Programming Fundamentals

Assessments

Developed by:
James Lane
lanejamr@gmail.com
Summer 2014, 2015, 2016

1. INTRODUCTION

The assessments included in this document go along with the text and progression of the Programming Fundamentals course.

2. CODING SKILLS PREASSESSMENT

The pre-assessment can be used to help identify the relative abilities of the students in class. Allow for student self-selection and pursuit of personal interests as the course progresses.

2.1. EXERCISES

Write a sentence (or two) to define the following computer and programming terminology.

1. CPU -
2. variable -
3. input statement -
4. print statement -
5. if-then-else statement -
6. for loop -
7. while loop -
8. a function or a procedure -
9. compiler -
10. syntax error -
11. run-time error -

For each item below, respond by filling in the blank with or circling the appropriate response.

1. I have written a program for a computer.

   YES NO

2. List the language(s) in which you have written code. If you've never written a computer program, write NONE. _____

3. When it comes to writing computer programs, I would consider my level of experience to be:

   BEGINNER INTERMEDIATE ADVANCED

4. I have written a program for my calculator.

   YES NO

5. I have written a program for my calculator with conditional (if-then) statements.

   YES NO

6. I have written a program for my calculator with looping (for or while) structures.

   YES NO

7. I have written a program for my calculator with graphics displays.

   YES NO

8. I consider myself a computer savvy person.

   YES SOMEWHAT NO

Using any programming language you are comfortable with, write the lines of code that would accomplish each task below. If you have never written programs before, leave these items blank.

1. Write a program that reads in the user's name, and writes "Hello, <User's name>!".

2. Write a program that reads in three whole numbers (integers) and writes out the sum.

3. Write a program that reads in a positive whole number and depending on the number, writes out the following message:

   Write "Too low" if the number is less than or equal to 0
   Write "Child" if the number is greater than 0 AND less than or equal to 12
   Write "Young adult" if number is greater than 12 AND less than 18
   Write "Adult" if number is greater than or equal to 18
   Write "Too high" if number is greater than 100

4. Write a program that writes the following 45 pairs of numbers:

1	1
2	1
2	2
3	1
3	2
3	3
4	1
4	2
4	3
4	4
.	.
.	.
.	.
9	7
9	8
9	9

For each program below, determine the output based on the User input.

5. User input: *Figure* = Triangle, *Base* = 6, *Height* = 4

```
Figure = input("The type of figure is a: ")
Base = input("The base is: ")
Height = input("The height is: ")

if Figure == "Triangle":
      Area = 0.5 * int(Base) * int(Height)
else:
       Area = int(Base) * int(Height)

print ("The area is ", Area)
```

6. User input: *n* = 10

```
total_sum = 0
n = input("Enter a number greater than 0 and less than 20: ")

for c in range(1, int(n), 2):
      total_sum = total_sum + c

print ("The sum of the odds up to " + n + " is: ", total_sum)
```

2.2. SOLUTIONS

Write a sentence (or two) to define the following computer and programming terminology.

1. CPU – the central processing unit is the hardware inside the computer that carries out the instructions, calculations, and input/output operations
2. variable – a variable is a letter or word in a program that names a memory location for storing data
3. input statement – an input statement allows the program to accept information from the user of the program
4. print statement – a print statement displays information to the user, typically on the computer monitor
5. if-then-else statement – this statement controls the flow of a program by making a decision based on a comparison
6. for loop – a for loop executes a series of instructions a prescribed number of times
7. while loop – a while loop executes a series of instructions until a condition is reached in the program
8. a function or a procedure – this is a set of steps that is called and executed within a program, also known as a sub-program
9. compiler – a compiler translates code written in programming language into code that the CPU can understand
10. syntax error – an error in how the lines of code are written, typically using incorrect symbols or commands; syntax errors will prevent code from compiling or successfully being executed
11. run-time error – this error occurs during the execution of the program resulting in incorrect output/solutions

Using whatever programming language you are comfortable with; write the lines of code that would accomplish each task below. (Solutions are written in Python)

