Gordon Ng

Lab 5

# Part 1

1)

|  |
| --- |
| >>> a, b = True, False  >>> c = not b  >>> print(not a, not b, not c)  False True False  >>> print(a and b, a and c)  False True  >>> print(a or b, a or c)  True True |

2)

|  |
| --- |
| >>> print(a == True, b == True, c == True)  True False True  >>> print(a, b, c)  True False True  #Its essentially comparing if a,b and c are equivalent to true and then outputting if they are in TF. I would prefer not over that because of how much shorter and easier to read it is. |

3)

|  |
| --- |
| >>> print(a != True, b == False, c == False)  False True False  >>> print(not a, not b, not c)  False True False  #The first one is comparing if a does not equal to true which returns false, and b/c compare if they are equal to false. |

4)

|  |
| --- |
| >>> print(a or b and not c)  True  #It returns true as it compares not c which is False to b and not c which is False, then a or b and not c which is True or False which returns true.  >>> print((a or b) and not c)  False  #This isolates a or b which returns true but then compares is to and not c which returns false. True and False returns False. |

5)

|  |
| --- |
| >>> print(a or d)  True  >>> print(b and d)  False  >>> print(a and d)  Traceback (most recent call last):  File "<pyshell#13>", line 1, in <module>  print(a and d)  NameError: name 'd' is not defined  >>> print(b or d)  Traceback (most recent call last):  File "<pyshell#14>", line 1, in <module>  print(b or d)  NameError: name 'd' is not defined  #Since d is not representative of anything, having it go a or d makes it ignore d altogether since a is already true. The same thing goes for b and d since b is already false. However in a and d / b or d, since a and b aren’t the proper values to bypass looking over d, the code will check for d and then have a traceback since d itself is not defined. |

6)

|  |
| --- |
| >>> print(bool(list()), bool([[]]))  False True  >>> print(bool(1e-324), bool(1e-323))  False True  >>> print(bool(0), bool(1))  False True  >>>  # Boolean of empty list is nothing, Boolean of a list containing an empty list is true  # Boolean of a number =< 12-324 returns false, Boolean of a number => 1e-323 returns true  # Boolean of a zero integer returns false while anything above it is true |

7)

|  |
| --- |
| >>> print("apple" or "egg")  apple  >>> print((0,) and [9])  [9]  >>> print (0.0 or -1.0)  -1.0  >>> print (0.0 and -1.0)  0.0  #Since it only evaluates the overriding side, it will choose to print the left side and ignore the right one for apple  #It will print [9] since it will look at the other side since the left one returns true which can be overridden by a false on the right side so it also checks that side  #It prints -1.0 since 0.0 returns false and in an or expression, False can be overturned by true so it checks the right side  #Since 0.0 returns false, the and expression ignores the right side as False overrides that side regardless of its value |

# Part 2

1)

|  |
| --- |
| P or not p |

2)

|  |
| --- |
| P or (q and r) |

3)

|  |
| --- |
| Not (p and q) and p or q |

4)

|  |
| --- |
| Not (p or q) and not r |

# Could test in idle using predefined values with original and simplified forms

# Part 3

|  |
| --- |
| >>> x = [False]\*4  >>> y = (5, 2, 9, 21, -6)  >>> print(any(x), all(x))  False False  >>> x.append(True)  >>> print(any(x), all(x))  True False  >>> print(any(y), all(y))  True True  >>> y = y + (0,)  >>> print(any(y), all(y))  True False  #It implies since all(x) is true, there are no falses inside so nothing is ever overridden by false. That means everything within returns true meaning that any(x) should also return true, pointing that there is at least one true in the list. |

# Part 4

1)

|  |
| --- |
| year = int(input("Give a year: "))  if (year < 1582):  print("Invalid year")  elif (year >= 1582) and ((((year % 4) == 0) and ((year % 100) != 0)) or ((year % 400) == 0)):  print ("Leap year")  else:  print("Not a leap year") |

2)

|  |
| --- |
| password = input("Give a password:")  digitInPass = 0  stringInPass = 0  for character in password:  if character.isdigit():  digitInPass = 1  if character.isalpha():  stringInPass = 1  if len(password) >= 16 and (" " not in password):  print("Valid")  elif (len(password) >= 10) and (" " not in password) and (digitInPass != 0) and (stringInPass != 0) and (("$" in password) or ("\*" in password) or ("&" in password) or ("%" in password) or ("#" in password) or ("." in password)):  print("Valid")  else:  print("Invalid") |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |