

# **Exposure CS 2021** **for CS1**

## **Chapter 4 Section 8-10 Slides**

**More Errors:  
Syntax, Run-time and Logic**

**PowerPoint Presentation  
created by:  
Mr. John L. M. Schram  
and Mr. Leon Schram  
Authors of Exposure  
Computer Science**



# Section 4.8

## More Syntax Errors

# Syntax Errors

These are errors in the syntax of the program.  
Simple typos cause many syntax errors.  
The interpreter catches ALL syntax errors, but  
the error message is not always accurate.

```
----jGRASP exec: python Variables03.py
Traceback (most recent call last):
  File "Variables03.py", line 6, in <module>
    print(x)
NameError: name 'x' is not defined

----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.
```

```
1 # MoreErrors01.py
2 # This program demonstrates what happens when
3 # you misspell a variable.
4 # Note that the error messages do not always
5 # identify the correct location of the error.
6
7
8 pi = 3.141592653589793
9
10 radisu = 12.5
11
12 circleArea = pi * radius ** 2
13
14 print()
15 print("A circle with a radius of",radius,
"has an area of",circleArea)
```

```
----jGRASP exec: python MoreErrors01.py
Traceback (most recent call last):
  File "MoreErrors01.py", line 12, in <module>
    circleArea = pi * radius ** 2
NameError: name 'radius' is not defined

----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.
```

```
8 pi = 3.141592653589793
9
10 radius = 12.5
11
12 circleArea = pi * radius ** 2
13
14 print()
15 print("A circle with a radius of",radius,
"has an area of",circleArea)
```

```
----jGRASP exec: python MoreErrors01.py
Traceback (most recent call last):
  File "MoreErrors01.py", line 12, in <module>
    circleArea = pi * radius ** 2
NameError: name 'radius' is not defined

----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.
```

```
8 pi = 3.141592653589793
9
10 radisu = 12.5
11
12 circleArea = pi * radius ** 2
13
14 print()
15 print("A circle with a radius of",radius,
"has an area of",circleArea)
```

There is a very important concept that this program demonstrates. Note that the error is caused by using the variable **radius** in line 12. The actual error is in line 10 where **radius** is misspelled.

```
1 # MoreErrors02.py
2 # This program demonstrates what happens
3 # when you do not follow case-sensitivity.
4
5
6 pi = 3.141592653589793
7
8 radius = 12.5
9
10 circleArea = pi * radius ** 2
11
12 print()
13 print("A circle with a radius of",radius,
14 "has an area of",circlearea)
```

```
----jGRASP exec: python MoreErrors02.py
Traceback (most recent call last):
  File "MoreErrors02.py", line 13, in <module>
    print("A circle with a radius of",radius,
"has an area of",circlearea)
NameError: name 'circlearea' is not defined

----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.
```

```
9
10 circleArea = pi * radius ** 2
11
12 print()
13 print("A circle with a radius of",radius,
"has an area of",circlearea)
14
```



```
1 # MoreErrors03.py
2 # The Issue with Addresses
3 # Combining numbers & strings does not work
4
5
6 houseNumber = 811
7 streetName = " Fleming Trail"
8
9 streetAddress = houseNumber + streetName
10
11 print()
12 print(streetAddress)
```

```
----jGRASP exec: python MoreErrors03.py
Traceback (most recent call last):
  File "MoreErrors03.py", line 9, in <module>
    streetAddress = houseNumber + streetName
TypeError: unsupported operand type(s) for +: 'int' and 'str'

----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.
```

```
1 # MoreErrors04.py
2 # Fixing the Address Issue
3 # By Type Casting the <houseNumber> as a string,
4 # the computer is able to concatenate it to the
5 # <streetName> to complete the <streetAddress>.
6
7
8 houseNumber = 811
9 streetName = " Fleming Trail"
10 streetAddress = str(houseNumber) + streetName
11
12 print()
13 print(streetAddress)
14
```

```
-----jGRASP exec: python MoreErrors04  
811 Fleming Trail  
-----jGRASP: operation complete.
```

