ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mentioning the names of those people who helped to make it possible. I owe my due reverence to the CBSE board for including practical work as a part of our curriculum and to my school, Shree Sarasswathi Vidhyaah Mandheer for giving us the opportunity and resources to complete this project effectively.

I take this opportunity to express my gratitude and respect in a few words to all those who helped me in the completion of this project. I would like to express my special thanks to our Correspondent Madam, Principal Mrs. Shantha Kumari and the teachers of Computer Science department Mrs.Surya and Mrs. Suganya who gave me the required help and support to do this project on this topic.

This project helped me in doing a lot of research and learning new things. Secondly I would like to thank CBSE for giving me such an opportunity without which I would not have been able to do such a project. I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame. Above all, I thank God for showering his plentiful blessing in keeping us safe all year through and enabling us to work without any obstacles.

PROJECT SYNOPSIS

About The Project

Result analyser is a project to analyse all the results of grade 10 and 12 in a simple and efficient manner. The results could be viewed in the form of an excel sheet where the marks are arranged in an order of highest to the lowest mark. The marksheet could be downloaded from the website .The user also has an option for viewing the subject toppers. Overall, result analyser could prove as an outstanding replacement for manually entering the marks in an Excel sheet.

CONTENTS

INTRODUCTION

The software project,"**Result Analyser**" is developed to view grade 10 and 12 marks.The purpose of the project is to make the analysis of the results easier.It also provides subject-wise toppers for each subjects.
The project also mainly focuses on the use of GUI[Graphical User Interface] .It can prove as an ultimate boost for an interactive analyser.
The software is user-friendly and does not require any training to be used.The project is coded using and designed using python IDE.

REQUIREMENT ANALYSIS

Problem:

When the results of the board exams are out every year,teachers face difficulties in entering marks manually in an excel sheet.In those cases ,the result analyser would be helpful by automating the process.

Advantages:

The project is user friendly and does not need any training to be used.The automated process acts as an icing to the cake as it saves a lot of time.The purpose is to create a software which can be used by all the staffs without much difficulty.

Project description:

- There is only 1 for this software.Just the user.
- The software can be accessed by anyone.
- The user selects the desired option which they want to for seeing the result they want.

SYSTEM ANALYSIS AND DESIGN

1.4 THEORETICAL BACKGROUND

WHAT IS PYTHON?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures,combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

WHAT IS PYTHON IDLE?

Every Python installation comes with a Integrated Development and Learning Environment, which you'll see shortened to IDLE or even IDE. These are a class of applications that help you write code more efficiently. While there are many IDE's for you to choose from, Python IDLE is very bare-bones, which makes it the perfect tool for a beginning programmer. Python IDLE as an interactive interpreter or as a file editor. The Python shell is an excellent place to experiment with small code snippets. You can access it through the terminal or command line app on your machine. You can simplify your workflow with Python IDLE, which will immediately start a Python shell when you open it. Every programmer needs to be able to edit and save text files. Python programs are files with the .py extension that contain lines of Python code.

Python IDLE gives you the ability to create and edit these files with ease. Python IDLE also provides several useful features that you'll see in professional IDLEs, like basic syntax highlighting, code completion, and auto-indentation.

WHAT IS HTML?

HTML (Hypertext Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behaviour (JavaScript)."Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web.By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.

HTML uses "markup" to annotate text,
images, and other content for display
in a Web browser. An HTML element is set
off from other text in a document by "tags", which
consist of the element name
surrounded by "<" and ">". The name of an element
inside a tag is case insensitive. That is, it can be
written in uppercase, lowercase, or a mixture

What is Flask?

Flask is a small and lightweight Python web framework that provides useful tools and features that helps in creating web applications in Python easier.It gives developers flexibility and it is more accessible framework for new developers since you can build a web application quickly using only a single Python file

1.6. SYSTEM REQUIREMENTS

Hardware requirements:

- A computer/laptop with operating system- window 7 or above
- X86 64-bit CPU (Intel/AMD architecture)
- 4 GB RAM
- 5 GB free disk space.

Software Requirements:

- Python 3.9 or high,
- Vitualenv pre-installed
- Flask library

Hardware used:

- Processor: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 2.00GHz
- RAM: 8.00 GB
- System type: 64-bit Operating System
- x64Based processor
- Windows edition: Windows 10 pro

Software's used:

- Microsoft Windows 10 as Operating System.
- Python 3.7 as Front-end for GUI development.
- MS-Word 2010 for documentation.

SOURCE CODE

Excel_Converter.py

```
def convertG10(file,opt=''):
    global ds, bd, x1, g
    f = open(file)
    if opt=='':
        g = open(file[:-4]+'.csv', 'w+', newline='')
    else:
        ds = open(file[:-4] + '(DS).csv', 'w+', newline='')
        bd = open(file[:-4] + '(BD).csv', 'w+', newline='')
        lst = open(opt)
        x1 = lst.readlines()
    x = f.readlines()

    c_name = ['ROLL_NO', 'NAME', 'ENG', 'TAMIL',
'HINDI', 'FRENCH', 'MATHS', 'SCIENCE',
'SOCIAL', 'TOTAL', 'RESULT']
    lst = []
    l2 = []
    p_f = []
    totalmark = []
    avg=[]
    final = []
```

```
for i in range(len(x)):
    y = x[i].split()
    if y == []:
        pass
    else:
        roll = ''
        name = ''
        total = 0
        marks_lst = []
        pass_fail = []
        total_lst = []

        for j in y:
            name_rollno = [roll, name[1:]]
            if y[0].isdigit() and j.isalpha() and j !=
('PASS' or 'FAIL'):
                name = name + j + ' '
            elif y[0].isdigit() and j.isdigit() and len(j) >
4:
                roll += j

        if len(name_rollno[0]) > 1 and
len(name_rollno[1]) > 1:
```

```
lst.append(name_rollno)

        if y[0].isdigit() and len(y[0])>4:
            sub_codes = []
            for code in y:
                if code.isdigit() and len(code)==3:
                    sub_codes.append(code)

            if sub_codes[1]=='006':
                y1=x[i + 1].split()
                for k in range(len(y1)):
                    if k%2==0:
                        marks_lst.append(y1[k])
                marks_lst.insert(2,'-')
                marks_lst.insert(3,'-')
            elif sub_codes[1]=='085':
                y1 = x[i + 1].split()
                for k in range(len(y1)):
                    if k % 2 == 0:
```

```
marks_lst.append(y1[k])
                marks_lst.insert(1, '-')
                marks_lst.insert(3, '-')
            elif sub_codes[1]=='018':
                y1 = x[i + 1].split()
                for k in range(len(y1)):
                    if k % 2 == 0:
                        marks_lst.append(y1[k])
                marks_lst.insert(1, '-')
                marks_lst.insert(2, '-')

        if marks_lst == []:
            pass
        else:
            l2.append(marks_lst)



        for p in y:
            if y[0].isdigit() and p.isalpha() and (p ==
'PASS' or p == 'FAIL'):
                pass_fail.append(p)

            if pass_fail == []:
```

```
            pass
        else:
            p_f.append(pass_fail)

    for tot in marks_lst:
        if tot!='-':
            total += int(tot)
    total_lst.append(str(total))

    if total_lst != ['0']:
        totalmark.append(total_lst)

for j in range(0, len(lst)):
    lk = lst[j] + l2[j] + totalmark[j] +
p_f[j]
    final.append(lk)

for s in range(len(final)):
    for total_lst in range(0, len(final) -
s - 1):
        if final[total_lst][-2] <
final[total_lst + 1][-2]:
```

```
final[total_lst], final[total_lst + 1] = final[total_lst +
1], final[total_lst]

  if opt==''':

    eng_no, eng_avg, tamil_no, tamil_avg, hindi_no,
hindi_avg, french_no, french_avg, math_no,
math_avg, sci_no, sci_avg, soc_no, soc_avg = 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    for x_1 in range(len(final)):
      if final[x_1][2] != '-':
        eng_avg += int(final[x_1][2])
        eng_no += 1
      if final[x_1][3] != '-':
        tamil_avg += int(final[x_1][3])
        tamil_no += 1
      if final[x_1][4] != '-':
        hindi_avg += int(final[x_1][4])
        hindi_no += 1
      if final[x_1][5] != '-':
        french_avg += int(final[x_1][5])
        french_no += 1
```

```
if final[x_1][6] != '-':
        math_avg += int(final[x_1][6])
        math_no += 1
    if final[x_1][7] != '-':
        sci_avg += int(final[x_1][7])
        sci_no += 1
    if final[x_1][8] != '-':
        soc_avg += int(final[x_1][8])
        soc_no += 1

avg_1 = round(eng_avg / eng_no, 2)
if tamil_no != 0:
    avg_2 = round(tamil_avg / tamil_no, 2)
else:
    avg_2 = 0
if hindi_no != 0:
    avg_3 = round(hindi_avg / hindi_no, 2)
else:
    avg_3 = 0
if french_no != 0:
    avg_4 = round(french_avg / french_no, 2)
```

```
  else:
        avg_4 = 0
      avg_5 = round(math_avg / math_no, 2)
      avg_6 = round(sci_avg / sci_no, 2)
      avg_7 = round(soc_avg / soc_no, 2)

      avg.extend(['', 'Average:', avg_1, avg_2, avg_3,
avg_4, avg_5, avg_6, avg_7])

      w = csv.writer(g, delimiter=',')
      w.writerow(c_name)
      w.writerows(final)
      w.writerow(avg)
      g.close()

   else:
      name_lst = []
      for i in x1:
         name_lst.append(i.strip())

      ds_lst=[]
      bd_lst=[]
```

```
for i in final:
        if i[0] not in name_lst:
            ds_lst.append(i)
        elif i[0] in name_lst:
            bd_lst.append(i)
    print(ds_lst)
    print(len(ds_lst))
    print(bd_lst)
    print(len(bd_lst))

    def option(var,f_ext):
        eng_no, eng_avg, tamil_no, tamil_avg,
hindi_no, hindi_avg, french_no, french_avg,
math_no, math_avg, sci_no, sci_avg, soc_no,
soc_avg = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        for x_1 in range(len(var)):
            if var[x_1][2] != '-':
