Project 1: Predicting Huntington’s Disease
Upload to Canvas:
hunt.py and a copy/paste of your test suite and end-to-end testing

Huntington’s disease is a genetic disorder that leads to degeneration of brain cells. Initial symptoms are uncontrolled movements. Irritable moods are also experienced. In the advanced stages of the disease there is cognitive decline and dementia.

Huntington’s disease is caused by a dominant mutation of either of an individual's two copies of the gene that codes for the protein huntingtin (HTT). In the mutant form of HTT that causes Huntington’s disease there are too many repeats of the codon CAG, resulting in a buildup of amyloids in the brain, causing mental deterioration.

Project Scenario

You work in the medical field analyzing the DNA of patients who want to know whether they will come down with Huntington’s disease. Couples want to know what their combined genetic outlook is on passing on Huntington’s disease to children. Patients get their DNA sequenced in the HTT region of the genome and give you the nucleotide sequence. You analyze the DNA, counting repeats of CAG, and let them know their classification and disease status according to this chart:

Repeat count	Classification	Disease status
<27	Normal	Unaffected
27–35	Intermediate	Unaffected
36–39	Reduced Penetrance	Somewhat Affected
>39	Full Penetrance	Affected

http://predictivetestingforhd.com/what-is-hd/genetics/cag-repeats/

If the news is bad then you advise the patients to make an appointment with a counselor who is equipped to help them deal with this difficult news.

You realize that you could write a simple program that you could give to the counselor. Then patients could make one office visit for both DNA analysis and counseling. Patients would not have to wait and this would be a cost saving measure.

The Assignment:

Write a python program that will take user input of the patient’s first name, last name, and DNA sequence (for CAG repeat at the end of their HTT gene). Name your script hunt.py. Your program will calculate the number of CAG repeats in the DNA, as well as the classification and disease status of the patient (from the above chart). **Output should include first and last names of the patient, the DNA sequence, the number of CAG repeats, the classification and disease status.** Your code should include at least 3 functions: a function to get the name and DNA input from the user, a function to count CAG repeats in the DNA and a function to calculate the classification and disease status depending on the number of CAG repeats.

The challenging task will be going through the DNA input to count repeats of "CAG". (**You may NOT use built-in functions like string count, code this yourself! *** see below.**) For simplicity **assume that the DNA input begins with the CAG repeats, and we will stop counting when we encounter either the end of the DNA string or the end of the succession of CAGs.** You may assume that the DNA sequence is accurate and you do not have to test if it has characters that are not A, C, T or G.

You will need to break the DNA string into the slices so you can check for CAG repeats. Start with a simple example, like dna="CAGCTG". How many repeats should you count? Put this and a few other test cases (ex: CAGCAGCA) into a docstring and then code so as to pass the tests.

Grading:

1. Correctness: 30 pts of your grade: Your program should compute the correct CAG repeat count for all possible DNA inputs (including input of empty string) and also produce the correct classification and disease status for each count. **Your program is required to prompt the user for and read keyboard input of first and last name, as well as DNA sequence, and return all these 3 values in a tuple**. **Of course all projects must be able to run** (ie – not have a syntax error), and there should be no crash when legal values are provided. Also, **although your software testing grade will be based on your own test cases, you must run my unit tests from the provided test suite**.

2. Function use: 10 pts: Use the coding style of writing a main function that is called by __main__. You must **call all 3 functions specified here and pass arguments from the main function.** Your main function should be at the bottom of your script, followed by its call from fully outdented code. Values obtained in functions must be **returned to main, stored in variables local to main, and output from main**.

Be sure to define the three required functions so they fulfill the specifications in section ***The Assignment*** above. You are **required to use the function names and headers provided below.**

3. Documentation: 10 pts: Provide **header documentation including the author's name, a *brief* description of what the program accomplishes**. Use **meaningful names for your variables**. In general, do not place comments inside a function body unless there is a complicated calculation that needs to be explained or referenced. However, be sure to **write docstrings that document each function briefly with 1) an accurate explanation of what it accomplishes or returns, 2) how its parameters are used, and 3) preconditions**, if there are any.

4. Testing: 20 pts: Testing serves as proof that **the program you turned in works correctly or that you have detected all errors.** Test thoroughly with **both unit tests and end-to-end testing.** Some unit tests are provided for functions countCAG and prediction (see provided function test_suite), but **your testing grade will be based on your own test cases**. You may test function get_input as part of your end-to-end testing.

For each function test all its paths of execution. Test all the different types of DNA strings that may be encountered. Your function must give correct results for every legal parameter, including **boundary values** (ex: 27 CAG repeats result in a classification that is different from 26 repeats). It is hoped that there are no errors in the hunt.py you upload to Canvas. However, if there are any, testing is expected to reveal them. You get full testing credit for testing an input that doesn't work correctly, as long as you note what error your testing reveals.

Copy/paste into a text file the results of running your test suite. Then comment out the function call that runs the test suite, then run end-to-end tests: Enter information as the user to **demonstrate that your program works as a whole, passing results from one function, to main, then from main to another function, etc**. Copy/paste **these end-to-end runs** also into your file for test cases. Upload the testcases file to Canvas separately from your python script.

5. Quality of solution: 10 pts: Do not make your solution unnecessarily complex. Also, do not condense your solution to the point where it is incomprehensible. Program with proper style: **function definitions at the top of the program and main program at the bottom**; do not intersperse __main__ throughout your program. **Do not use global variables inside functions**.

Collaboration: Students are to do their own work on all projects. You may talk with others about general approaches or to understand the problem statement, but each student is to develop his or her own solution. In addition, using solutions from any other source is forbidden. All projects are to be individual and original efforts.

To run your test suite, paste this code at the top of your file, then add your own unit tests:

```
import sys

def test(did_pass):
   """  Print the result of a test.  """
   linenum = sys._getframe(1).f_lineno   # Get the caller's line number.
   if did_pass:
      msg = "Test at line {0} ok.".format(linenum)
   else:
      msg = ("Test at line {0} FAILED.".format(linenum))
   print(msg)
```

```
def test_suite():
    """ Run the suite of tests for code in this module (this file).
    """
    test(countCAG("C")== 0)
    test(countCAG("CAGCA") == 1)
    test(countCAG("CAGCATCAGCAGCAG") == 1) #see specs

test(countCAG("CAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAG
CAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAGCA
GCAGCAGCAGCAGCAGCA") == 41)
    test(prediction(27) == ('Normal', 'Unaffected') )
    test(prediction(34) == ('Intermediate', 'Unaffected'))
    test(prediction(38) == ('Reduced Penetrance', 'Somewhat Affected'))
    test(prediction(45) == ('Full Penetrance', 'Affected'))
```

Function Names:
You are required to use the following function names and headers: You may download this file and copy-paste the function headers and doctests to your hunt.py.

```
def countCAG(dna):
    """
    Place your function description here.
    """

def prediction(numCAG):
    """
    Place your function description here.
    """

def get_input():
    """
    Place your function description here.
    """
```

NB: Your get_input function should return a tuple containing whatever the user input for first name, last name and DNA, such as:
('Patricia', 'Francis-Lyon', 'CAGCAG')

******* The code you **write** **may use the bracket operator ([]), the concatenation operator (+), the len function, and slices**. You may use **no other built-in functions or operations.** NB: this means No min function, NO **in** operator, etc. The exception is that you may use the **in** operator within the for loop header (but not the body).