Avoid repetition, use the same piece of code in different ways

• Better organized, easier-to-understand code
  • harder to make mistakes, easier to find them
  • “Self documenting code”
”Self documenting code”

def upgma_iteration(data):
    dist_matrix = calculate_distance_matrix(data)
    smallest_pair = find_smallest_dist(dist_matrix)
    merged_node_data = merge_nodes(smallest_pair, data)
    return merged_node_data
A few notes on functions

• When you define a function, nothing happens - it doesn’t run until you call it:

```python
def list_of_hannahs(count):
    hannah_list = []
    for i in range(count):
        hannah_list.append('hannah')
    return hannah_list

# nothing happens ....
list_of_hannahs(10)

['hannah', 'hannah', 'hannah', 'hannah', 'hannah', 'hannah', 'hannah', 'hannah', 'hannah', 'hannah']
```
Homework 2 programming assignment

• Try converting this program into a function that returns the answer instead of printing it

```python
import sys

codon = sys.argv[1]
ok_nucs = 'augcAUGC'
if codon[0] not in ok_nucs or codon[1] not in ok_nucs or codon[2] not in ok_nucs or len(codon) != 3:
    print 'Error! Invalid input!'
elif codon.upper() == 'AUG':
    print 'Start'
elif codon.upper() == 'UAA' or codon.upper() == 'UAG' or codon.upper() == 'UGA':
    print 'Stop'
else:
    print 'Amino acid'
```

Now that we know for loops, is there a better way to write this?