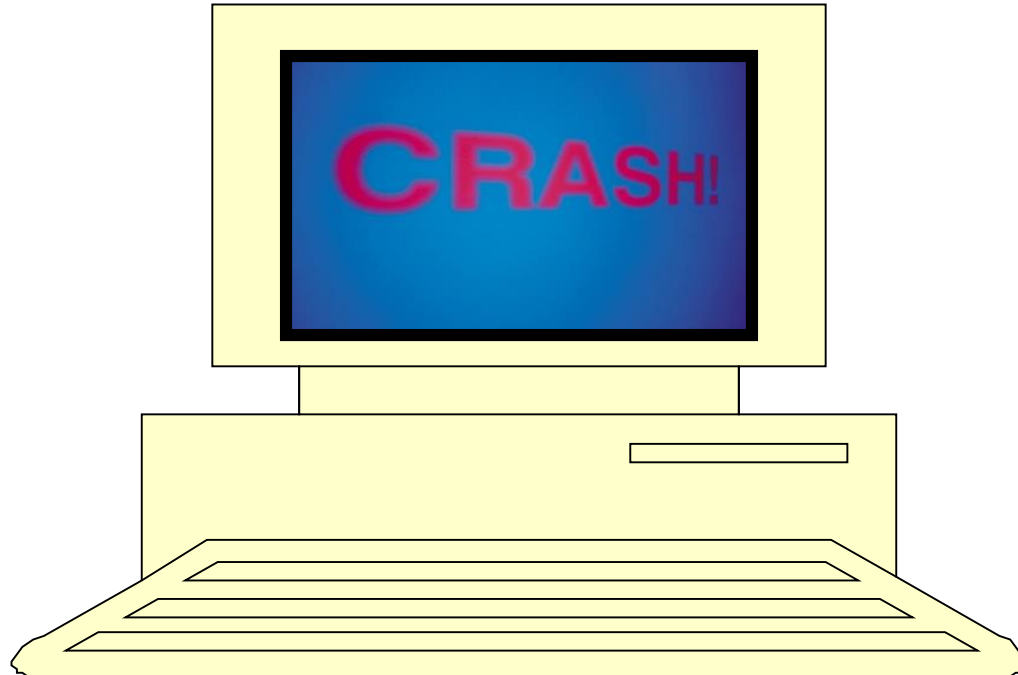
```
7  
8 houseNumber = 811  
9 streetName = " Fleming Trail"  
10 streetAddress = str(houseNumber) + streetName  
11  
12 print()  
13 print(streetAddress)  
14
```

# Section 4.9

# Other Types of Errors

# Run-time/Execution Errors

These errors are not detected until the program runs or executes. They are triggered when an attempt is made to do something improper. Essentially, the program will execute and then it will CRASH.



```
1 # MoreErrors05.py
2 # This program demonstrates a "Run-time Error".
3 # While the program has no Syntax Errors and does
4 # execute, it CRASHES when it attempts to divide by 0.
5
6
7 print()
8 print("Execution Begins")
9 print()
10
11 a = 1
12 b = 0
13 c = a / b
14
15 print("c =", c)
```

```
1 # MoreEr
2 # This pr
3 # While th
4 # execute
```

```
5
6
7 print()
8 print("Execution Begins")
9 print()
10
11 a = 1
12 b = 0
13 c = a / b
14
15 print("c =", c)
```

```
----jGRASP exec: python MoreErrors05.py
```

Execution Begins

Traceback (most recent call last):

