HW – Week 2  
Saeed Maleki

1. What possible values can a Boolean expression have?  
   False  
   True
2. Where does the term Boolean originate?  
   The term "Boolean" originates from the work of George Boole, who invented Boolean Logic in the 19th century.  
   In computing, it refers to data items that can have a value of "true" or "false"
3. What is an integer equivalent to True in Python?  
   In Python, the integer equivalent of true is 1 .
4. What is the integer equivalent to False in Python?  
   ‌In Python, the integer equivalent of false is 0
5. Is the value -16 interpreted as True or False?  
   In Python, the value -16 is interpreted as true
6. Given the following definitions:   
   x, y, z = 3, 5, 7   
   evaluate the following Boolean expressions:  
   (a) x == 3   
   (b) x < y   
   (c) x >= y   
   (d) x <= y   
   (e) x != y - 2   
   (f) x < 10   
   (g) x >= 0 and x < 10   
   (h) x < 0 and x < 10   
   (i) x >= 0 and x < 2   
   (j) x < 0 or x < 10   
   (k) x > 0 or x < 10   
   (l) x < 0 or x > 10  
     
   (a) true  
   (b) true  
   (c) false  
   (d) false  
   (e) false  
   (f) true  
   (g) true  
   (h) false  
   (i) false  
   (j) true  
   (k) true  
   (l)false
7. Given the following definitions:   
   x, y = 3, 5   
   b1, b2, b3, b4 = True, False, x == 3, y < 3   
   evaluate the following Boolean expressions:  
     
   (a) b3     
   #evaluates to True  
   (b) b4     
   #evaluates to False   
   (c) not b1   
   #evaluates to False   
   (d) not b2   
   #evaluates to true   
   (e) not b3   
   #evaluates to False   
   (f) not b4   
   #evaluates to true   
   (g) b1 and b2    
   #evaluates to false  
   (h) b1 or b2     
   #evaluates to True  
   (i) b1 and b3    
   #evaluates to True  
   (j) b1 or b3   
   #evaluates to true  
   (k) b1 and b4   
   #evaluates to False  
   (l) b1 or b4   
   #evaluates to True  
   (m) b2 and b3   
   #evaluates to True  
   (n) b2 or b3   
   #evaluates to true  
   (o) b1 and b2 or b3   
   #evaluates to False  
   (p) b1 or b2 and b3   
   #evaluates to True  
   (q) b1 and b2 and b3   
   #evaluates to false  
   (r) b1 or b2 or b3   
   #evaluates to True  
   (s) not b1 and b2 and b3   
   #evaluates to True  
   (t) not b1 or b2 or b3   
   #evaluates to True  
   (u) not (b1 and b2 and b3)   
   #evaluates to False  
   (v) not (b1 or b2 or b3)  
   #evaluates to false  
   (w) not b1 and not b2 and not b3   
   #evaluates to false  
   (x) not b1 or not b2 or not b3   
   #evaluates to false  
   (y) not (not b1 and not b2 and not b3)   
   #evaluates to false  
   (z) not (not b1 or not b2 or not b3)  
   #evaluates to True
8. Express the following Boolean expressions in simpler form; that is, use fewer operators or fewer   
   symbols. x is an integer.  
     
   (a) not (x == 2)   
   (b) x < 2 or x == 2   
   (c) not (x < y)   
   (d) not (x <= y)   
   (e) x < 10 and x > 20   
   (f) x > 10 or x < 20   
   (g) x != 0   
   (h) x == 0  
   Reply:  
   (a') not x == 2   
   (b') x <= 2   
   (c') x >= y   
   (d') x > y   
   (e') x  
   (f') x > 10 or x < 20   
   (g') not x == 0   
   (h') x != 0
9. Express the following Boolean expressions in an equivalent form without the not operator. x and y   
   are integers.  
     
   (a) not (x == y)   
   (b) not (x > y)   
   (c) not (x < y)   
   (d) not (x >= y)   
   (e) not (x <= y)   
   (f) not (x != y)   
   (g) not (x != y)   
   (h) not (x == y and x < 2)   
   (i) not (x == y or x < 2)   
   (j) not (not (x == y))  
     
         Reply:  
     
   (a') x != y   
   (b') x <= y   
   (c') x >= y   
   (d') x < y   
   (e') x > y  
   (f') x == y   
   (g') x != y   
   (h') x != y or x >= 2   
   (i') not (x != y and x >= 2)
10. What is the simplest tautology?  
    The simplest tautology is a Boolean expression that is always true  
    . Examples of tautologies include "True" (TRUE, true, 1 or whatever, depending on language or field)  
    , and statements that are always true regardless of the individual statements
11. What is the simplest contradiction?  
    The simplest contradiction is a Boolean expression that is always false  
    . Examples of contradictions include "False" (FALSE, false, 0 or whatever, depending on language or field)  
    , and statements that are always false regardless of the individual statements
12. Write a Python program that requests an integer value from the user. If the value is between 1 and   
    100 inclusive, print ”OK;” otherwise, do not print anything!  
      
    value = int(input("Enter an integer: "))  
    if 1 <= value <= 100:  
        print("OK")
13. Write a Python program that requests an integer value from the user. If the value is between 1 and   
    100 inclusive, print ”OK;” otherwise, print ”Out of range.”  
      
    value = int(input("Enter an integer: "))  
    if 1 <= value <= 100:  
        print("OK")  
    else:  
        print("Out of range.")
14. Write a Python program that allows a user to type in an English day of the week (Sunday, Monday,   
    etc.). The program should print the Spanish equivalent, if possible.  
      
    days = {  
        "sunday": "domingo",  
        "monday": "lunes",  
        "tuesday": "martes",  
        "wednesday": "miércoles",  
        "thursday": "jueves",  
        "friday": "viernes",  
        "saturday": "sábado"}day =  
    input("Enter a day of the week: ")  
    lower()if day in days:  
        print(days[day])  
    else:  
        pass
15. Consider the following Python code fragment:  
    if i < j:   
    if j < k:   
    i = j   
    else:   
    j = k   
    else:   
    if j > k:   
    j = i   
    else:   
    i = k   
    print("i =", i, " j =", j, " k =", k)  
      
    What will the code print if the variables i, j, and k have the following values?  
      