1. Write a program that reads in the user's name, and writes "Hello, <User's name>!".

```
Name = input("What is your name? ")
print ("Hello, " + Name + "!")
```

2. Write a program that reads in three whole numbers (integers) and writes out the sum.

```
n1 = input("Enter first number: ")
n2 = input("Enter second number: ")
n3 = input("Enter third number: ")
print ("The sum is: " + str(int(n1) + int(n2) + int(n3)))
```

3. Write a program that reads in a positive whole number and depending on the number, writes out the following message:

```
n = input("Enter a number: ")
Number = int(n)
if Number <= 0:
      print ("Too low")
elif Number > 0 and Number <= 12:
      print ("Child")
elif Number > 12 and Number <18:
      print ("Young adult")
elif Number >= 18 and Number <= 100:
     print ("Adult")
else:
      print ("Too high")
```

4. Write a program that writes the following 45 pairs of numbers:

```
for n in range(1,10):
    for c in range(1,n+1):
        print(str(n) + "    " + str(c))
```

For each program below, determine the output based on the User input.

5. Output: The area is 12

6. Output: The sum…to 10 is: 24

3. ASSESSMENT 1

Students should complete this assessment without the use of a computer after 10 to 14 days. Students should be prepared for this quiz after they have completed Chapters 1 and 2 of Dawson. It is meant to assess a student's ability to read and write Python code using appropriate input, output, and variable assignment statements. The use of the ***eval()*** function at this point in the course, while not conventional, serves to simplify assigning a value as a number, rather than a string. As the course continues its development, students will be taught more conventional ways of writing these statements using functions like ***int()*** or ***str()***. If possible, print this assessment in color so the code is more readable and looks like the Python shell.

Alternatively, this assessment can be made more challenging if a teacher is interested in assessing some critical thinking skills. Exercise 4 can be more open-ended using a modified prompt and/or eliminating the "Note". The following is an example of a modified prompt: write a program that estimates the number of days the user has been alive. Output a statement that includes the user's name and approximately how many days he/she has been alive.

3.1. EXERCISES

Use the program in the box to respond to exercises 1 – 3.

```
1. # This program calculates a school fee based on the
2. # number of classes a student is taking.
3.
4. # User input
5. name = input("What is your name? ")
6. classes = eval(input("How many classes are you taking? "))
7.
8. # Fee calculation
9. fee = 25 * classes
10.
11. # Output
12. print(name + ", you owe a fee of $", fee, ".")
13.
14. input("Hit 'Enter' when you are done.")
```

1. Identify at least one variable that stores data as a string and one variable that stores data as a number.

2. What does the ***eval()*** function in line 6 do?

3. Fill in the Output Table cells with an example of the program executing; all of the cells may not need to be filled in. Include all of the interaction between the user and the computer.

Input prompt	User input
Program output	

4. Write a program that asks the user for their name, birth year, and current year, then estimates the number of days the user has been alive. Output a statement that includes the user's name and approximately how many days he/she has been alive. ****Note: for simplicity, your program may assume each year equates to 365 days.**

Program planning

You do not need to include comments or skipped lines, however, use indentation where appropriate.

1 _____

2 _____

3 _____

4 _____

5 _____

6 _____

7 _____

8 _____

9 _____

10 _____

3.2. SOLUTIONS

1. There are three variables in the program: name (string), classes (numeric), and fee (numeric).
2. The ***eval()*** or ***int()*** function is used to assign the user input as a numeric value so the variable can be used in a calculation.
3. Output Table:

Input prompt	User input
What is your name?	Kim
How many classes are you taking?	6
Program output	
Kim, your fee is $150.	
Hit 'Enter' when you are done.	

