**Module 8: Lab Activity – Booleans**

**Joel Navarrete**

**CSS 225**

**Deliverables:**

* Python program solutions to the following 7 problems

**Make Sure You:**

* Add comments

# Your name

# The date

# What the program does

* Test your program
* Fix any bugs (try out the debugging techniques you read about)

Game Example from How to Think Like a Computer Scientist:

For example, suppose we can slay the dragon only if our magic lightsabre sword is charged to 90% or higher, and we have 100 or more energy units in our protective shield. We find this fragment of Python code in the game:

|  |  |
| --- | --- |
|  | **if** **not** ((sword\_charge >= 0.90) **and** (shield\_energy >= 100)):  print("Your attack has no effect, the dragon fries you to a crisp!")  **else**:  print("The dragon crumples in a heap. You rescue the gorgeous princess!") |

de Morgan’s laws together with the logical opposites would let us rework the condition in a (perhaps) easier to understand way like this:

|  |  |
| --- | --- |
|  | **if** (sword\_charge < 0.90) **or** (shield\_energy < 100):  print("Your attack has no effect, the dragon fries you to a crisp!")  **else**:  print("The dragon crumples in a heap. You rescue the gorgeous princess!") |

We could also get rid of the not by swapping around the then and else parts of the conditional. So here is a third version, also equivalent:

|  |  |
| --- | --- |
|  | **if** (sword\_charge >= 0.90) **and** (shield\_energy >= 100):  print("The dragon crumples in a heap. You rescue the gorgeous princess!") **else**:  print("Your attack has no effect, the dragon fries you to a crisp!") |

**Problem 1** – Write a function that takes two inputs from a user and prints whether they are equal or not.

*#author: Joel Navarrete  
#This program compares two numbers to see if they're equal or not*def compare(n1, n2):  
 if n1 == n2:  
 print(**"Numbers are equal"**)  
 else:  
 print(**"Numbers are not equal"**)  
  
def main():  
  
 num1 = int(input(**"Enter first number: "**))  
 num2 = int(input(**"Enter second number: "**))  
  
 compare(num1, num2)  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()M8

**Problem 2** – Write a function that takes two inputs from a user and prints whether the sum is greater than 10, less than 10, or equal to 10.

*#author: Joel Navarrete  
#This program compares two numbers to see if they're equal or not to 10*def compare(n1, n2):  
 n3 = n1 +n2  
  
 if n3 > 10:  
 print(**"Greater than 10"**)  
 elif n3 < 10:  
 print(**"Less than 10"**)  
 else:  
 print(**"Equal to 10"**)  
  
def main():  
  
 num1 = int(input(**"Enter first number: "**))  
 num2 = int(input(**"Enter second number: "**))  
  
 compare(num1, num2)  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

**Problem 3** – Write a function that takes a list and prints if the value 5 is in that list.  
def compare\_list(list):  
 if 5 in list:  
 print(**"5 in the list"**, list)  
 else:  
 print(**"No 5s in the list"**, list)  
  
def main():  
  
 list = [4, 5, 6, 7, 8, 9]  
 list2 = [4, 6, 7, 8, 9, 10]  
  
 compare\_list(list)  
 compare\_list(list2)  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

**Problem 4** – Write a function that takes a year as a parameter and returns True if the year is a leap year, False if it is otherwise.

Consider the requirements of a leap year:

* The year is evenly divisible by 4
* If the year can be evenly divided by 100 it is NOT a leap year, unless:
  + If the year is also evenly divisible by 400, then it is a leap year.
* *#author: Joel Navarrete  
  #This program lets the user know if a specific year is a leap year or not*def leap\_year(year):  
   if (year % 4) == 0:  
   if (year % 100) == 0:  
   if (year % 400) == 0:  
   *#print("{0} is a leap year".format(year))* return True  
   else:  
   *#print("{0} is not a leap year".format(year))* return False  
   else:  
   *#print("{0} is a leap year".format(year))* return True  
   else:  
   *#print("{0} is not a leap year".format(year))* return False  
    
  def main():  
    
   year = int(input(**"Enter a year: "**))  
   result = leap\_year(year)  
   if leap\_year(year):  
   print(**"{0} is a leap year"**.format(year))  
   else:  
   print(**"{0} is not a leap year"**.format(year))  
    
  if \_\_name\_\_ == **"\_\_main\_\_"**:  
   main()

**Problem 5** - Write a function that checks whether your game character has picked up all the items needed to perform certain tasks and checks against any status debuffs. Confirm checks with print statements.

Game Character has the following item list: [pan, paper, idea, rope, groceries]

Game Character has the following status debuffs: [slow]

Task 1: Climb a mountain – needs rope, coat, and first aid kit, cannot have slow

Task 2: Cook a meal – needs pan, groceries, cannot have small

Task 3: Write a book – needs pen, paper, idea, cannot have confusion

*#author: Joel Navarrete  
#This program lets the user know if a task can be completed when all req. are there*items = [**"pan"**, **"paper"**, **"idea"**, **"rope"**, **"groceries"**]  
debuffs = [**"slow"**]  
  
def list\_check(list1, list2):  
 for i in list1:  
 if i in items:  
 pass  
 else:  
 return False  
  
 for i in list2:  
 if i in debuffs:  
 return False  
  
 return True  
  
def main():  
  
 task1\_items = [**"rope"**, **"coat"**, **"first aid kit"**]  
 task1\_debuffs = [**"slow"**]  
 result = list\_check(task1\_items, task1\_debuffs)  
 if result == True:  
 print(**"Task 1 can be performed"**)  
 else:  
 print(**"Task 1 can NOT be performed"**)  
  
 task2\_items = [**"pan"**, **"groceries"**]  
 task2\_debuffs = [**"small"**]  
 result = list\_check(task2\_items, task2\_debuffs)  
 if result == True:  
 print(**"Task 2 can be performed"**)  
 else:  
 print(**"Task 2 can NOT be performed"**)  
  
  
 task3\_items = [**"pen"**, **"paper"**, **"idea"**]  
 task3\_debuffs = [**"confusion"**]  
 result = list\_check(task3\_items, task3\_debuffs)  
 if result == True:  
 print(**"Task 3 can be performed"**)  
 else:  
 print(**"Task 3 can NOT be performed"**)  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()