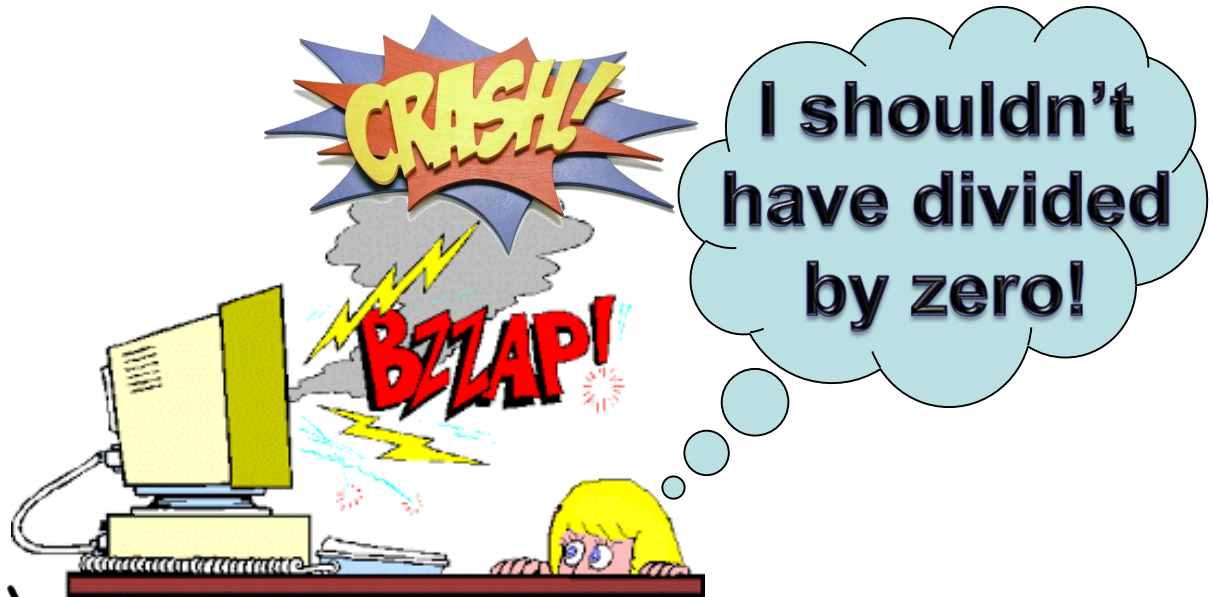
```
File "MoreErrors05.py", line 13, in <module>
```

```
c = a / b
```

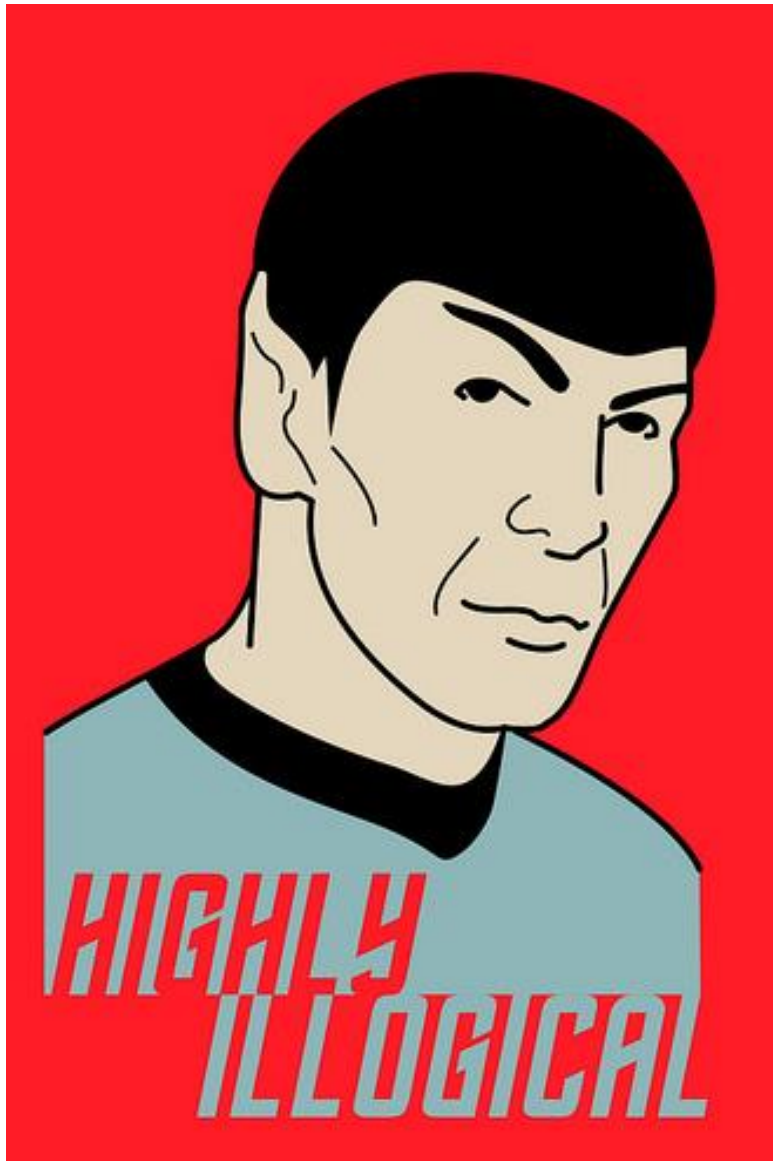
ZeroDivisionError: division by zero

```
----jGRASP wedge2: exit code for process is 1.
```

```
----jGRASP: operation complete.
```



# Logic Errors



These are the most frustrating errors because there are no error messages.

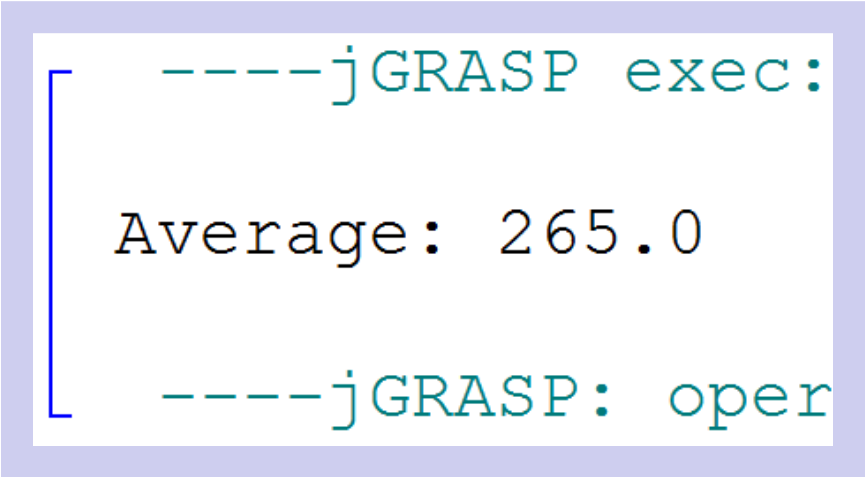
The entire program executes without crashing; however, the program does not do what you want it to do.

This means the logic of your program is not correct.



```
1 # MoreErrors06.py
2 # This program demonstrates a "Logic Error".
3 # These can be the most frustrating errors
4 # because no error is actually detected.
5 # The program executes just fine.
6 # It just does not do what you want it to do.
7 # In this case the average is wrong because
8 # "Order of Operations" was not considered.
9
10
11 num1 = 70
12 num2 = 80
13 num3 = 90
14 num4 = 100
15
16 average = num1 + num2 + num3 + num4 / 4
17
18 print()
19 print("Average:", average)
```

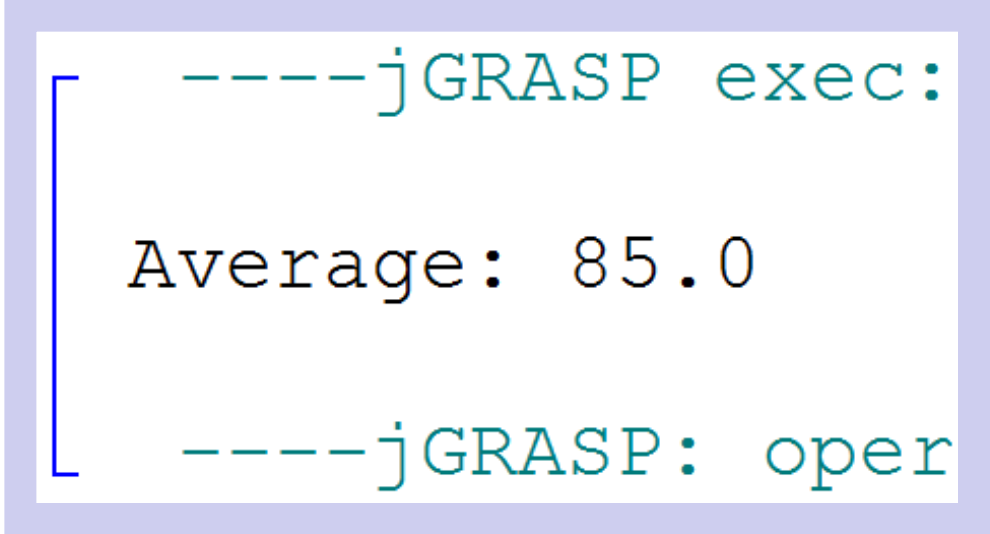
```
1 # MoreErrors06.py
2 # This program demonstrates a "Logic Error".
3 # These can be the most frustrating errors
4 # because no error is actually detected.
5 # The program executes just fine.
6 # It just does not do what you want it to do.
7 # In this case the average is wrong because
8 # "Order of Operations" was not considered.
9
10
11 num1 = 70
12 num2 = 80
13 num3 = 90
14 num4 = 100
15
16 average = num1 + num2 + num3 + num4 / 4
17
18 print()
19 print("Average:", average)
```



----jGRASP exec:  
Average: 265.0  
----jGRASP: oper

```
1 # MoreErrors07.py
2 # This program fixes the PEMDAS issue
3 # of the previous program by adding a
4 # strategic set of parentheses.
5
6
7 num1 = 70
8 num2 = 80
9 num3 = 90
10 num4 = 100
11
12 average = (num1 + num2 + num3 + num4) / 4
13
14 print()
15 print("Average: ",average)
16
```

```
1 # MoreErrors07.py
2 # This program fixes the PEMDAS issue
3 # of the previous program by adding a
4 # strategic set of parentheses.
5
6
7 num1 = 70
8 num2 = 80
9 num3 = 90
10 num4 = 100
11
12 average = (num1 + num2 + num3 + num4) / 4
13
14 print()
15 print("Average: ", average)
16
```



----jGRASP exec:  
Average: 85.0  
----jGRASP: oper

# 3 Types of Errors Review

There are 3 main types of errors with computer programs:

## *Syntax Errors*

```
pwint("Hello)
```

## *Run-time Errors*

```
a = 1  
b = 0  
c = a / b
```

## *Logic Errors*

```
total = 3 + 12  
print("3 dozen =", total)
```



# Asking For Help, Important Note



**There will be times that your program does not work, and you will need to ask for help from your computer science teacher.**

**While you may not know exactly what is wrong with your program, you should at least be able to tell your teacher whether you have a syntax, run-time or logic error.**