4. Program Code:

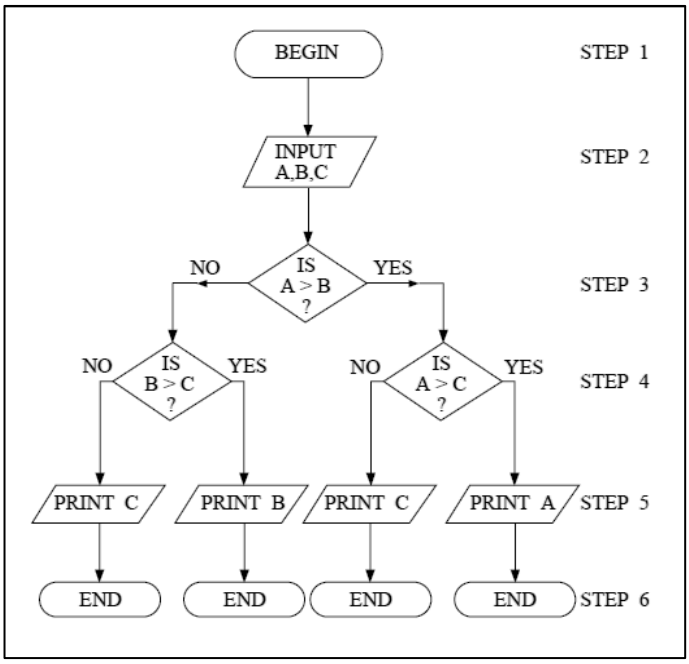
```
name = input("What is your name: ")

birth_year = int(input("What year were you born? "))

year = int(input("Enter today's year: "))
```

4. Assessment 2

Students should complete this assessment without the use of a computer after 10 or 14 days. Students are prepared for this assessment after they have learned about flowcharts and created some of their own flowcharts to solve coding problems. It is meant to assess a student's ability to read and create a flowchart. The steps of analyzing a problem and designing an algorithm are key elements of students successfully writing a program. There are resource documents included with these materials in the Syllabus.

4.1. EXERCISES

Use the flowchart in the box to answer the problems 1 – 2.



1. Here are six sets of values for the variables A, B, and C.

   (a) A=9, B=5, C=1
   (b) A=6, B=3, C=8
   (c) A=2, B=9, C=7
   (d) A=7, B=3, C=9
   (e) A=6, B=3, C=5
   (f) A=4, B=8, C=2

   Indicate which letter is printed for each set of values.

   (a)
   (b)
   (c)
   (d)
   (e)
   (f)

2. What does this procedure do?

3. On the back of this page, construct a flowchart that accepts numeric test grades as input, calculates a simple average, and outputs one of three comments. You may write your own appropriate comments.

   A) "Comment 1!" if the average is greater than or equal to 83
   B) "Comment 2." if the average is greater than or equal to 70 and less than 83
   C) "Comment 3!" if the average is less than 70

