# Homework 2 – Week 2

# Python Language

1. What possible values can a Boolean expression have?

A Boolean expression can have two values: true and false (Boolean literals).

2. Where does the term Boolean originate?

 George Boole, an English mathematician in the 19th century, developed "Boolean Logic" in order to combine certain concepts and exclude certain concepts when searching databases.

3. What is an integer equivalent to True in Python?

True == 1

4. What is the integer equivalent to False in Python?

False == 0

5. Is the value -16 interpreted as True or False?

In the boolean type, any numeric value is True, but only zero, 0 is False.

6. Given the following definitions: x, y, z = 3, 5, 7 evaluate the following Boolean expressions:

(a) x == 3 True

(b) x < y True

(c) x >= y False

(d) x <= y True

(e) x != y - 2 False

(f) x < 10 True

(g) x >= 0 and x < 10 True

(h) x < 0 and x < 10 False

(i) x >= 0 and x < 2 False

(j) x < 0 or x < 10 True

(k) x > 0 or x < 10 True

(l) x < 0 or x > 10 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 True

(b) b4 False

(c) not b1 False

(d) not b2 True

(e) not b3 False

(f) not b4 True

(g) b1 and b2 False

(h) b1 or b2 True

(i) b1 and b3 True

(j) b1 or b3 True

(k) b1 and b4 False

(l) b1 or b4 True

(m) b2 and b3 False

(n) b2 or b3 True

(o) b1 and b2 or b3 True

(p) b1 or b2 and b3 True

(q) b1 and b2 and b3 False

(r) b1 or b2 or b3 True

(s) not b1 and b2 and b3 False

(t) not b1 or b2 or b3 True

(u) not (b1 and b2 and b3) True

(v) not (b1 or b2 or b3) False

(w) not b1 and not b2 and not b3 False

(x) not b1 or not b2 or not b3 True

(y) not (not b1 and not b2 and not b3) True

(z) not (not b1 or not b2 or not b3) False

8. Express the following Boolean expressions in simpler form; that is, use fewer operators. x is an integer.

(a) not (x == 2) x != 2

(b) x < 2 or x == 2 x <= 2

(c) not (x < y) x >= y

(d) not (x <= y) x > y

(e) x < 10 and x > 20 False

(f) x > 10 or x < 20 True

(g) x != 0 True

(h) x == 0 False

9. Express the following Boolean expressions in an equivalent form without the not operator. x and y are integers.

(a) not (x == y) x != y

(b) not (x > y) x <= y

(c) not (x < y) x >= y

(d) not (x >= y) x < y

(e) not (x <= y) x > y

(f) not (x != y) x == y

(g) not (x != y) x == y

(h) not (x == y and x < 2) (x != y or x >= 2)

(i) not (x == y or x < 2) (x != y and x >=2)

(j) not (not (x == y)) x == y

10. What is the simplest tautology?

If it's more about the outcome, a tautology simply means, it's always true. So "True" (TRUE, true, 1 or whatever, depending on language or field) would be the simplest tautology value wise.

11. What is the simplest contradiction?

while "False" would be the simplest contradiction by the opposite line of reasoning.

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.

P03 in my Github

x = int(input("Enter an integer value: "))  
if 1 < x < 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.”

P04 in my Github

x = int(input("Enter an integer value: "))  
if 1 < x < 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.

P05 in my Github

x = str(input("Enter the days of the week to see the Spanish equivalent: "))  
if x == "Monday":  
 print("lunes")

elif x == "Tuesday":  
 print("Martes")  
elif x == "Thursday":  
 print("Jueves")  
elif x == "Wednesday":  
 print("Miércoles")  
elif x == "Thursday":  
 print("Jueves")  
elif x == "Friday":  
 print("Viernes")  
elif x == "Saturday":  
 print("sábado")

elif x == "Sunday":  
 print("domingo")  
else:  
 print("This entry is not from weekdays.")

15. Consider the following Python code fragment:

# i, j, and k are numbers

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 i is 5 , j is 5, k is 7

(b) i is 3, j is 7, and k is 5 i is 3, j is 5, k is 5

(c) i is 5, j is 3, and k is 7 i is 7, j is 3, k is 7

(d) i is 5, j is 7, and k is 3 i is 5, j is 3, k is 3

(e) i is 7, j is 3, and k is 5 i is 5, j is 3, k is 5

(f) i is 7, j is 5, and k is 3 i is 7, j is 7, k is 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 wow

(b) 21 whoa 21

(c) 5 nothing

(d) 17 nothing

(e) -5 wow

17. Consider the following two Python programs that appear very similar:

1:

n = int(input())  
if n < 1000:  
print('\*', end='')  
if n < 100:  
print('\*', end='')  
if n < 10:  
print('\*', end='')  
if n < 1:  
print('\*', end='')  
print()

2:

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?

1: 2:

(a) 0 \*\*\*\* (a) 0 \*

(b) 1 \*\*\* (b) 1 \*

(c) 5 \*\*\* (c) 5 \*

(d) 50 \*\* (d) 50 \*

(e) 500 \* (e) 500 \*

(f) 5000 nothing (f) 5000 nothing

Why do the two programs behave as they do?

In the first case, because we only have the if structure, the condition is such that all lines of the program are checked and the stars are placed together, but in the second case, because we have if and elif, the lines are checked one by one. If the first condition is not discussed, it goes to The next condition and a star will be printed.

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.

P06 in my Github

a = int(input("Enter first value : "))  
b = int(input("Enter second value : "))  
c = int(input("Enter third value : "))  
d = int(input("Enter forth value : "))  
e = int(input("Enter fifth value : "))  
maximum = a  
minimum = a  
if b >= maximum:  
 maximum = b  
elif b <= minimum:  
 minimum = b  
  
if c >= maximum:  
 maximum = c  
elif c <= minimum:  
 minimum = c  
  
if d >= maximum:  
 maximum = d  
elif d <= minimum:  
 minimum = d  
  
if e >= maximum:  
 maximum = e  
elif e <= minimum:  
 minimum = e  
  
print("minimum:",minimum," & maximum:", maximum)

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".

P07 in my Github

a = int(input("first number: "))  
b = int(input("second number: "))  
c = int(input("third number: "))  
d = int(input("forth number: "))  
e = int(input("fifth number: "))  
if a==b or a==c or a==d or a==e:  
 print("DUPLICATES : ",a)  
elif b==a or b==c or b==d or b==e:  
 print("DUPLICATES : ",b)  
elif c==a or c==b or c==d or c==e:  
 print("DUPLICATES : ",c)  
elif e==a or e==b or e==c or e==d:  
 print("DUPLICATES : ",e)  
else:  
 print("ALL UNIQUE")