                eng_avg += int(var[x_1][2])
                eng_no += 1
            if var[x_1][3] != '-':
                tamil_avg += int(var[x_1][3])
                tamil_no += 1
```

```
if var[x_1][4] != '-':
            hindi_avg += int(var[x_1][4])
            hindi_no += 1
        if var[x_1][5] != '-':
            french_avg += int(var[x_1][5])
            french_no += 1
        if var[x_1][6] != '-':
            math_avg += int(var[x_1][6])
            math_no += 1
        if var[x_1][7] != '-':
            sci_avg += int(var[x_1][7])
            sci_no += 1
        if var[x_1][8] != '-':
            soc_avg += int(var[x_1][8])
            soc_no += 1

    avg_1 = round(eng_avg / eng_no, 2)
    if tamil_no != 0:
        avg_2 = round(tamil_avg / tamil_no, 2)
    else:
        avg_2 = 0
```

```
if hindi_no != 0:
        avg_3 = round(hindi_avg / hindi_no, 2)
    else:
        avg_3 = 0
    if french_no != 0:
        avg_4 = round(french_avg / french_no, 2)
    else:
        avg_4 = 0
    avg_5 = round(math_avg / math_no, 2)
    avg_6 = round(sci_avg / sci_no, 2)
    avg_7 = round(soc_avg / soc_no, 2)

    avg.extend(['', 'Average:', avg_1, avg_2, avg_3,
avg_4, avg_5, avg_6, avg_7])

    w = csv.writer(f_ext, delimiter=',')
    w.writerow(c_name)
    w.writerows(var)
    w.writerow(avg)
    avg.clear()
    f_ext.close()

  option(ds_lst,ds);option(bd_lst,bd)
```

```
def convertG12(file,opt=''):
    global ds, bd, x1, g
    f = open(file)
    if opt=='':
        g = open(file[:-4]+'.csv', 'w+', newline='')
    else:
        ds = open(file[:-4] + '(DS).csv', 'w+', newline='')
        bd = open(file[:-4] + '(BD).csv', 'w+', newline='')
        lst = open(opt)
        x1 = lst.readlines()
    x = f.readlines()
    c_name = ['ROLL_NO', 'NAME', 'ENG', 'MATHS',
'PHY', 'CHE', 'C.S', 'APP.MATHS', 'BIO', 'ECO',
'ACC.', 'B.S', 'TOTAL', 'CUT-OFF', 'RESULT']
    lst = []
    l2 = []
    p_f = []
    totalmark = []
    cutoff = []
    avg = []
    final = []
```

```
for i in range(len(x)):
    y = x[i].split()
    if y == []:
        pass
    else:
        roll = ''
        name = ''
        eng, maths, phy, che, cs, ap_mat,
bio, eco, acc, bs = '-', '-', '-', '-', '-', '-', '-', '-', '-', '-'
        total = 0
        marks_lst = []
        pass_fail = []
        total_lst = []
        c_o=[]

        for j in y:
            name_rollno = [roll, name[1:]]
            if y[0].isdigit() and j.isalpha() and j != ('PASS' or
'FAIL'):
                name += j + ' '
            elif y[0].isdigit() and j.isdigit() and len(j) > 4:
                roll += j
```

```
if len(name_rollno[0]) > 1 and len(name_rollno[1]) >
1:
            lst.append(name_rollno)

        if y[0].isdigit() and len(y[0]) > 4:
            sub_codes = []
            marks_lst =
[eng,maths,phy,che,cs,ap_mat,bio,eco,acc,bs]

            for code in y:
                if code.isdigit() and len(code) == 3:
                    sub_codes.append(code)

            y_1 = x[i+1].split()
            for pos in range(len(y_1)):
                try:
                    if y_1[pos].isdigit()==False:
                        y_1.remove(y_1[pos])
                except:
                    break
```

```
for m in range(len(sub_codes)):
        if sub_codes[m]=='301':
          marks_lst[0]=y_1[m]
        elif sub_codes[m]=='041':
          marks_lst[1]=y_1[m]
        elif sub_codes[m]=='042':
          marks_lst[2]=y_1[m]
        elif sub_codes[m]=='043':
          marks_lst[3]=y_1[m]
        elif sub_codes[m]=='083':
          marks_lst[4]=y_1[m]
        elif sub_codes[m]=='241':
          marks_lst[5]=y_1[m]
        elif sub_codes[m] == '044':
          marks_lst[6]=y_1[m]
        elif sub_codes[m] == '030':
          marks_lst[7]=y_1[m]
        elif sub_codes[m] == '055':
          marks_lst[8]=y_1[m]
        elif sub_codes[m] == '054':
          marks_lst[9]=y_1[m]
        else:
          print('New sub_code found!',y[m])
```

```
        if marks_lst == []:
            pass
        else:
            l2.append(marks_lst)


        for p in y:
            if y[0].isdigit() and p.isalpha() and (p == 'PASS' or
p == 'FAIL'):
                pass_fail.append(p)

        if pass_fail == []:
            pass
        else:
            p_f.append(pass_fail)

        for tot in marks_lst:
            if tot!='-':
                total += int(tot)
        total_lst.append(str(total))

        if total_lst != ['0']:
            totalmark.append(total_lst)

        if marks_lst == []:
            pass
```

```
   elif marks_lst[1]!='-' and marks_lst[2]!='-' and
marks_lst[3]!='-':
            cut_off = int(marks_lst[1])+
round(int(marks_lst[2])/2) + round(int(marks_lst[3])/2)
            c_o.append(str(cut_off))
        else:
            c_o.append('-')

        if c_o == []:
            pass
        else:
          cutoff.append(c_o)

  for j in range(0, len(lst)):
     lk = lst[j] + l2[j] + totalmark[j] + cutoff[j] + p_f[j]
     final.append(lk)

  for s in range(len(final)):
     for total_lst in range(0, len(final) - s - 1):
        if final[total_lst][-3] < final[total_lst + 1][-3]:
           final[total_lst], final[total_lst + 1] =
final[total_lst + 1], final[total_lst]
```

```
if opt==':

    eng_no, eng_avg, maths_no, maths_avg, phy_no,
phy_avg, che_no, che_avg, cs_no, cs_avg, app_mat_no,
app_mat_avg, bio_no, bio_avg, eco_no, eco_avg
, acc_no, acc_avg, bs_no,
bs_avg = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    for x_1 in range(len(final)):
        if final[x_1][2] != '-':
            eng_avg += int(final[x_1][2])
            eng_no += 1
        if final[x_1][3] != '-':
            maths_avg += int(final[x_1][3])
            maths_no += 1
        if final[x_1][4] != '-':
            phy_avg += int(final[x_1][4])
            phy_no += 1
        if final[x_1][5] != '-':
            che_avg += int(final[x_1][5])
            che_no += 1
        if final[x_1][6] != '-':
            cs_avg += int(final[x_1][6])
            cs_no += 1
        if final[x_1][7] != '-':
            app_mat_avg += int(final[x_1][7])
            app_mat_no += 1
        if final[x_1][8] != '-':
            bio_avg += int(final[x_1][8])
            bio_no += 1
```

```
if final[x_1][9] != '-':
        eco_avg += int(final[x_1][9])
        eco_no += 1
    if final[x_1][10] != '-':
        acc_avg += int(final[x_1][10])
        acc_no += 1
    if final[x_1][11] != '-':
        bs_avg += int(final[x_1][11])
        bs_no += 1

  if eng_no != 0:
    avg_1 = round(eng_avg / eng_no, 2)
  else:
    avg_1 = 0
  if maths_no != 0:
    avg_2 = round(maths_avg / maths_no, 2)
  else:
    avg_2 = 0
  if phy_no != 0:
    avg_3 = round(phy_avg / phy_no, 2)
  else:
    avg_3 = 0
  if che_no != 0:
    avg_4 = round(che_avg / che_no, 2)
  else:
    avg_4 = 0
  if che_no != 0:
    avg_5 = round(cs_avg / cs_no, 2)
  else:
    avg_5 = 0
```

```
if app_mat_no != 0:
        avg_6 = round(app_mat_avg / app_mat_no, 2)
    else:
        avg_6 = 0
    if bio_no != 0:
        avg_7 = round(bio_avg / bio_no, 2)
    else:
        avg_7 = 0
    if eco_no != 0:
        avg_8 = round(eco_avg / bio_no, 2)
    else:
        avg_8 = 0
    if acc_no != 0:
        avg_9 = round(acc_avg / bio_no, 2)
    else:
        avg_9 = 0
    if bs_no != 0:
        avg_10 = round(bs_avg / bio_no, 2)
    else:
        avg_10 = 0
    avg.extend(['', 'Average:', avg_1, avg_2, avg_3, avg_4,
avg_5, avg_6, avg_7, avg_8, avg_9, avg_10])

    w = csv.writer(g, delimiter=',')
    w.writerow(c_name)
    w.writerows(final)
    w.writerow(avg)
    g.close()
```

```
else:
    name_lst = []
    for i in x1:
      name_lst.append(i.strip())
    print(name_lst)
    print(final)
    ds_lst = []
    bd_lst = []
    for i in final:
      if i[0] not in name_lst:
        ds_lst.append(i)
        print(i)
      elif i[0] in name_lst:
        bd_lst.append(i)
        print(i)

    print(ds_lst)
    print(bd_lst)

    def option(var, f_ext):
      eng_no, eng_avg, maths_no, maths_avg, phy_no,
 phy_avg, che_no, che_avg, cs_no, cs_avg, app_mat_no,
app_mat_avg, bio_no, bio_avg, eco_no, eco_avg
, acc_no, acc_avg, bs_no, bs_avg = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```
for x_1 in range(len(var)):
        if var[x_1][2] != '-':
            eng_avg += int(var[x_1][2])
            eng_no += 1
        if var[x_1][3] != '-':
            maths_avg += int(var[x_1][3])
            maths_no += 1
        if var[x_1][4] != '-':
            phy_avg += int(var[x_1][4])
            phy_no += 1
        if var[x_1][5] != '-':
            che_avg += int(var[x_1][5])
            che_no += 1
        if var[x_1][6] != '-':
            cs_avg += int(var[x_1][6])
            cs_no += 1
        if var[x_1][7] != '-':
            app_mat_avg += int(var[x_1][7])
            app_mat_no += 1
        if var[x_1][8] != '-':
            bio_avg += int(var[x_1][8])
            bio_no += 1
        if var[x_1][9] != '-':
            eco_avg += int(var[x_1][9])
            eco_no += 1
```

```
if var[x_1][10] != '-':
            acc_avg += int(var[x_1][10])
            acc_no += 1
        if var[x_1][11] != '-':
            bs_avg += int(var[x_1][11])
            bs_no += 1

    if eng_no != 0:
        avg_1 = round(eng_avg / eng_no, 2)
    else:
        avg_1 = 0
    if maths_no != 0:
        avg_2 = round(maths_avg / maths_no, 2)
    else:
        avg_2 = 0
    if phy_no != 0:
        avg_3 = round(phy_avg / phy_no, 2)
    else:
        avg_3 = 0
    if che_no != 0:
        avg_4 = round(che_avg / che_no, 2)
    else:
        avg_4 = 0
    if che_no != 0:
        avg_5 = round(cs_avg / cs_no, 2)
    else:
        avg_5 = 0
```

```
    avg_6 = round(app_mat_avg / app_mat_no, 2)
    else:
        avg_6 = 0
    if bio_no != 0:
        avg_7 = round(bio_avg / bio_no, 2)
    else:
        avg_7 = 0
    if eco_no != 0:
        avg_8 = round(eco_avg / bio_no, 2)
    else:
        avg_8 = 0
    if acc_no != 0:
        avg_9 = round(acc_avg / bio_no, 2)
    else:
        avg_9 = 0
    if bs_no != 0:
        avg_10 = round(bs_avg / bio_no, 2)
    else:
        avg_10 = 0
    avg.extend(['', 'Average:', avg_1,
avg_2, avg_3, avg_4, avg_5, avg_6, avg_7, avg_8,
avg_9, avg_10])
```

```
w = csv.writer(f_ext, delimiter=',')
        w.writerow(c_name)
        w.writerows(var)
        w.writerow(avg)
        avg.clear()
        f_ext.close()

    option(ds_lst, ds);option(bd_lst, bd)
```

