WEEK 3: Assignment

In [ ]:

```
Q1.  Calculate the Salary using Python Programming
Read the question carefully and follow the input and output format.
Karen got salary for this month and she spends 20% of her salary for food,
10% of her salary for entertainment and 10% of her salary for travel.
If she takes care of other shifts she will get 4% of the salary per day.
Given her salary and the number of shifts she handled.Calculate how much she can save in her pocket after spending all these?
Input and Output Format :
First line of input consists of an integer, salary. Next line correspond to the number of shifts.
Output consist of an integer, which is saving.
1. Print "Salary too large" when salary is greater than 10000.
2. Print "Shifts too small" when the shift is less than 0.
3. Print "Salary too small" when the salary is less than 0.
Include a function named calculateSal(salary,shifts) to calcuate salary and print saving
```

In [19]:

```
def calculate_salary(salary,shifts):
    if salary>10000:
        print("Salary too large")
    elif salary<0:
        print("Salary too small")
    elif shifts<0:
        print("Shifts too small")
    else:
         total_salary=salary+((salary*0.04)*shifts)
         food_spends=salary*0.2
         entertainment_spends=salary*0.1
         travel_spends=salary*0.1
         total_spends=food_spends+travel_spends+entertainment_spends
         total_savings=total_salary-total_spends
         return int(total_savings)
salary=int(input("Salary : "))
shifts=int(input("Shifts : "))
calculate_salary(salary,shifts)
```

```
Salary : 6000
Shifts : 3
```

Out[19]:

```
4320
```

In [ ]:

```
Q2.  write python code to read n as integer,from STDIN. For all non-negative integers i<n ,print i**2.
Example
The list of non-negative integers that are less than n=3 is[0,1,2]. Print the square of each number on a separate line.
014

Input Format
The first and only line contains the integer,n .

Output Format
Print n lines, one corresponding to each i .
Sample Input
5
Sample Output
0149
16
```

In [22]:

```
n=int(input())
if n>=0:
 for i in range(n):
    print(i**2 ,end="\t")
```

```
5
0       1       4       9       16
```

In [24]:

```
Q3.  An extra day is added to the calendar almost every four years as February 29,
and the day is called a leap day. It corrects the calendar for the fact that our planet
takes approximately 365.25 days to orbit the sun. A leap year contains a leap day.

In the Gregorian calendar, three conditions are used to identify leap years:
The year can be evenly divided by 4, is a leap year, unless:
The year can be evenly divided by 100, it is NOT a leap year, unless:
The year is also evenly divisible by 400. Then it is a leap year.
This means that in the Gregorian calendar, the years 2000 and 2400 are leap years,
while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. Source

Task
Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean True,
otherwise return False.
Note that the code stub provided reads from STDIN and passes arguments to the is_leap function.
It is only necessary to complete the is_leap function.

Input Format
Read , the year as integer to test.

Output Format
The function must return a Boolean value (True/False)

Sample Input
1990
Sample Output
False
```

```
  Input In [24]
    Q3.Anextra day is added to the calendar almost every four years as February 29,
               ^
SyntaxError: invalid syntax
```

In [28]:

```
def leap(y):
 if y%4==0 or (y%100==0 and y%400==0):
    return True
 else:
    return False
year=int(input("Year : "))
leap(year)
```

```
Year : 1996
```

Out[28]:

```
True
```

In [ ]:

In [ ]: