```
#Program - 1st
#WAP that takes variable number of positional argument as i/p how to check if the arguments that are passed are more than 5
'''
def pos_arg(*args): #positional argument
    if len(args) > 5:
        return f'the pos arguments are {len(args)}'
    else:
        print("the arguments are less than 5")
print(pos_arg(1,2,3,4,5,6))
'''
#o/p -
#the pos arguments are 6


#*********************************************************************************************
#Program - 2nd
## WAF TO PRINT THE BELOW O/P
    #func('TRACXN',0)  # SHOULD PRINT RCN
    #func('TRACXN',1)  #SHOULD PRINT TAX
'''
def func(string, value):
    #Using slicing
    if value == 0:
        print(string[1::2])
    elif value == 1:
        print(string[::2])
func('TRACXN',0)
func('TRACXN',1)
```

```
'''
#O/p -
#RCN
#TAX
#Program - 1st
#WAP that takes variable number of positional argument as i/p how to check if the arguments that are passed are more than 5
'''
def pos_arg(*args): #positional argument
    if len(args) > 5:
        return f'the pos arguments are {len(args)}'
    else:
        print("the arguments are less than 5")
print(pos_arg(1,2,3,4,5,6))
'''
#o/p -
#the pos arguments are 6


#**********************************************************************************
***********
#Program - 2nd
## WAF TO PRINT THE BELOW O/P
    #func('TRACXN',0)  # SHOULD PRINT RCN
    #func('TRACXN',1)  #SHOULD PRINT TAX
'''
def func(string, value):
    #Using slicing
    if value == 0:
        print(string[1::2])
```

```
    elif value == 1:
        print(string[::2])
func('TRACXN',0)
func('TRACXN',1)
'''
#O/p -
#RCN
#TAX

#**********************************************************************************
***********

                                        #Assignments

#Prime numbers are natural numbers that are divisible by only 1 and the number itself. In other words,
#prime numbers are positive integers greater than 1 with exactly two factors,
#1 and the number itself. Some of the prime numbers include 2, 3, 5, 7, 11, 13, etc.

#Program - 3rd
#Waf to check if the no. is prime or n?

'''
def isprime(num):
    if num > 1:
        for n in range(2,num):
            if num %n == 0:
                return f'{num} is not prime number'
        return f'{num} is prime number'
```

```
print(isprime(5)) #5 is prime number
'''

#**********************************************************************************************
#Program - 4th
#Write a method that returns the last digit of an integer. for eg. the call of get_last_digit(3572) should return 2

'''
# Find the last digit
def get_last_digit(n) :
    return n % 10
res = get_last_digit(3572)
print(res)

'''

#**********************************************************************************************
#Program - 5th
#Make a function named tail that takes a sequence (like a list , string or tuple) and a number n and returns the last n elements from
#  the given sequence as a list

# The length of the tail
'''
def tail(sequence, n):
    return list(sequence[-n :])
print(tail('hello',2))
```

'''

#OR

'''

```
def tail(sequence, n):
   return (sequence [-n :])
print(tail([1,2,3,4,5], 3))
```

'''

#O/p

#[3, 4, 5]

#**********

#Program - 6th

#WAF named is_perfect_square that accepts a number and returns True if it's a perfect square and Fasle if its not

'''

```
def is_perfect_square(num):
   for i in range(0,num): # If you are not sure about the number then go and just mention num
      if i*i==num:
         return True
   else:
      return False
print(is_perfect_square(4))
```

'''

