Csci 1523
Spring 2015
Study Guide - Chapter 3 Dierbach
Assigned: 2/08/16
Due: 2/22/16

Student (Print): Amethyst O'Connell

This study guide contains **??** pages (including this cover page) and **??** problems. Check to see if any pages are missing. Enter all requested information on the top of this page, and put your initials on the top of every page, in case the pages become separated.

You may use your books, notes, calculator or internet sources while completing this study guide.

Please try to answer the sections clearly and PRINT your answers legibly.

Chapter 3 Study Guide
As described in your text control structures are categorized into 3 groups:

1. **Sequence** - each instruction executes in the order encountered.
2. **Selection** - the flow of execution is altered depending on logical conditions encountered during program execution.
3. **Iterative** - the flow of execution results in a set of instructions being repeated with the number of repetitions dependent on a logical condition.

Below find a set of questions concerning the syntax and behavior of each of the control structures. Use your text or other resources when answering each of these.

1. A program which consists of a set of instructions which does not contain either a selection or iteration structure is called a:
   sequential control structure; "straight-line program"

2. Explain briefly in the space below what is meant by a *sequence structure*
   Things happen in the order that they're written, so like, those historical fiction series

3. Explain in the space provided below the difference between a *control statement* and a *control structure*
   A control statement is the statement that determines the control flow
   the building blocks of a control structure

4. In Python a boolean data type contains one of two values. From the choices below select the values that are implemented in Python:

   A. TRUE
   B. true
   C. True
   D. 1

E. FALSE

F. false

G. False

H. 0

5. Relational expressions evaluate to:

A. Either 0 or 1

B. True or False

C. An alphanumeric, depending on the expressions

D. Nothing, they just change the order of operations

E. None of the above

6. Which of the following is the *"is not equal to"* relational operator:

A. ==

B. !=

C. >

D. <

E. >=

F. <=

G. <>

H. eq

I. neq

7. Which of the following is the *"is equal to"* relational operator:

A. ==

B. =

C. !=

D. >

E. <

F. <>

G. eq

H. neq

8. Which of the following is the *"greater than or equal to"* relational operator:

A. ==

B. =

C. !=

D. >

E. <

F. <>

G. eq

H. neq

I. None of the above.

9. When using relational operators to compare strings Python utilizes a character ordering scheme referred to as:

lexographical (dictionary) ordering

10. Python uses a set of membership operators to determine whether or not a particular collection contains an element or not. From the list below select the correct operators:

A. ==

B. in

C. $x >$ *alist*

D. >

E. =

F. not in

G. $x! =$ *alist*

H. None of the above.

11. In the space provided below explain the difference between a *relational* and a *boolean* operator in Python.

relational operators work by comparing values, whereas boolean operators work like gates in digital electronics, and work by basically computing the results of the results of relational operators

12. Complete the truth tables shown below:

    Boolean "or" operator:

Arg 1	*Arg 2*	*Result*
True	True	True
True	False	True
False	True	True
False	False	False

    Boolean "and" operator:

Arg 1	*Arg 2*	*Result*
True	True	True
True	False	False
False	True	False
False	False	False

13. Looking over the tables completed in the previous question explain the function of a *"short circuited"* operator in Python in the context of those tables:

    If you have an and gate, and you get a false, you could, and python does ignore the second operation and just evaluate the thing as false, so what this means is that if you're using boolean expression, you should use the most important one first.

14. In the space provided below list the arithmetic, logical and relational operators defined in the Python language in order of their execution precedence, (See Figure 3-6 in your text):

    ** (exponentiation, rtl), - (negation, ltr), * / // % (multi, div, truncating div modulo, ltr), + - (addition, subtration, ltr), < > <= >= != == (relational operators), not (ltr), and (ltr), or (ltr)

15. Define a *selection* control statement:
a statement that provides selective control. Basically, running a piece of a program is reliant on a relational operation.

16. Given the code listing shown below:

Listing 1: *if* structure and relational operators

```
1  exam_score = 65
2  student_present = True
3
4  if (exam_score > 89 and student_present):
5      grade = 'A'
6  elif (exam_score > 79 and student_present):
7      grade = 'B'
8  elif (exam_score > 69 and student_present):
9      grade = 'C'
10 elif (exam_score > 59 and student_present):
11     grade = 'D'
12 else:
13     grade = 'F'
14
15 print(grade)
```

(a) What value will be output to the terminal:
C

(b) Rewrite the code segment given above using *nested if-else* statements in place of the *elif* blocks:

```
exam_score = 65
student_present = True
if (exam_score > 89 and student_present) :
    print('A')
else:
    if (exam_score > 79 and student_present) :
        print('B')
    else:
        if (exam_score > 69 and student_present) :
            print('C')
        else:
            if (exam_score > 59 and student_present) :          ··· 16b.py
```

17. In Python we use logic structures which contain *headers* and *clauses*. In the space provided describe each:

a header is a specific keyword followed by a colon, like if: or else: which is followed by a suite, basically a group. It is important to indent the suite all onto the same line, which groups it together. The suite and it's header together are known as a clause.

18. Which of the following are *headers* in Python:

A. y = True
B. if (*condition*):
C. x not z
D. else:
E. int
F. str
G. elif:
H. None of the above.
I. All of the above.

19. What is an *iterative* control structure:
a structure that allows the repeated execution of a set of instructions, like how my robot that I made for Minnehacks drives forward, then makes a right turn, and then repeats, drawing a square-ish repeatedly

20. List the *iterative* control stuctures implemented in Python:
while: allows you to repeat a program over and over until the condition in while doesn't check out.

21. Given the code listing shown below:

Listing 2: *if* structure and relational operators

```
1 i = 6
2
3 while (i >= 0):
4     j = i + i
5     i -= 1
6 print(i,j)
```

(a) When the *while* structure terminates, what values print out for i and j:
i = -1, j=0

(b) Suppose we change the value of i in the example above to 19, what values print out for i and j:
i = -1, j=0

22. Infinite loops:

(a) What is an *infinite* loop:
a loop that does not end, it repeats over and over again.

(b) Give a coding scenario in which we would like to implement an *infinite* loop:
I mean, you probably don't ever want an infinite loop, but let's say you want your robot to drive in a square forever. You could have an infinite loop that would have it drive forward and then take a 90 degree right turn over and over again. Nothing really goes on forever though...

(c) In Listing 2 above which line would need to be modified in order that it would become an infinite loop:

delete i -= 1 from line 5.

. . . . . .

. . . . . .