Part 1

1. Evaluate the following expressions for `num1 = 10` and `num2 = 20`.

(a) `not (num1 < 1) and num2 < 10`

`True`

(b) `not (num1 < 1) and num2 < 10 or num1 + num3 < 100`

`false`

(c) `not (num2 > 1) or num1 > num2 - 10`

`True`

2. Give an appropriate if statement for each of the following
(the value of `num` is not important):

(a) Displays `'within range'` if `num` is between 0 and 100, inclusive.

```
def within_range():
    num1=int(input("enter a number:"))
    if num1 <=100:
        print("inclusive")
    else:
        print("non inclusive")
within_range()
```

~/Desktop/python/co.py

```
sers/macbookpro/Desktop/python/week9.py
enter a number:0
inclusive
macbookpro@ec2-100-22-136-78 python %
```

(b) Displays `'within range'` if `num` is between 0 and 100, inclusive, and displays `'out of range'` otherwise.

```python
def within_range():
    num1=int(input("enter a number:"))
    if num1 <=100:
        print("inclusive")
    else:

        print("out of range")
within_range()
```

```
enter a number:200
out of range
macbookpro@ec2-100-22-136-78 python % cd /Use
o/Desktop/python
/usr/bin/python3 /Users/macbookpro/Desktop/py
y
macbookpro@ec2-100-22-136-78 python % /usr/bi
sers/macbookpro/Desktop/python/week9.py
enter a number:99
inclusive
macbookpro@ec2-100-22-136-78 python %
```

3. Rewrite the following if-else statements using a single if statement and elif:

```
if temperature >= 85 and humidity > 60:
    print ('muggy day today')
else:
    if temperature >= 85:
        print ('warm, but not muggy today')
    else:
        if temperature >= 65:
            print ('pleasant today')
        else:
            if temperature <= 45:
                print ('cold today')
            else:
                print ('cool today')
```

```python
def temp():
    temperature=float(input("enter the tempetaure:"))
    humidity =(float)
    if temperature >= 85 and humidity > 60:
        print ('muggy day today')

    else:
        if temperature >= 85:
            print ('warm, but not muggy today')

        else:

            if temperature >= 65:
                print ('pleasant today')
            else:
                if temperature <= 45:
                    print ('cold today')
                else:
                    print ('cool today')
temp()
```

4. Write a Python program in which:

(a) The user enters either `'A', 'B', or 'C'`. If `'A'` is entered, the program should display the word `'Apple'`; if `'B'` is entered, it displays `'Banana'`; and if 'C' is entered, it displays `'Coconut'`. Use nested if statements for this.

```python
def showFruits():
        while True:
         userInput = input(" Enter 'A','B', or'C': ")
         userInput = userInput.lower()
         if(userInput == 'a'):
             print("Apple")
             break
         else:
             if (userInput == 'b'):
               print("Banana")
               break
             else:
               if(userInput == 'c'):
                print("Coconut")
                break
               else:
                continue

showFruits()
```

```
 Enter 'A','B', or'C': a
Apple
macbookpro@Avishek-macbook-pro python %
```

(b) Repeat question **(a)** using an if statement with `elif` headers instead.

```python
def showFruits():
        while True:
         userInput = input(" Enter 'A','B', or'C': ")
         userInput = userInput.lower()
         if(userInput == 'a'):
             print("Apple")
             break
         elif (userInput == 'b'):
              print("Banana")
              break
         elif (userInput == 'c'):
              print("Coconut")
              break

showFruits()
```

(c) A student enters the number of college credits earned. If the number of credits is greater than or equal to 90, 'Senior Status' is displayed; if greater than or equal to 60, 'Junior Status' is displayed; if greater than or equal to 30, 'Sophomore Status' is displayed; else, 'Freshman Status' is displayed.

```python
credit = int(input('Enter your current credit hours: '))
if (credit >= 90):
    print("Senior Status")
elif (credit >= 60):
    print("Junior Status")
elif (credit >= 30):
    print('Sophomore Status')
else:
    print('Freshman Status')
```

```
Enter your current credit hours: 89
Junior Status
PS G:\My Drive\Year One\Introduction to Programming\Week VI>
```