#**********

#Program - 7

```
#Write a function which returns the sum of lengths of all the iterables
'''
l = [1,2,3,4,5]
t = (4,5,6)

def sum_of_lengths(l1, t1):
    return len(l1) + len(t1)
print(sum_of_lengths(l, t)) #8
'''

#**********************************************************************************
******************
#Program - 8
#Python program to check whether a number is  prime or not
'''
def isprime(num):
    if num > 1:
        for n in range(2,num):
            if num %n == 0:
                return f'{num} is not prime number'
        return f'{num} is prime number'

print(isprime(5)) #5 is prime number

#Print 10th Fibonacci no.
n = 10
a, b = 0, 1
for i in range(n-1):
    c = a + b
    a, b = b, c
    print(a)
```

```
#O/p -
'''
1
1
2
3
5
8
13
21
34
'''

#Given n is fibonacci r not
def fibo(number):
    a = 0
    b = 1
    while a <= number:
        if a == number:
            return f'{number} is a fibonacci number'
        c = a + b
        a, b = b,c
    else:
        return f'{number} is not a fibonacci number'

print(fibo(13))
print(fibo(9))
#O/p -
'''
13 is a fibonacci number
9 is not a fibonacci number
'''

#1. "Building a dict of word and length pair

#words = "This is a bunch of words"

'''

words = "This is a bunch of words"

l = words.split()

d = {}

for word in l:

    d[word] = len(word)
```

```
print(d) #{'This': 4, 'is': 2, 'a': 1, 'bunch': 5, 'of': 2, 'words': 5}
'''
#Comprehension
#d1 = {word : len(word) for word in l}
#print(d1) #{'This': 4, 'is': 2, 'a': 1, 'bunch': 5, 'of': 2, 'words': 5}

#**********************************************************************************
**********
#2. Flipping keys and values of the dictionary using dict comprehension
'''
d = {'a': 1, 'b': 2, 'c': 3}

d1 ={}

for key, value in d.items():
   d1[value] = key
print(d1)
'''
#{1: 'a', 2: 'b', 3: 'c'}

#Dict Comprehension
#d2 = {value : key for key.value in d.items()}
#print(d2)

#**********************************************************************************
**********
#3. Counting the number of each character in a String
```

```
#Using Normal Program
#my_string = 'guido van rossum'
'''
my_string = 'guido van rossum'
d ={}

for ch in my_string:
   if ch in d:
      d[ch] += 1
   else:
      d[ch] = 1
print(d)
'''
#O/p - {'g': 1, 'u': 2, 'i': 1, 'd': 1, 'o': 2, ' ': 2, 'v': 1, 'a': 1, 'n': 1, 'r': 1, 's': 2, 'm': 1}

#Using Normal #Using Normal Program
#when you try to access or modify a key that's not present in the dictionary, a default value is automatically given to that key .
#dict
#Just we are initializing the value
#0 false ()
#range and xrange

'''
from collections import defaultdict

string = 'guido van rossum'
dd = defaultdict(int)
for ch in string:
```

```
    dd[ch] +=1
print(dd)
'''
#you cannot update the values and u cnt use the loop in Comprehension



#Dict Comprehension
#string = 'guido van rossum'
#d1 = {ch:string.count(ch) for ch in string}
#print(d1)

#**********************************************************************************
**********

#5. Dictionary of character and ascii value pairs
#s = 'abcABC'
#Normal:
'''
s = 'abcABC'

d = {}

for ch in s:
    d[ch] = ord(ch)
print(d)'''
#{'a': 97, 'b': 98, 'c': 99, 'A': 65, 'B': 66, 'C': 67}

#Dict Comprehension
#d1 = {ch : ord(ch) for ch in s}
```

```
#print(d1) #{'a': 97, 'b': 98, 'c': 99, 'A': 65, 'B': 66, 'C': 67}


#*************************************************************************************
***********
#6. Creating Dictionary of city and population pairs using Dict Comprehension


#Normal:
'''
cities = ['Tokyo',
        'Delhi',
        'Shanghai',
        'Sao Paulo',
        'Mumbai'
        ]

population = ['38,001,000',
            '25,703,168',
            '23,740,778',
            '21,066,245',
            '21,042,538'
            ]
'''


#zip() function returns



#Dict Comprehension
#d = {cities : population for cities, population in zip(cities, population)}
#print(d)
```

```
#{'Tokyo': '38,001,000', 'Delhi': '25,703,168', 'Shanghai': '23,740,778', 'Sao Paulo': '21,066,245',
'Mumbai': '21,042,538'}


#*************************************************************************************
***********

#7. Create a dictionary of dialcode and country from the following list

'''

dial_codes = [

   (86, 'China'),

   (91, 'India'),

   (1, 'United States'),

   (62, 'Indonesia'),

   (55, 'Brazil'),

   (92, 'Pakistan'),

   (880, 'Bangladesh'),

   (234, 'Nigeria'),

   (7, 'Russia'),

   (81, 'Japan')

   ]

d = {}

for item in dial_codes: #unpack it

   key,value = item

   d[key] = value

print(d)

'''

#{86: 'China', 91: 'India', 1: 'United States', 62: 'Indonesia', 55: 'Brazil', 92: 'Pakistan', 880: 'Bangladesh',
234: 'Nigeria', 7: 'Russia', 81: 'Japan'}

#*************************************************************************************
***********
```

#8. Left Triangle number Pattern

'''

   1

  12

 123

1234

12345

'''

#Mam

'''

pat = ""

for row in range(1,6):

    pat = pat + str(row)

    print(f"{pat:>5}")

'''

#Qns by mam - print

#O/p -

"""

   1

  12

 123

1234

12345

"""

#**********
****

#10.

```
'''
*
* *
*
* * *
*
* * * *
*
* * * * *
'''

for row in range(2,6):
    print("*")
    print("* "*row)

'''
*
* *
*
* * *
*
* * * *
*
* * * * *

'''
```

```
#*********************************************************************************
**********

#11.

'''

1 2 3 *

1 2 * 4

1 * 3 4

* 2 3 4

'''

#1st way:

i = 4

for row in range(1,5):

   pat =""

   for j in range(1,5): #j=1,2,4

      if j == i:

         pat =pat + ' ' + '*'

         i -= 1

      else:

         pat = pat + " " +str(j)

   print(pat)


'''

1 2 3 *

1 2 * 4

1 * 3 4

* 2 3 4

'''

#2nd way:

for i in range(1,5):
```

```
    for j in range(1,5):
        if i + j == 5:
            print("*", end =" ")
        else:
            print(j, end =" ")
    print()


'''
1 2 3 *
1 2 * 4
1 * 3 4
* 2 3 4
'''
#*************************************************************************************
***********
```