Python Assignment #1

Name: Ritabroto Ganguly
Roll: 001910501090

1. Write a prime generator program using only primes and using python loops.

```
x = int(input("Enter value till which primes have to be generated: "))

for i in range(2,x):
 for j in range(2,i):
  if(i%j==0):
   break
 else:
  print(i)
```

```
Enter value till which primes have to be generated: 10
2
3
5
7
```

2. Write a discount coupon code using dictionary in Python with different rate coupons for each day of the week.

```
d = dict(mon=10,tue=15,wed=20,thu=15,fri=10,sat=5,sun=25)
s = input("Enter day of week: ")
t = s.strip().lower()
t = t[:3]
print('discount for '+s+ f' is {d[t]}')
```

```
Enter day of week: tuesday
discount for tuesday is 15
```

3. Print first 10 odd and even numbers using iterators and compress. You can use duck typing.

```
import itertools
even_selector = [True if x%2==0 else False for x in range(20)]
odd_selector = [False if x%2==0 else True for x in range(20)]
#print(even_selector)
#print(odd_selector)
evens = list(itertools.compress(range(20),even_selector))
odds = list(itertools.compress(range(20),odd_selector))
print(evens,odds,sep='\n')
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

4. Write a regular expression to validate a phone number.

```
import re

pattern = re.compile('\d{10}$')
number = input("Enter a phone number: ").strip()
rslt = pattern.match(number)
#print(rslt)
try:
 print(rslt.group(0))
except:
 print("No match")
```

```
Ritobrotos-MacBook-Air:python1 rgdgr8$ python3 4.py
Enter a phone number: 1234567890
1234567890
Ritobrotos-MacBook-Air:python1 rgdgr8$ python3 4.py
Enter a phone number: 12345
No match
Ritobrotos-MacBook-Air:python1 rgdgr8$ python3 4.py
Enter a phone number: asfs1313
No match
```

5. Write first seven Fibinacci numbers using generator next function/ yield in python. Trace and memorize the function.

```
mem = [0,1]

def fib_gen(fib=-1):
 a = 1
 b = 0
 yield b
 yield a
 i = 1
 while(i!=fib):#infinite sequence by default, since default value of fib is -1
  c = a
  a = a+b
  b = c
  mem.append(a) #to memorize the series
  i = i+1
  yield a

x = int(input("First how many fibonacci numbers do you want? "))
fibs = fib_gen(x-1)
try:
 while(True):
  (next(fibs))
except:
 print(mem)
 print(f"That's the first {x} fibonacci numbers")
```

```
First how many fibonacci numbers do you want? 7
[0, 1, 1, 2, 3, 5, 8]
That's the first 7 fibonacci numbers
```

8. Create a list of all the numbers up to N=50 which are multiples of five using anonymous function.

```
l = [x for x in range(51)]
l = list(filter((lambda x : x%5==0),l))
print(l)
```

```
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

10. Filter out the odd squares using map, filter, list.

```
from math import sqrt

l = [x for x in range(100)]
print(f"Original list is {l}")

t = list(map(lambda x: sqrt(x),filter(lambda y : True if y%2!=0 else False,l))) #square rooting the odd values from l
#print(f"Sqrt list is {t}")

def whole_number_check(x): #to check if the square root value is a whole number of not
 temp = int(x)
 if(x==temp):
    return True
 return False

t = list(map(lambda x: int(x*x),filter(whole_number_check,t))) #filtering only the odd squares whose square roots are whole numbers
print(f"\nList of odd squares is {t}")
```

```
Original list is [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72,
73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99]

List of odd squares is [1, 9, 25, 49, 81]
```

11. Let's find all Pythagorean triples whose short sides are numbers smaller than ten. Use filter and comprehension.

```
l = [[(y,x,(y*y + x*x)) for x in range(10)] for y in range(10)]
print(l)
```

```
[[(0, 0, 0), (0, 1, 1), (0, 2, 4), (0, 3, 9), (0, 4, 16), (0, 5, 25), (0, 6, 36), (0, 7, 49), (0, 8, 64), (0, 9, 81)],
[(1, 0, 1), (1, 1, 2), (1, 2, 5), (1, 3, 10), (1, 4, 17), (1, 5, 26), (1, 6, 37), (1, 7, 50), (1, 8, 65), (1, 9, 82)],
[(2, 0, 4), (2, 1, 5), (2, 2, 8), (2, 3, 13), (2, 4, 20), (2, 5, 29), (2, 6, 40), (2, 7, 53), (2, 8, 68), (2, 9, 85)],
[(3, 0, 9), (3, 1, 10), (3, 2, 13), (3, 3, 18), (3, 4, 25), (3, 5, 34), (3, 6, 45), (3, 7, 58), (3, 8, 73), (3, 9, 90)]
, [(4, 0, 16), (4, 1, 17), (4, 2, 20), (4, 3, 25), (4, 4, 32), (4, 5, 41), (4, 6, 52), (4, 7, 65), (4, 8, 80), (4, 9, 9
7)], [(5, 0, 25), (5, 1, 26), (5, 2, 29), (5, 3, 34), (5, 4, 41), (5, 5, 50), (5, 6, 61), (5, 7, 74), (5, 8, 89), (5, 9
, 106)], [(6, 0, 36), (6, 1, 37), (6, 2, 40), (6, 3, 45), (6, 4, 52), (6, 5, 61), (6, 6, 72), (6, 7, 85), (6, 8, 100),
(6, 9, 117)], [(7, 0, 49), (7, 1, 50), (7, 2, 53), (7, 3, 58), (7, 4, 65), (7, 5, 74), (7, 6, 85), (7, 7, 98), (7, 8, 1
13), (7, 9, 130)], [(8, 0, 64), (8, 1, 65), (8, 2, 68), (8, 3, 73), (8, 4, 80), (8, 5, 89), (8, 6, 100), (8, 7, 113), (
8, 8, 128), (8, 9, 145)], [(9, 0, 81), (9, 1, 82), (9, 2, 85), (9, 3, 90), (9, 4, 97), (9, 5, 106), (9, 6, 117), (9, 7,
 130), (9, 8, 145), (9, 9, 162)]]
```

12. Enumerate the sequence of all lowercase ASCII letters, starting from 1, using enumerate.

```
l = [chr(x+97) for x in range(26)]
l = list(enumerate(l,1))
print(l)
```

```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd'), (5, 'e'), (6, 'f'), (7, 'g'), (8, 'h'), (9, 'i')
, (10, 'j'), (11, 'k'), (12, 'l'), (13, 'm'), (14, 'n'), (15, 'o'), (16, 'p'), (17, 'q'),
 (18, 'r'), (19, 's'), (20, 't'), (21, 'u'), (22, 'v'), (23, 'w'), (24, 'x'), (25, 'y'),
(26, 'z')]
```

13. Write a code which yields all terms of the geometric progression a, aq, aq^2 , aq^3 , ... When the progression produces a term that is greater than 100,000, the generator stops (with a return statement). Compute total time and time within the loop.

```
import time
a,q = [int(x) for x in input("Value of a and q for a,aq,... GP: ").split()]
t1 = time.time()
while(a<100000):
 print(a)
 a = a*q
t2 = time.time()
print("That's the GP")
print(f'Loop time is {(t2-t1)}')
t3 = time.time()
print(f'Total time is {(t3-t1)}')
```

```
Value of a and q for a,aq,... GP: 3 4
3
12
48
192
768
3072
12288
49152
That's the GP
Loop time is 0.00021982192993164062
Total time is 0.0003597736358642578
```