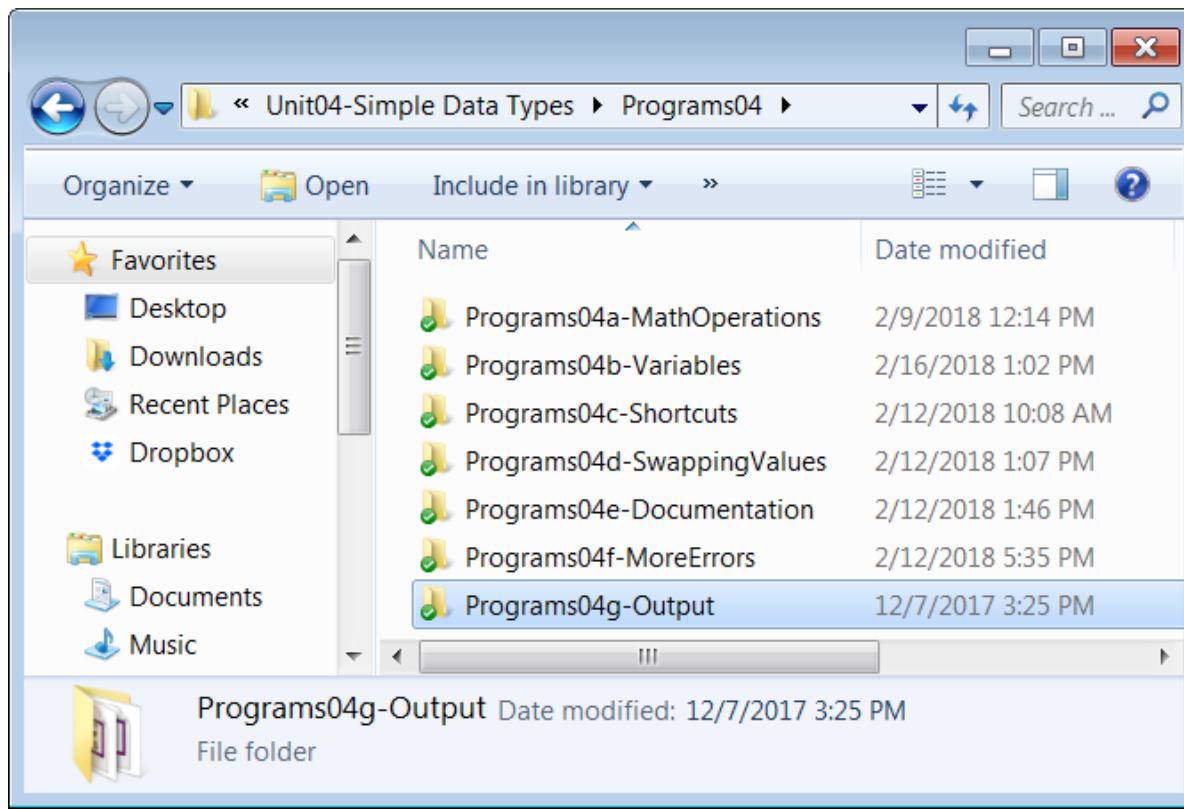
# Section 4.10

Output Programs,

Slides, Exercises & Quizzes

# Output Programs

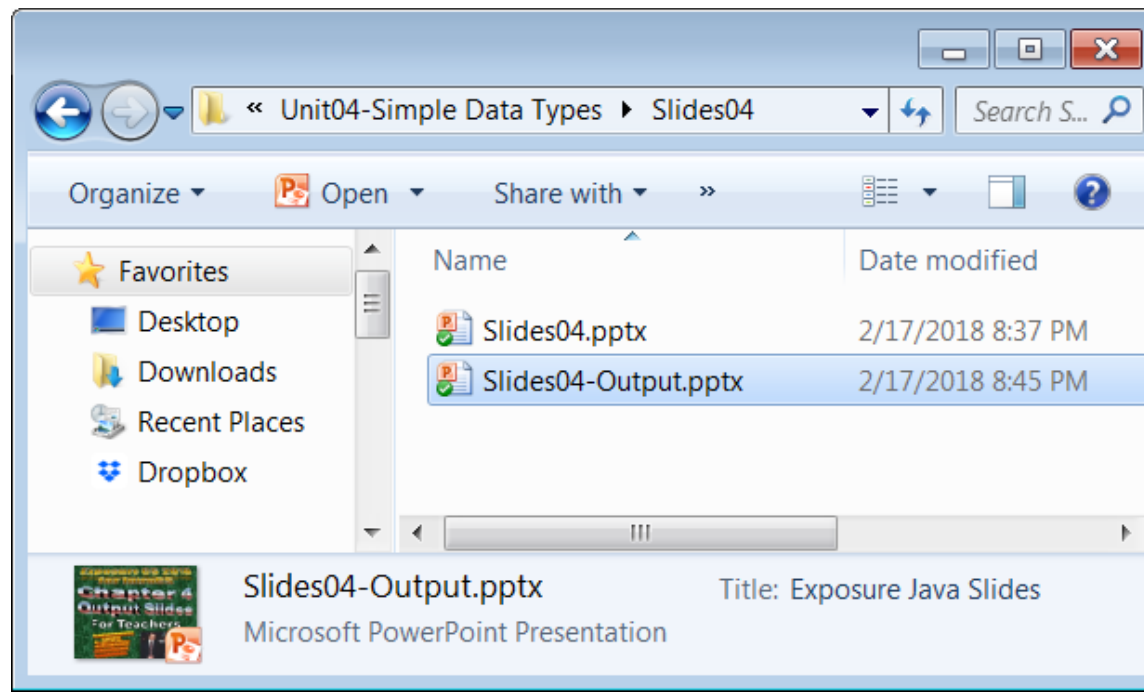
Several chapters in this textbook will end with a special “Output” section. This is the first. Unlike the earlier sections, you will not see these programs explained in this or any chapter.





