**CSS 225 Module 4 Lab Activity – Debugging**

**Deliverables:**

Submit the corrected versions of each of the following programs:

* **time.py**

currentTimeStr = input(**"What is the current time (in hours 0-23)?"**)  
waitTimeStr = input(**"How many hours do you want to wait?"**)  
  
currentTimeInt = int(currentTimeStr)  
waitTimeInt = int(waitTimeStr)  
  
finalTimeInt = currentTimeInt + waitTimeInt  
print(finalTimeInt)

* **pirate.py**

greeting = input(**"Hello, possible pirate! What's the password?"**)  
if greeting in (**" Arrr!"**):  
 print(**"Go away, pirate."**)  
else:  
 print(**"Greetings, hater of pirates!"**)

* **collections.py**

*# Create a collection of these authors and  
# the year they kicked the bucket;  
# print the collection in the following format:  
  
# Charles Dickens died in 1870.  
  
# Charles Dickens, 1870  
# William Thackeray, 1863  
# Anthony Trollope, 1882  
# Gerard Manley Hopkins, 1889*authors = {  
 **"Charles Dickens"**: **"1870"**,  
 **"William Thackeray"**: **"1863"**,  
 **"Anthony Trollope"**: **"1882"**,  
 **"Gerard Manley Hopkins"**: **"1889"**}  
  
for authors, date in authors.items():  
 print(**"%s"** % authors, **" died in "**, **"%s."** % date)

* **branching.py**

*# A time traveler has suddenly appeared in your classroom!  
  
# Create a variable representing the traveler's  
# year of origin (e.g., year = 2000)  
# and greet our strange visitor with a different message  
# if he is from the distant past (before 1900),  
# the present era (1900-2020) or from the far future (beyond 2020).  
# we have to define year as an input, check quotation marks, and spelling  
# don’t use & use and*year = int(input(**"Greetings! What is your year of origin? "**))  
  
if year <= 1900:  
 print (**"Woah, that's the past!"**)  
elif year > 1900 and year < 2020:  
 print (**"That's totally the present!"**)  
else:  
 print (**"Far out, that's the future!!"**)

* **grading.py**

*# Calculating Grades (ok, let me think about this one)*

*# Write a program that will average 3 numeric exam grades, return an average test score, a corresponding letter grade, and a message stating whether the student is passing.*

*# Average Grade*

*# 90+ A*

*# 80-89 B*

*# 70-79 C*

*# 60-69 D*

*# 0-59 F*

*# Exams: 89, 90, 90*

*# Average: 90*

*# Grade: A*

*# Student is passing.*

*# Exams: 50, 51, 0*

*# Average: 33*

*# Grade: F*

*# Student is failing.*

exam\_one = int(input("Input exam grade one: "))

exam\_two = int(input("Input exam grade two: "))

exam\_three = int(input("Input exam grade three: "))

grades = [exam\_one, exam\_two, exam\_three]

sum = 0

for grade in grades:

sum = sum + grade

avg = sum / len(grades)

if avg >= 90:

letter\_grade = "A"

elif avg >= 80 and avg < 90:

letter\_grade = "B"

elif avg > 69 and avg < 80:

letter\_grade = "C"

elif avg <= 69 and avg >= 60:

letter\_grade = "D"

else:

letter\_grade = "F"

for grade in grades:

print("Exam: " + str(grade))

print("Average: " + str(avg))

print("Grade: " + letter\_grade)

if letter\_grade == "F":

print("Student is failing.")

else:

print("Student is passing.")

**Common Errors and How to Find them:**

With experience you will see that many errors that come up in code, from beginners to expert developers, are usually based on a few common types of problems:

**Syntax:** (ParseError) Check parentheses, quotations, brackets, if/elsif/else statements, braces, commas, etc.

**Variables:** (NameError) Be consistent with using your variable names. They should be defined before they are used for the first time. Use naming schemes that convey their use, so you know what they are meant to do.

**Functions:** (we haven’t covered functions yet, but for later) Know how the functions you are using work. What do they return?

If you get stuck, try these steps to help you through the debugging process:

1 – Use the print statement to print out variable values at different points. This will help you verify if they are returning what you are expecting to see.

2 – Check the coloring in your text editor (IDLE). If the colors don’t look like you would expect them to, that could be the place of the error.

3 – Check the error output in the terminal (shell).

4 – Work through sections of the code piece by piece. This will help you narrow your search.

5 – Read the Python documentation.

6 – Use commenting in your code. This will help make sure you understand what your code is doing.

7 – Ask for help!

**Problems**: For each given program, find the bug and fix it.