4.2. SOLUTIONS

1. (a) Prints A

   (b) Prints C

   (c) Prints B

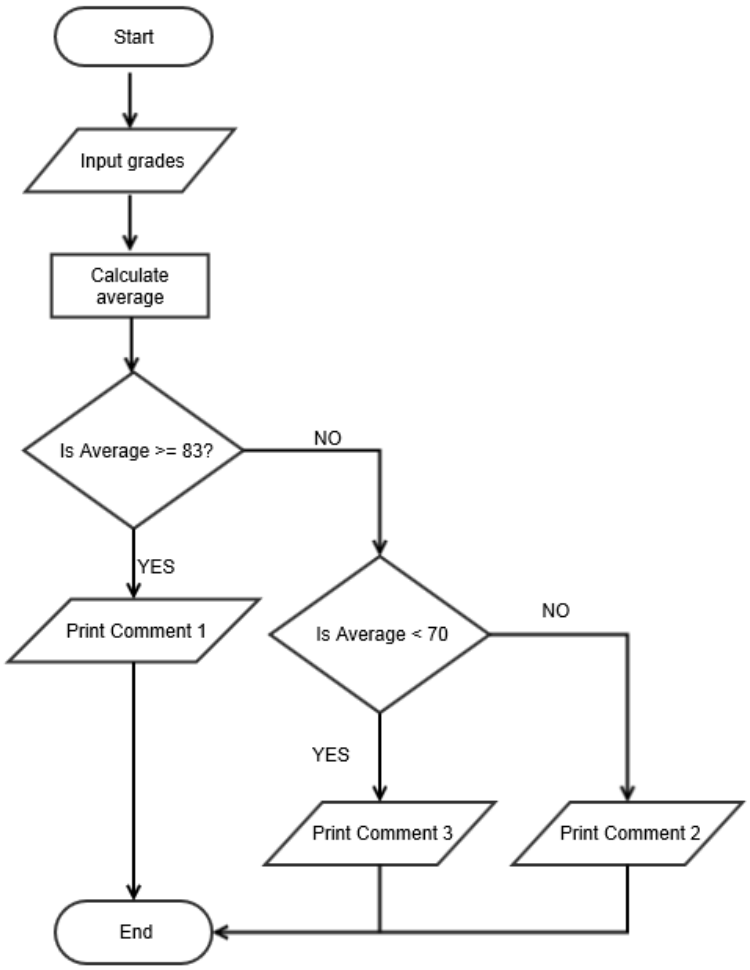
   (d) Prints C

   (e) Prints A

   (f) Prints B

2. Based on the sample data, a student is likely to respond that this procedure prints the largest of the three values input by the user. An attentive student will also comment about what happens when two or three values are equal, however, that is not the point of this item. This item can be used to have a class discussion about paying attention to the finer details of program decision making and addressing exceptions.

3. Flowchart:



5. ASSESSMENT 3

Students should complete this assessment without the use of a computer after 27 – 30 days. Students should be prepared for this quiz after they have completed Chapters 1 through 3 of Dawson. It is meant to assess a student's ability to interpret operators, comparisons, and Booleans in conditional statements and while loops.

5.1. EXERCISES

1. Given a = 5, b = -3, c = 4, d = 60, and e = 30, determine the value of each operation.
   a. a + b * c =
   b. (b + c) * d =
   c. b – d / a =
   d. (a – b) / c =
   e. a ** 2 =
   f. e / c =
   g. e // c =
   h. e % c =

2. Determine the truth of each statement – write True or False.
   a. 5 == 8
   b. not 4 == 7
   c. 6 != 8
   d. not (-3 <= 0)
   e. (7 != 2) and (1 > 0)
   f. (9 < 9) or (6 >= 6)
   g. not (10 <= 1) and (10 == 10)
   h. (4 != 5) or (4 != 10)

3. Using the code in the grey box, write the output.

```
1. num = 240
2. while num > 20:
3.     print(num)
4.     num /= 2
```

4. Using the code in the grey box, write the output.

```
1. num = 10
2. while True:
3.     if num < 7:
4.         break
5.     print(num)
6.     num -= 1
```

5. Using the code in the grey box, write the output.

```
1. num = 1
2. c = 4
3. while c != 0:
4.     num *= 3
5.     c -= 1
6. print(num)
```

6. Consider the program written in the grey box below. Complete the table below, tracing the value of each variable and indicating the output each time the "print" statement is executed.

```
1. n = 10
2. i = 10
3.
4. while i > 0:
5.         print (i)
6.         if i % 2 == 0:
7.             i = i / 2
8.         else:
9.             i += 1
```

Iteration	Output	Variable *n*	Variable *i*
0	-	10	10
1	10		
2			
3			
4			
5			
6			
7			
8			
9			

Without comments, it is difficult to know what this program is doing. There are a number of problems with this code. Besides the lack of comments, describe at least one other problem that makes this poor code to execute. In the box below, state an explanation for what you think the program is supposed to be doing.

5.2. SOLUTIONS

1. Order of operations is critical for each of these answers.

a. -7
b. 60
c. -15
d. 2
e. 25
f. 7.5 in v3.4, 7 in v2.7
g. 7
h. 2

2. Order of comparison is critical for each of these answers.
   a. False
   b. True
   c. True
   d. True
   b. True
   d. False
   f. True
   h. True

3. 240
   120
   60
   30

4. 10
   9
   8
   7

5. 3
   9
   27
   81

6. The program contains an infinite loop. It seems like it is supposed to print numbers divisible by 2.

Iteration	Output	Variable *n*	Variable *i*
0	-	10	10
1	10	10	5
2	5	10	6
3	6	10	3
4	3	10	4
5	4	10	2
6	2	10	1
7	1	10	2
8	2	10	1
9	1	10	2

6. ASSESSMENT 4

For some people, one of the frustrations of programming can be the process of debugging. However, others view the debugging process as a fun challenge. No matter which way you feel, to develop good programming skills, students should be able to examine an error message, interpret the message to determine where the problem occurs, and make corrections in order to eliminate the error in the code.

Students should complete this assessment without the use of a computer after 35 – 40 days. Students should be prepared for this quiz after they have completed Chapters 1 through 4 of Dawson. It is meant to assess a student's ability to examine an error, determine where it occurs in the program, and correct the error. If possible, print this assessment in color so the code is more readable and looks like the Python shell.

Students can complete this assessment individually, or students can work on it individually for a short period of time, then move to small groups to compare and finish the assessment.

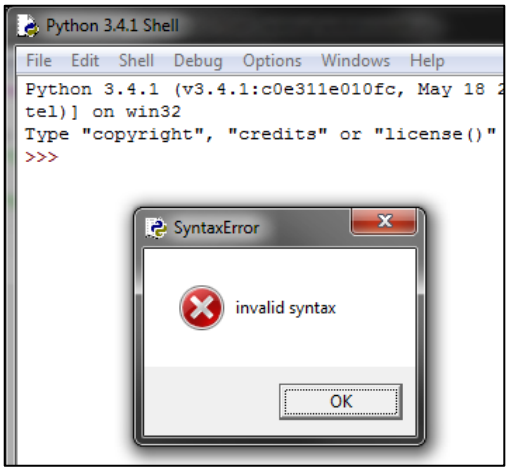
6.1. EXERCISES

Students created the programs that follow. In each instance, either a syntax error or a run-time error has occurred. The program code is located in the grey box on the left side, and the corresponding error message that appeared in the IDLE Shell is on the right side. For each program:

(a) identify the line number where the error occurs, and
(b) explain what you would do to fix the error.

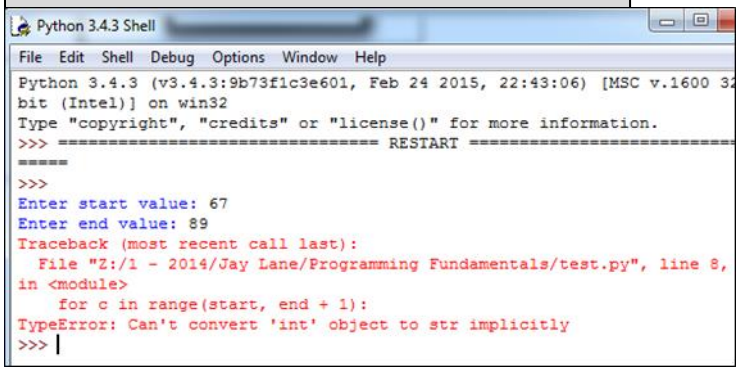
1. Program 1.

```
1. # This program shows the sale price of an item if
2. # its price is greater than 10 dollars.
3.
4. amount = int(input("Amount of purchase: $"))
5.
6. if amount > 10
7.     print("Discounted price: $", 0.9 * amount)
8. else:
9.     print("No discount")
```