(e) The user enters a number. If the number is divisible by 3, the word ‘Fizz’ should be displayed; if the number is divisible by 5 the word ‘Buzz’ should be displayed and if the number is divisible by both ‘FizzBuzz’ should be displayed.

```python
num = int(input("Enter a number "))
if num % 3 == 0 and num % 5 == 0:
    print("FizzBuzz")
elif num % 3 == 0:
    print("Fizz")
elif num % 5 == 0:
    print("Buzz")
else:
    print("Neither Fizz, nor Buzz")
```

```
Enter a number 3
Fizz
PS G:\My Drive\Year One\Introduction to Programmin
```

5. Sam wants to store his series of car to a list. The list of a car are: (up to you). After creating a list he add some car and delete some car and at last there are still 5 cars left in his list. Additionally, he wants his car to be shuffled every time when the list is being displayed. [Hint: shuffle from random]

```python
import random
cars = ["Lamborgini", "Ferrari", "BMW", "Toyota", "Bentley"]
cars.append("Nissan")
cars.remove("Toyota")
random.shuffle(cars)
print(cars)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:
rive/Year One/Introduction to Programming/Week IX/Workshop/parti5.py"
['Ferrari', 'Bentley', 'Nissan', 'Lamborgini', 'BMW']
PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop>
```

Part 2

1. Write a program that:

partii1.py > ...

```python
sum = 0
for i in range(100, 201, 2):
    sum += i

print(sum)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS G:\My Drive\Year One\Introduction to Programming\Week
rive/Year One/Introduction to Programming/Week IX/Worksho
7650
```

(a) Uses a loop to add up all the even numbers between 100 and 200, inclusive.

partii1.py > ...

```python
sum = 0
for i in range(100, 201, 2):
    sum += i

print(sum)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS G:\My Drive\Year One\Introduction to Programming\Week
rive/Year One/Introduction to Programming/Week IX/Worksho
7650
```

(b) Sums a series of (positive) integers entered by the user, excluding all numbers that are greater than 100.

```
Partii2.py > ...
sum = 0
while True:
    num = int(input("Input number to sum(a negative number exits the loop): "))
    if num < 0:
        break
    if num > 100:
        break
    if num <= 100:
        sum += num
print(sum)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedbaldy/AppDa
rive/Year One/Introduction to Programming/Week IX/Workshop/Partii2.py"
Input number to sum(a negative number exits the loop): 10
Input number to sum(a negative number exits the loop): 12
Input number to sum(a negative number exits the loop): 19
Input number to sum(a negative number exits the loop): -7
41
```

(c) Solves Q2 but this time using an infinite loop, break and continue statements.

```
Partii2.py > ...
sum = 0
while True:
    num = int(input("Input number to sum(a negative number exits the loop): "))
    if num < 0:
        break
    if num > 100:
        continue
    sum += num
print(sum)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedbaldy/AppDat
rive/Year One/Introduction to Programming/Week IX/Workshop/Partii2.py"
Input number to sum(a negative number exits the loop): 10
Input number to sum(a negative number exits the loop): 20
Input number to sum(a negative number exits the loop): 30
Input number to sum(a negative number exits the loop): 1000
Input number to sum(a negative number exits the loop): -1
60
```

(d) Prompts the user to enter any number of positive and negative integer values, then displays the number of each type that were entered.

```
partII4.py > ...
num = int(input("Input a number: "))
if num < 0:
    print("Negative number")
elif num > 0:
    print("Positive number")
else:
    print("Invalid")
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\W
rive/Year One/Introduction to Programming/Week IX/Wor
Input a number: 101
Positive number
PS G:\My Drive\Year One\Introduction to Programming\W
```

2. The following while loop is meant to multiply a series of integers input by the user, until a sentinel value of 0 is entered. Indicate any errors in the code given. See if you can fix the program and get it running.

```
product = 1
num = input('Enter first number: ')
while num != 0:
    num = input('Enter first number: )
    product = product * num
  print('product = ', product)
```

```
error.py > ...
product = 1
num = int(input('Enter first number '))
while num != 0:
    product = product*num
    num = int(input('Enter first number '))
print('product= ', product)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week
rive/Year One/Introduction to Programming/Week IX/Worksho
Enter first number 10
Enter first number 20
Enter first number 0
product=  200
PS G:\My Drive\Year One\Introduction to Programming\Week
```

3. For each of the following, indicate which the definite loop is, and which an indefinite loop, explain your reasoning.

(a)

```
num = input('Enter a non-zero value:')
while num == 0:
    num = input('Enter a non-zero value: ')
```

(b)

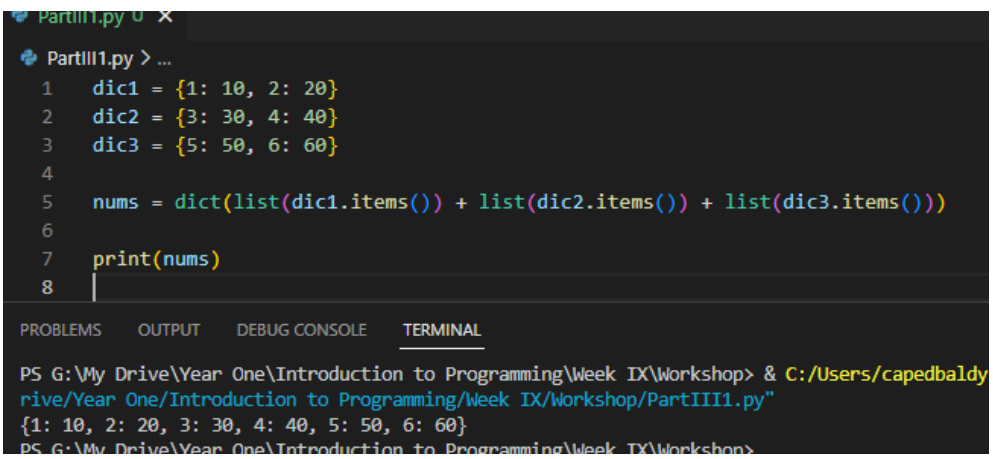
```
num = 0
while n < 10:
    print 2 ** n
    n = n + 1
```

Part 3

1. Create three dictionaries:

```
dic1 = {1:10, 2:20}
dic2 = {3:30, 4:40}
dic3 = {5:50, 6:60}
```

a. Write code to concatenate these dictionaries to create a new one. Create a variable called `nums` to store the resulting dictionary. There are multiple ways to do this, however, one of the easiest is to convert each of the dictionaries items to a list (which can be added together) and pass them to the `dict()` constructor.



b. Write code to add a new key/value pair to the dictionary `nums`: `(7, 70)`

```
PartIII1.py > ...
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

nums = dict(list(dic1.items()) + list(dic2.items()) + list(dic3.items()))
nums[7] = 70

print(nums)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedbalc
rive/Year One/Introduction to Programming/Week IX/Workshop/PartIII1.py"
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60, 7: 70}
```

c. Write code to update the value of the item with key `3` in `nums` to `80`

```
PartIII1.py U ×
PartIII1.py > ...
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

nums = dict(list(dic1.items()) + list(dic2.items()) + list(dic3.items()))
nums[7] = 70
nums[3] = 80

print(nums)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedbalc
rive/Year One/Introduction to Programming/Week IX/Workshop/PartIII1.py"
{1: 10, 2: 20, 3: 80, 4: 40, 5: 50, 6: 60, 7: 70}
```

d. Write code to remove the third item from dictionary `nums`.

```python
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

nums = dict(list(dic1.items()) + list(dic2.items()) + list(dic3.items()))
nums[7] = 70
nums[3] = 80
del nums[3]

print(nums)
```

```
PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedba
rive/Year One/Introduction to Programming/Week IX/Workshop/PartIII1.py"
{1: 10, 2: 20, 4: 40, 5: 50, 6: 60, 7: 70}
```