    (a) i is 3, j is 5, and k is 7  
    :The code fragment prints "i = 3 j = 5 k = 7"  
      
    (b) i is 3, j is 7, and k is 5  
    :The code fragment prints "i = 5 j = 7 k = 5"  
      
    (c) i is 5, j is 3, and k is 7  
    :The code fragment prints "i = 7 j = 3 k = 7"  
      
    (d) i is 5, j is 7, and k is 3  
    :The code fragment prints "i = 7 j = 3 k = 3"  
      
    (e) i is 7, j is 3, and k is 5  
    :The code fragment prints "i = 7 j = 5 k = 5"  
      
    (f) i is 7, j is 5, and k is 3  
    :The code fragment prints "i = 5 j = 5 k = 3"
16. Consider the following Python program that prints one line of text:   
    val = int(input())   
    if val < 10:   
    if val != 5:   
    print("wow ", end='')   
    else:   
    val += 1   
    else:   
    if val == 17:   
    val += 10   
    else:   
    print("whoa ", end='')   
    print(val)  
      
    What will the program print if the user provides the following input?  
      
    (a) 3  
    The program prints "wow" since val is less than 10 and not equal to 5  
    #val:  
    3  
      
    (b) 21  
    The program prints "whoa" since val is greater than or equal to 10 and not equal to 17.   
    #val:  
    21  
      
    (c) 5  
    This program prints "6" because val is less than 10 and equal to 5  
    #val:  
    5  
      
    (d) 17  
    This program prints "27" because val is equal to 17!  
    #val:  
    17  
      
    (e) -5  
    The program prints "wow" since val is less than 10 and not equal to 5.  
    #val:  
    -5
17. Consider the following two Python programs that appear very similar:  
    n = int(input())   
    if n < 1000:   
    print('\*', end='')   
    if n < 100:   
    print('\*', end='')   
    if n < 10:   
    print('\*', end='')   
    if n < 1:   
    print('\*', end='')   
    print()  
      
    n = int(input())   
    if n < 1000:   
    print('\*', end='')   
    elif n < 100:   
    print('\*', end='')   
    elif n < 10:   
    print('\*', end='')   
    elif n < 1:   
    print('\*', end='')   
    print()  
      
    How do the two programs react when the user provides the following inputs?   
    (a) 0   
    (b) 1   
    (c) 5   
    (d) 50   
    (e) 500   
    (f) 5000   
    Why do the two programs behave as they do?  
    (a) 0  (1) ---> \*\*\*\* && (2) ---> \*  
      
    (b) 1  (1) ---> \*\*\* &&(2) ---> \*  
      
    (c) 5 (1) ---> \*\*\* && (2) ---> \*  
      
    (d) 50  (1) ---> \*\* && (2) ---> \*  
      
    (e) 500  (1) ---> \* &&          (2) ---> \*  
    (f) 5000 (1) --->         (2) --->
18. Write a Python program that requests five integer values from the user. It then prints the maximum   
    and minimum values entered. If the user enters the values 3, 2, 5, 0, and 1, the program would   
    indicate that 5 is the maximum and 0 is the minimum. Your program should handle ties properly; for   
    example, if the user enters 2, 4, 2, 3, and 3, the program should report 2 as the minimum and 4 as   
    maximum.  
      
    numbers = []  
    for i in range(5):            numbers.append(int(input("Enter an integer: ")))  
    print(f"Maximum: {max(numbers)}")  
    print(f"Minimum: {min(numbers)}")
19. Write a Python program that requests five integer values from the user. It then prints one of two things:  
                if any of the values entered are duplicates, it prints "DUPLICATES"; otherwise, it prints "ALL UNIQUE".  
      
                    < THE CODE >  
      
                    n1 = int(input("num 1 --> "))  
                    n2 = int(input("num 2 --> "))  
                    n3 = int(input("num 3 --> "))  
                    n4 = int(input("num 4 --> "))  
                    n5 = int(input("num 5 --> "))  
          
                    dublicated = 0  
                    if n1 == n2 :  
                        dublicated += 1  
                    if n1 == n3 :  
                        dublicated += 1  
                    if n1 == n4 :  
                        dublicated += 1  
                    if n1 == n5 :  
                        dublicated += 1  
                    if n2 == n3 :  
                        dublicated += 1  
                    if n2 == n4 :  
                        dublicated += 1  
                    if n2 == n5 :  
                        dublicated += 1   
                    if n3 == n4 :  
                        dublicated += 1  
                    if n3 == n5 :  
                        dublicated += 1  
                    if n4 == n5 :  
                        dublicated += 1  
                          
                    if dublicated != 0 :  
                        print(" \* DUPLICATES  \*")  
                    else :  
                        print("ALL UNIQUE")