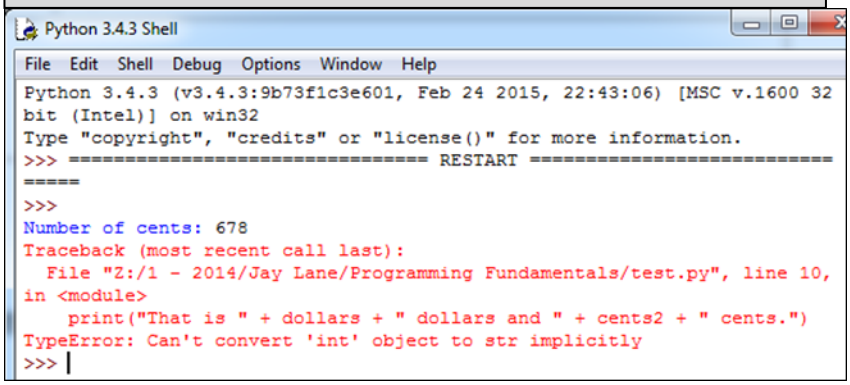
2. Program 2.

```
1. # This program prints a list of numbers
2. # from start value to end value.
3.
4. start = input("Enter start value: ")
5.
6. end = input("Enter end value: ")
7.
8. for c in range(start, end + 1):
9.     print (c)
```



3. Program 3.

```
1. # This program inputs a number of cents and outputs the
2. # amount as dollars and cents.
3.
4. cents = int(input("Number of cents: "))
5.
6. dollars = cents // 100
7.
8. cents2 = cents % 100
9.
10. print("That is " + dollars + " dollars and " + cents2 + " cents.")
```



4. Program 4.

```
1. # This program determines the cost of shipping an
2. # item based on its price and whether it is an
3. # overnight delivery.
4.
5. item = input("Item ordered: ")
6. price = input("Enter the price: $")
7. overnite = input("Overnight delivery (yes or no): ")
8.
9. if price >= 10 and overnite == 'yes':
10.     shipping = 8
11. elif price >= 10 and overnite == 'no':
12.     shipping = 3
13. else:
14.     shipping = 2
15.
16. print("Invoice for - " + item + ": $", price)
17. print("Shipping: $", shipping)
18. print("Total: $", shipping + price)
```

```
Python 3.4.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 3
2 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
======
>>>
Item ordered: shirt
Enter the price: $45
Overnight delivery (yes or no): no
Traceback (most recent call last):
  File "Z:/1 - 2014/Jay Lane/Programming Fundamentals/test.py", line 9,
in <module>
    if price >= 10 and overnite == 'yes':
TypeError: unorderable types: str() >= int()
>>>
```

Sometimes programs run without generating an error, but the output is wrong. This type of situation is referred to as a logic or semantic error. For each program:

(a) identify what caused the output to be incorrect, and
(b) explain what you would do to fix the error.

5. This set of statements converts the number of feet to inches.

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:2
tel)] on win32
Type "copyright", "credits" or "license()" for more inf
>>> def main():
        feet = input("How many feet did you jump? ")
        inches = 12 * feet
        print("You jumped ", inches, " inches.")


>>> main()
How many feet did you jump? 8
You jumped  888888888888  inches.
>>>
```

6. This set of statements prints the numbers from one (1) up to and including ***list_end***.

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> def main():
        list_end = eval(input("Enter a number for the end " \
                              "of the list of numbers. "))
        for c in range(1, list_end):
            print(c)


>>> main()
Enter a number for the end of the list of numbers. 10
1
2
3
4
5
6
7
8
9
>>>
```

6.2. SOLUTIONS

1. Error in line 6…add missing colon (:).
2. Error in line 4 and 6…add ***eval()*** or ***int()*** to ***input()*** statements. The ***range()*** function needs integers for its parameters.
3. Error in line 7…use commas to separate variables from strings; delete plus signs.
4. Error in line 2…add ***eval()*** or ***int()*** to ***input()*** statement. Comparison needs to be done with numeric values.
5. Assignment statement for ***inches*** is multiplying a string by 12, rather than the number of feet by 12. Add ***eval()*** or ***int()*** to ***input()*** statement.
6. The ***range()*** function iterates up to but does not include the end value. Change ***list_end*** to ***list_end*** + 1 so that the last value is printed.