Introductory Python Programming Class 3

MCP 743

the elephant in the room

GitHub!

GitHub purpose

Purpose: magically keep track to changes to files

Purpose: keep a history of changes to files

Purpose: version control

Purpose: teams of programmers can work on problems together, without the need to worry about breaking the current work version of a program

Purpose: individuals can work on their own project with the need to worry about breaking the current working version of their program

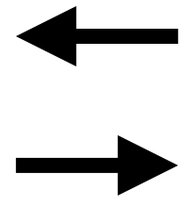
GitHub purpose

We are using GitHub in an unusual way,
so that you learn how to use it!

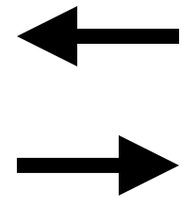
If you do not figure out how to use GitHub to your benefit, you put yourself at a dramatic disadvantage for learning in this class!

The 3 main components of the GitHub pipeline

repository:
a directory
on your
computer



GitHub App:
is always
monitoring
files for
changes



GitHub:
a directory
on the internet
for the members
of your team
to see

Why is GitHub important for this class?

If you need help, your classmates answers are posted.

If you need help, my answers may be posted.

I can see when you start an assignment.

I can see if you are making progress on an assignment.

I can see when you finish an assignment.

I can see what the difficulties are.

Set your pride aside, and post to GitHub incrementally and often

5 years from now…

The point is to learn…not be tested.

Share your progress with others.

It is likely your progress is unique (more on this).

Because most progress is unique, I know where concepts originate(more on this).

I’m a bit concerned about time management, but I could wrong…

lack of slack

It’s OK to notify people that you have made progress

If you can, avoid direct messaging me.
Set the pride aside.

If you have a particular question,
likely someone else does too, and they would
benefit from seeing you question

Anyone can answer a question on slack

Symptom of beginning the assignment too late?

Dan

```
# 1)
i = 0
while i < len(dna) - 2:
    iCodon = dna[i:i+3]
    if iCodon == startCodon:
        print("Found start codon", iCodon, "at index", i)
    i += 1
```

Katelyn!

```
# 1
i = 0
while True:
    i = dna.find(startCodon, i + 1)
    if i == -1: break
    print("Found start codon at index:", i)
```

Dan

```
#3)
minStopCodon, minStopCodonIndex = "", 10000000 # variables for storing the answer
for stopCodon in stopCodons:
     i = 0
     while i < len(dna) - 2:
          iCodon = dna[i:i+3]
          # If the current codon is equal to (double == in Python), then tell me so.
          ##########################################################
          if iCodon == stopCodon:
               stopCodonIndex = i
               # Uses the modulo operation (%) to identify substring slices exactly divisible by 3.
               ##########################################################
               if ((stopCodonIndex + 3) - startCodonIndex) % 3 == 0:
                    # The condition was true, so a stop codon has been found.
                    # Record the find by resetting minStopCodon and minStopCodonIndex
                    ##########################################################
                    minStopCodon = iCodon
                    minStopCodonIndex = stopCodonIndex
                    print("In frame codon is:", iCodon, i)
                    break # Once you find the first instance of each codon type, you can quit the loop
          # Increment the index counter.
          ##############################
          i += 1
print("The first in frame stop codon is:", minStopCodon, "at index", minStopCodonIndex)
```

Anna!

```
# 4)

firstStartCodon = dna.find(startCodon)
lengthofDNA = len(dna) # identify and define length of dna
readingFrame = list(range(firstStartCodon,lengthofDNA,3))

stopCodonIndices_in_readingFrame = [val for val in stopCodonIndices if val in readingFrame]

firstStopCodon_in_readingFrame = (min(stopCodonIndices_in_readingFrame))

print("The first stop codon in the reading frame is",
dna[firstStopCodon_in_readingFrame:firstStopCodon_in_readingFrame+3], "and it is located at
position", firstStopCodon_in_readingFrame, ".")
```

Python list comprehension
http://www.secnetix.de/olli/Python/list_comprehensions.hawk

Chuan!

```
# 4)encountered in the DNA sequence.
for stopCodon in stopCodons:
    j = firststartcodon
    while j < len(dna) - 2:
jCodon = dna[j:j+3]
if jCodon == stopCodon:
    print("Found stop codon (method 3)", jCodon, “at index", j)
    stopCodonCount += 1
j += 3
```

Nadia!

```
# C = 2 * r * pi
# C = circumference of circle
# r = radius of circle
pi = 3.14
r = 2.5
C = 2 * r * pi
print (C)
```

```
# C = 2 * r * pi
# C = circumference of circle
# r = radius of circle
from math import pi
r = 2.5
C = 2 * r * pi
print (C)
```

an example getting pi from the math module that is part of the standard python library

https://docs.python.org/3/library/

WARNING!

Python 3.0 is hiding the truth from you.

Silent index errors as an example

Breaking infinite looping

A quick word on variable convention

By “convention”:

All capital names are clunky and typically reserved for constant value variables

Make variable names descriptive so that you can think through your algorithm more easily

Simple iterators are most often named i, j, k, etc.

My loop variable naming convention…

An observation on print statements and string conversion…

Within print statements, non-string types do not be converted to strings to be printed: they will just magically print

However, if you want to create a string that includes a non-string (usually a number), you have to convert the number using str()

SOC programming topics

lists/tuples

files

if/else control flow statements

dictionaries

Python standard library modules