e. Write code to sum all the items in the dictionary `nums`

```python
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

nums = dict(list(dic1.items()) + list(dic2.items()) + list(dic3.items()))
nums[7] = 70
nums[3] = 80
del nums[3]
total = sum(nums.values())
print(total)
```

```
PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedbaldy
rive/Year One/Introduction to Programming/Week IX/Workshop/PartIII1.py"
250
```

f. Write code to multiply all the items in the dictionary `nums`

(g) Write code to retrieve the maximum and minimum values in `nums`.

```
PartIII1.py > ...
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

nums = dict(list(dic1.items()) + list(dic2.items()) + list(dic3.items()))
max_values = max(nums.values())
min_values = min(nums.values())
print("Maximum values ", max_values)
print("Minimum values ", min_values)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedba
rogramming/Week IX/Workshop/PartIII1.py"
Maximum values  60
Minimum values  10
```

3. Create a dictionary named `password_lookup` that contains usernames as keys and passwords as associated string values. Make up data for five entries.

```
PartIII3.py > ...
password_lookup = {'senha': 'Hello1234', 'gib': '12345678',
                   'indius': 'password123', 'luffy': 'meat0987', 'sunnena': 'dairy///'}
print(password_lookup.values())

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Workshop> & C:/Users/capedbaldy/AppData/Loca
rive/Year One/Introduction to Programming/Week IX/Workshop/PartIII3.py"
dict_values(['Hello1234', '12345678', 'password123', 'meat0987', 'dairy///'])
```

4. Write a program that creates an initially empty dictionary named `password_lookup`, prompting one-by-one for usernames and passwords

(until a username of 'z' is read) entering each into the dictionary.

```
PartIII4.py > ...
password_lookup = {}
while True:
    alpha = input("Enter a username('z' exits): ")
    if alpha == 'z':
        break
    password = input("Enter a password: ")
    password_lookup[alpha] = password

    print(password_lookup)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS G:\My Drive\Year One\Introduction to Programming\Week IX\Worksh
rive/Year One/Introduction to Programming/Week IX/Workshop/PartIII
Enter a username('z' exits): sujan shrestha
Enter a password: hello8e5
{'sujan shrestha': 'hello8e5'}
Enter a username('z' exits): sanima
Enter a password: sdgbjashd
{'sujan shrestha': 'hello8e5', 'sanima': 'sdgbjashd'}
Enter a username('z' exits):
```