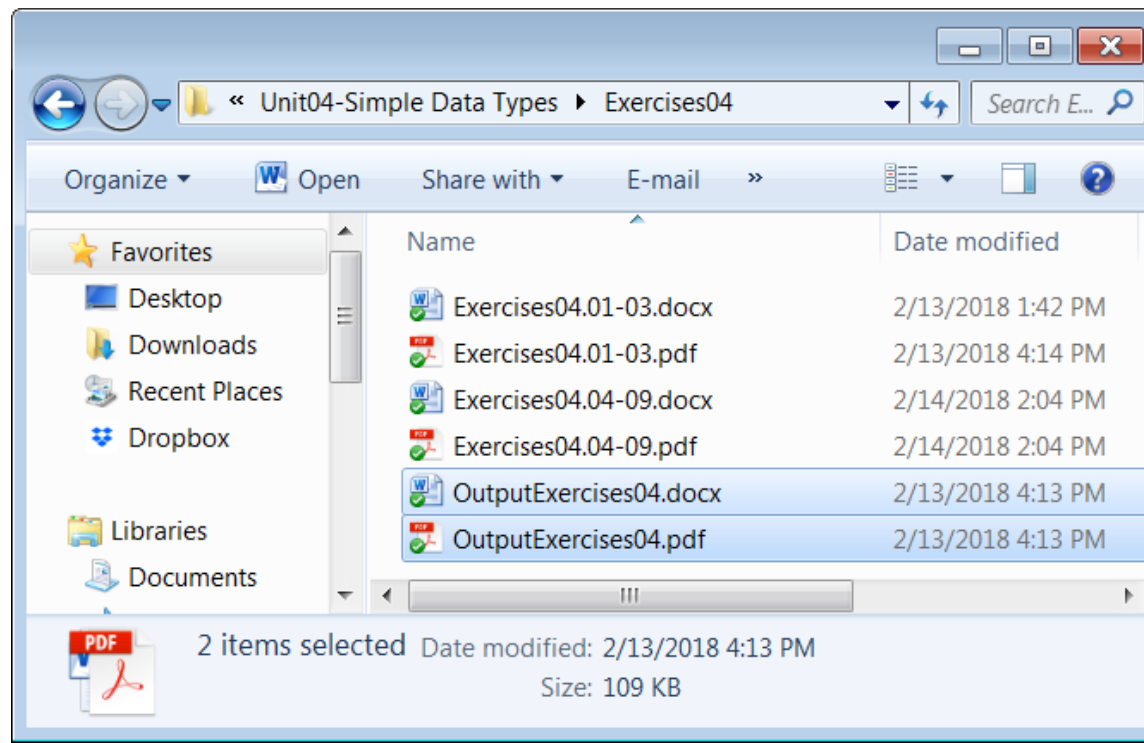
# Output Slides

You will see these programs in a separate PowerPoint presentation. For this chapter, if you look in the **Slides04** folder, you will find this presentation file and another called **Slides04-Output.pptx**. Frequently, this presentation is used as a review before a test.



# Output Exercises

While normal Homework Exercises contain “Short Answer” questions, “Output Exercises” contain programs similar to those shown in the “Output Slides”. As with the slides, students need to determine the output of each program.



# **Output Quizzes**

**These contain programs similar to those shown in the “Output Slides” and “Output Exercises”. As with the other 2, students need to determine the exact output of each program.**