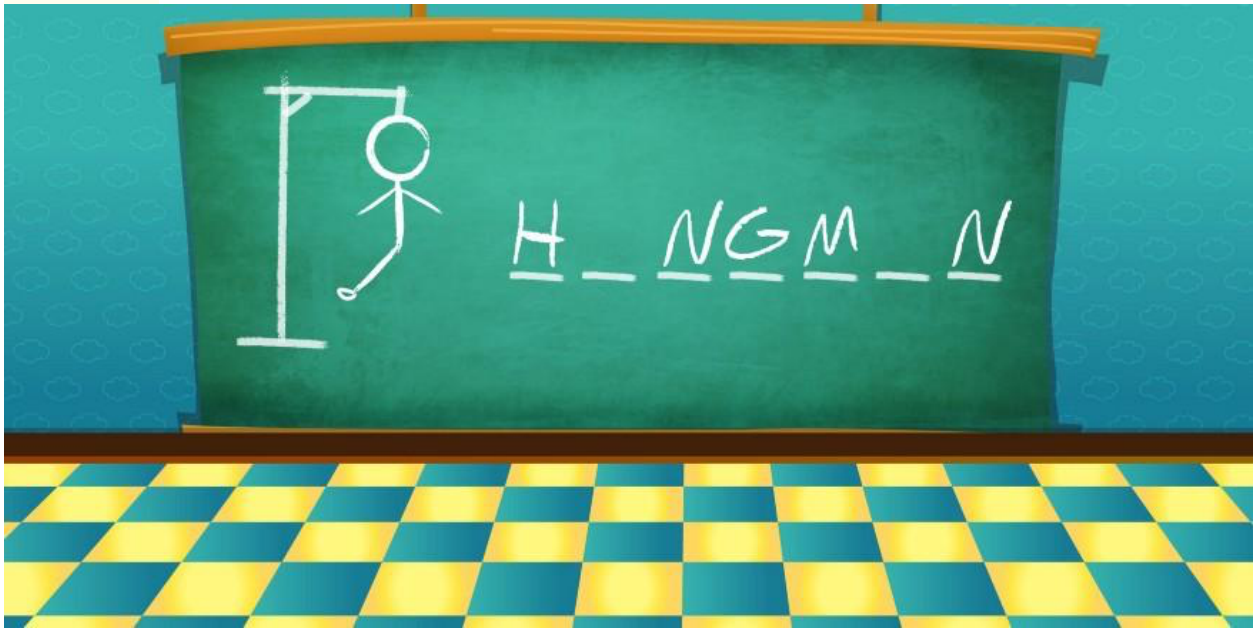
5. Create a dictionary named `password_hint` that contains email addresses as keys, and associated values that contain both the users' "password security question," and the answer to the question. Make up data for dictionary entries.

```
PartIII5.py > ...
password_hints = {'helo@gmail.com': {'security question': 'what is your fravoute food', 'answer': 'suitcase'},
                  'google@gmail.com': {'security question': 'what is you hobby', 'answer': 'sleeping'},
                  'govinda@gmail.com': {'security question': 'what is your fravoute movie', 'answer': 'govinda'},
                  'tututututututut@gmail.com': {'security question': 'what is your fravoute song', 'answer': 'tututut'},
                  'panipuri@gmail.com':{'security question':'what is your fravoute food','answer':'chatpate'}}
```

6. Create a dictionary named `member_table` that contains users' email addresses as keys, and answers to their password hints as the associated values, and a function that generates a temporary new password and stored in the table.

Part 4 (Home Task)

1. The hangman game introduces many new concepts like *methods*, which are functions attached to values. You'll also need to learn about a data type called a *list*. Once you understand these concepts, it will be much easier to program



1. You will need *random* module.
2. You will need to use the concept of *list.*

```python
import random
import time

# Initial Steps to invite in the game:
print("\nWelcome to Hangman game by IT SOURCECODE\n")
name = input("Enter your name: ")
print("Hello " + name + "! Best of Luck!")
time.sleep(2)
print("The game is about to start!\n Let's play Hangman!")
time.sleep(3)


# The parameters we require to execute the game:
def main():
    global count
    global display
    global word
    global already_guessed
    global length
    global play_game
    words_to_guess = ["january","border","image","film","promise","kids","lungs","doll","rhyme",
    "damage"
                    ,"plants"]
    word = random.choice(words_to_guess)
    length = len(word)
    count = 0
    display = '_' * length
    already_guessed = []
    play_game = ""
```

```python
# A loop to re-execute the game when the first round ends:

def play_loop():
    global play_game
    play_game = input("Do You want to play again? y = yes, n = no \n")
    while play_game not in ["y", "n","Y","N"]:
        play_game = input("Do You want to play again? y = yes, n = no \n")
    if play_game == "y":
        main()
    elif play_game == "n":
        print("Thanks For Playing! We expect you back again!")
        exit()

# Initializing all the conditions required for the game:
def hangman():
    global count
    global display
    global word
    global already_guessed
    global play_game
    limit = 5
    guess = input("This is the Hangman Word: " + display + " Enter your guess: \n")
    guess = guess.strip()
    if len(guess.strip()) == 0 or len(guess.strip()) >= 2 or guess <= "9":
        print("Invalid Input, Try a letter\n")
        hangman()
```

```
This is the Hangman Word: ______ Enter your guess:
j
   _____
  |
  |
  |
  |
  |
  |
__|__

Wrong guess. 4 guesses remaining

This is the Hangman Word: ______ Enter your guess:
```