Docx_Converter.py

```
import docx

def sub_top_g10(file,opt=''):
    global ds, bd, x1, g ,doc1, doc
    f = open(file)
    if opt == '':
        doc = docx.Document()
    else:
        namelist = open(opt)
        doc = docx.Document()
        doc1 = docx.Document()
        x1 = namelist.readlines()
    x = f.readlines()
    lst = []
    l2 = []
    final = []
```

```
def sorting(sub_name):
    for s in range(len(sub_name)):
        for total_lst in range(0, len(sub_name) - s - 1):
            if sub_name[total_lst][-1] <
sub_name[total_lst + 1][-1]:
                sub_name[total_lst], sub_name[total_lst +
1] = sub_name[total_lst + 1], sub_name[total_lst]

for i in range(len(x)):
    y = x[i].split()
    if y == []:
        pass
    else:
        roll = ''
        name = ''
        marks_lst = []

        for j in y:
            name_rollno = [roll, name[1:]]
            if y[0].isdigit() and j.isalpha() and j != ('PASS'
or 'FAIL'):
                name = name + j + ' '
            elif y[0].isdigit() and j.isdigit() and len(j) > 4:
                roll += j
```

```
if len(name_rollno[0]) > 1 and len(name_rollno[1])
> 1:
   lst.append(name_rollno)

if y[0].isdigit() and len(y[0]) > 4:
   sub_codes = []
   for code in y:
      if code.isdigit() and len(code) == 3:
         sub_codes.append(code)

   if sub_codes[1] == '006':
      y1 = x[i + 1].split()
      for k in range(len(y1)):
         if k % 2 == 0:
            marks_lst.append(y1[k])
      marks_lst.insert(2, '-')
      marks_lst.insert(3, '-')
   elif sub_codes[1] == '085':
      y1 = x[i + 1].split()
      for k in range(len(y1)):
         if k % 2 == 0:
            marks_lst.append(y1[k])
            marks_lst.insert(1, '-')
            marks_lst.insert(3, '-')
```

```
        elif sub_codes[1] == '018':
            y1 = x[i + 1].split()
            for k in range(len(y1)):
                if k % 2 == 0:
                    marks_lst.append(y1[k])
            marks_lst.insert(1, '-')
            marks_lst.insert(2, '-')
        if marks_lst == []:
            pass
        else:
            l2.append(marks_lst)

for j in range(0, len(lst)):
    lk = lst[j] + l2[j]
    final.append(lk)

if opt == '':
    eng = []
    mat = []
    tam = []
    hin = []
    frc = []
    sci = []
    soc = []
```

```
for a in final:
    if a[2].isdigit():
        eng.append(a[1:3])
    if a[3].isdigit():
        tam.append([a[1]] + [a[3]])
    if a[4].isdigit():
        hin.append([a[1]] + [a[4]])
    if a[5].isdigit():
        frc.append([a[1]] + [a[5]])
    if a[6].isdigit():
        mat.append([a[1]] + [a[6]])
    if a[7].isdigit():
        sci.append([a[1]] + [a[7]])
    if a[8].isdigit():
        soc.append([a[1]] + [a[8]])

sorting(eng);
sorting(tam);
sorting(hin);
sorting(frc);
sorting(mat);
sorting(sci);
sorting(soc)
```

```
count_eng = 1
for t in range(len(eng)):
    if count_eng == 4:
        eng = eng[:t]
        break
    elif eng[t][-1] != eng[t + 1][-1]:
        eng[t].insert(0, count_eng)
        count_eng += 1
    else:
        eng[t].insert(0, count_eng)

count_tam = 1
for t in range(len(tam)):
    if count_tam == 4:
        tam = tam[:t]
        break
    elif tam[t][-1] != tam[t + 1][-1]:
        tam[t].insert(0, count_tam)
        count_tam += 1
    else:
        tam[t].insert(0, count_tam)

count_hin = 1
for t in range(len(hin)):
```

```
if count_hin == 4:
hin = hin[:t]
break
elif hin[t][-1] != hin[t + 1][-1]:
hin[t].insert(0, count_hin)
count_hin += 1
else:
hin[t].insert(0, count_hin)

count_frc = 1
for t in range(len(frc)):
  if count_frc == 4:
    frc = frc[:t]
    break
  elif frc[t][-1] != frc[t + 1][-1]:
    frc[t].insert(0, count_frc)
    count_frc += 1
  else:
    frc[t].insert(0, count_frc)

count_mat = 1
for t in range(len(mat)):
  if count_mat == 4:
    mat = mat[:t]
    break
```

```
 elif mat[t][-1] != mat[t + 1][-1]:
 mat[t].insert(0, count_mat)
 count_mat += 1
 else:
 mat[t].insert(0, count_mat)

count_sci = 1
for t in range(len(sci)):
 if count_sci == 4:
 sci = sci[:t]
 break
 elif sci[t][-1] != sci[t + 1][-1]:
 sci[t].insert(0, count_sci)
 count_sci += 1
 else:
 sci[t].insert(0, count_sci)

count_soc = 1
for t in range(len(soc)):
   if count_soc == 4:
      soc = soc[:t]
      break
   elif soc[t][-1] != soc[t + 1][-1]:
      soc[t].insert(0, count_soc)
      count_soc += 1
```

```
else:
 soc[t].insert(0, count_soc)

def table(sub):
   if sub == eng:
      doc.add_heading("English Toppers", 0)
   if sub == mat:
      doc.add_heading("Maths Toppers", 0)
   if sub == tam:
      doc.add_heading("Tamil Toppers", 0)
   if sub == frc:
      doc.add_heading("French Toppers", 0)
   if sub == hin:
      doc.add_heading("Hindi Toppers", 0)
   if sub == sci:
      doc.add_heading("Science Toppers", 0)
   if sub == soc:
      doc.add_heading("Social Toppers", 0)
   table = doc.add_table(rows=1, cols=3)
   row = table.rows[0].cells
   row[0].text = 'Place'
   row[1].text = 'Name'
   row[2].text = 'Mark'
```

```
        for place, name, mark in sub:
            row = table.add_row().cells
            row[0].text = str(place)
            row[1].text = name
            row[2].text = str(int(mark))
        doc.add_page_break()
        doc.save(file[:-4] + '.docx')

    table(eng);
    table(tam);
    table(hin);
    table(frc);
    table(mat);
    table(sci);
    table(soc)

else:
    name_lst = []
    for i in x1:
        name_lst.append(i.strip())
ds_lst = []
bd_lst = []
for i in final:
    if i[0] not in name_lst:
        ds_lst.append(i)
```

```
    elif i[0] in name_lst:
        bd_lst.append(i)
def option(var,f_ext):
    eng = []
    mat = []
    tam = []
    hin = []
    frc = []
    sci = []
    soc = []
    for a in var:
        if a[2].isdigit():
            eng.append(a[1:3])
        if a[3].isdigit():
            tam.append([a[1]] + [a[3]])
        if a[4].isdigit():
            hin.append([a[1]] + [a[4]])
        if a[5].isdigit():
            frc.append([a[1]] + [a[5]])
        if a[6].isdigit():
            mat.append([a[1]] + [a[6]])
        if a[7].isdigit():
            sci.append([a[1]] + [a[7]])
        if a[8].isdigit():
            soc.append([a[1]] + [a[8]])
```

```
sorting(eng);
sorting(tam);
sorting(hin);
sorting(frc);
sorting(mat);
sorting(sci);
sorting(soc)

count_eng = 1
for t in range(len(eng)):
  if count_eng == 4:
    eng = eng[:t]
    break
  elif eng[t][-1] != eng[t + 1][-1]:
    eng[t].insert(0, count_eng)
    count_eng += 1
  else:
    eng[t].insert(0, count_eng)

count_tam = 1
for t in range(len(tam)):
  if count_tam == 4:
    tam = tam[:t]
    break
  elif tam[t][-1] != tam[t + 1][-1]:
```

```
 tam[t].insert(0, count_tam)
 count_tam += 1
 else:
 tam[t].insert(0, count_tam)

count_hin = 1
if len(hin)!=0:
   for t in range(len(hin)):
      if t==len(hin)-1:
         if hin[t][-1] != hin[t -1][-1]:
            hin[t].insert(0,count_hin)
         elif count_hin!=1:
            hin[t].insert(0,count_hin-1)
         else:
            hin[t].insert(0,count_hin)
      elif count_hin == 4:
         hin = hin[:t]
         break
      elif hin[t][-1] != hin[t + 1][-1]:
         hin[t].insert(0, count_hin)
         count_hin += 1
      else:
         hin[t].insert(0, count_hin)
else:
   pass
```

```
count_frc = 1
if len(frc)!=0:
    for t in range(len(frc)):
        if t==len(frc)-1:
            if frc[t][-1] != frc[t -1][-1]:
                frc[t].insert(0,count_frc)
            elif count_frc!=1:
                frc[t].insert(0,count_frc-1)
            else:
                frc[t].insert(0,count_frc)
        elif count_frc == 4:
            frc = frc[:t]
            break
        elif frc[t][-1] != frc[t + 1][-1]:
            frc[t].insert(0, count_frc)
            count_frc += 1
        else:
            frc[t].insert(0, count_frc)

count_mat = 1
for t in range(len(mat)):
    if count_mat == 4:
        mat = mat[:t]
        break
    elif mat[t][-1] != mat[t + 1][-1]:
```

```
 mat[t].insert(0, count_mat)
 count_mat += 1
 else:
 mat[t].insert(0, count_mat)

count_sci = 1
for t in range(len(sci)):
   if count_sci == 4:
      sci = sci[:t]
      break
   elif sci[t][-1] != sci[t + 1][-1]:
      sci[t].insert(0, count_sci)
      count_sci += 1
   else:
      sci[t].insert(0, count_sci)
count_soc = 1
for t in range(len(soc)):
   if count_soc == 4:
      soc = soc[:t]
      break
   elif soc[t][-1] != soc[t + 1][-1]:
      soc[t].insert(0, count_soc)
     count_soc += 1
else:
 soc[t].insert(0, count_soc)
```

```
def table(sub,f_ext):
  if sub == eng:
    f_ext.add_heading("English Toppers", 0)
  if sub == mat:
    f_ext.add_heading("Maths Toppers", 0)
  if sub == tam:
    f_ext.add_heading("Tamil Toppers", 0)
  if sub == frc :
    f_ext.add_heading("French Toppers", 0)
  if sub == hin:
    f_ext.add_heading("Hindi Toppers", 0)
  if sub == sci:
    f_ext.add_heading("Science Toppers", 0)
  if sub == soc:
    f_ext.add_heading("Social Toppers", 0)
  table = f_ext.add_table(rows=1, cols=3)
  row = table.rows[0].cells
  row[0].text = 'Place'
  row[1].text = 'Name'
  row[2].text = 'Mark'
  for place, name, mark in sub:
      row = table.add_row().cells
       row[0].text = str(place)
       row[1].text = name
```

```
        row[2].text = str(int(mark))
        f_ext.add_page_break()
        if f_ext == doc:
            f_ext.save(file[:-4] + '(DS).docx')
        else:
            f_ext.save(file[:-4] + '(BD).docx')

if len(eng) != 0:
    table(eng,f_ext)
if len(tam) != 0:
    table(tam,f_ext)
if len(hin)!=0:
    table(hin,f_ext)
if len(frc)!=0:
    table(frc,f_ext)
if len(mat) != 0:
    table(mat,f_ext)
if len(sci) != 0:
    table(sci,f_ext)
if len(soc) != 0:
    table(soc,f_ext)
 eng.clear(); mat.clear(); tam.clear(); hin.clear();
frc.clear(); sci.clear(); soc.clear()
option(ds_lst, doc)
option(bd_lst, doc1)
```

```
def sub_top_g12(file,opt=''):
    global ds, bd, x1, g, doc1 ,doc
    f = open(file)
    if opt == '':
        doc = docx.Document()
    else:
        namelist = open(opt)
        doc = docx.Document()
        doc1 = docx.Document()
        x1 = namelist.readlines()
    x = f.readlines()
    lst = []
    l2 = []
    eng = []
    mat = []
    physics = []
    chemistry = []
    C_S = []
    ap_maths = []
    biology = []
    economics = []
    account = []
    b_s = []
```

```
final = []

  def sorting(sub_name):
    for s in range(len(sub_name)):
      for total_lst in range(0, len(sub_name) - s - 1):
        if sub_name[total_lst][-1] <
sub_name[total_lst + 1][-1]:
            sub_name[total_lst], sub_name[total_lst +
1] = sub_name[total_lst + 1], sub_name[total_lst]

  for i in range(len(x)):
    y = x[i].split()
    if y == []:
      pass
    else:
      roll = ''
      name = ''
      english, maths, phy, che, cs, ap_mat, bio, eco,
acc, bs = '-', '-', '-', '-', '-', '-', '-', '-', '-', '-'
      marks_lst = []

      for j in y:
        name_rollno = [roll, name[1:]]
```

```
if y[0].isdigit() and j.isalpha() and j != ('PASS' or
FAIL'):
            name = name + j + ' '
        elif y[0].isdigit() and j.isdigit() and len(j) > 4:
            roll += j

    if len(name_rollno[0]) > 1 and
len(name_rollno[1]) > 1:
        lst.append(name_rollno)

    if y[0].isdigit() and len(y[0]) > 4:
        sub_codes = []
        marks_lst = [english, maths, phy, che, cs,
ap_mat, bio, eco, acc, bs]

        for code in y:
            if code.isdigit() and len(code) == 3:
                sub_codes.append(code)

        y_1 = x[i + 1].split()
        for pos in range(len(y_1)):
            try:
                if y_1[pos].isdigit() == False:
```

```
y_1.remove(y_1[pos])
            except:
              break

          for m in range(len(sub_codes)):
            if sub_codes[m] == '301':
              marks_lst[0] = y_1[m]
            elif sub_codes[m] == '041':
              marks_lst[1] = y_1[m]
            elif sub_codes[m] == '042':
              marks_lst[2] = y_1[m]
            elif sub_codes[m] == '043':
              marks_lst[3] = y_1[m]
            elif sub_codes[m] == '083':
              marks_lst[4] = y_1[m]
            elif sub_codes[m] == '241':
              marks_lst[5] = y_1[m]
            elif sub_codes[m] == '044':
              marks_lst[6] = y_1[m]
            elif sub_codes[m] == '030':
              marks_lst[7] = y_1[m]
            elif sub_codes[m] == '055':
              marks_lst[8] = y_1[m]
```

```
elif sub_codes[m] == '054':
                marks_lst[9] = y_1[m]
            else:
                print('New sub_code found!', y[m])

        if marks_lst == []:
            pass
        else:
            l2.append(marks_lst)

    for j in range(0, len(lst)):
        lk = lst[j] + l2[j]
        final.append(lk)

    if opt=='':
        eng = []
        mat = []
        physics = []
        chemistry = []
        C_S = []
        ap_maths = []
        biology = []
        economics = []
```

```
account = []
b_s = []
for a in final:
        if a[2].isdigit():
           eng.append(a[1:3])
        if a[3].isdigit():
           mat.append([a[1]] + [a[3]])
        if a[4].isdigit():
           physics.append([a[1]] + [a[4]])
        if a[5].isdigit():
           chemistry.append([a[1]] + [a[5]])
        if a[6].isdigit():
           C_S.append([a[1]] + [a[6]])
        if a[7].isdigit():
           ap_maths.append([a[1]] + [a[7]])
        if a[8].isdigit():
           biology.append([a[1]] + [a[8]])
        if a[9].isdigit():
           economics.append([a[1]] + [a[9]])
        if a[10].isdigit():
           account.append([a[1]] + [a[10]])
        if a[11].isdigit():
           b_s.append([a[1]] + [a[11]])
```

```
count_eng = 1
    for t in range(len(eng)):
        if count_eng == 4:
            eng = eng[:t]
            break
        elif eng[t][-1] != eng[t + 1][-1]:
            eng[t].insert(0, count_eng)
            count_eng += 1
        else:
            eng[t].insert(0, count_eng)

    count_mat = 1
    for t in range(len(mat)):
        if count_mat == 4:
            mat = mat[:t]
            break
        elif mat[t][-1] != mat[t + 1][-1]:
            mat[t].insert(0, count_mat)
            count_mat += 1
        else:
            mat[t].insert(0, count_mat)
```

```
count_phy = 1
    for t in range(len(physics)):
      if count_phy == 4:
        physics = physics[:t]
        break
      elif physics[t][-1] != physics[t + 1][-1]:
        physics[t].insert(0, count_phy)
        count_phy += 1
      else:
        physics[t].insert(0, count_phy)

    count_che = 1
    for t in range(len(chemistry)):
      if count_che == 4:
        chemistry = chemistry[:t]
        break
      elif chemistry[t][-1] != chemistry[t + 1][-1]:
        chemistry[t].insert(0, count_che)
        count_che += 1
      else:
        chemistry[t].insert(0, count_che)
```

```
count_cs = 1
    for t in range(len(C_S)):
      if count_cs == 4:
        C_S = C_S[:t]
        break
      elif C_S[t][-1] != C_S[t + 1][-1]:
        C_S[t].insert(0, count_cs)
        count_cs += 1
      else:
        C_S[t].insert(0, count_cs)

    count_ap_mat = 1
    for t in range(len(ap_maths)):
      if count_ap_mat == 4:
        ap_maths = ap_maths[:t]
        break
      elif ap_maths[t][-1] != ap_maths[t + 1][-1]:
        ap_maths[t].insert(0, count_ap_mat)
        count_ap_mat += 1
      else:
        ap_maths[t].insert(0, count_ap_mat)
```

```
count_bio = 1
    for t in range(len(biology)):
      if count_bio == 4:
        biology = biology[:t]
        break
      elif biology[t][-1] != biology[t + 1][-1]:
        biology[t].insert(0, count_bio)
        count_bio += 1
      else:
        biology[t].insert(0, count_bio)

    count_eco = 1
    for t in range(len(economics)):
      if count_eco == 4:
        economics = economics[:t]
        break
      elif economics[t][-1] != economics[t + 1][-1]:
        economics[t].insert(0, count_eco)
        count_eco += 1
      else:
        economics[t].insert(0, count_eco)
```

```
count_acc = 1
    for t in range(len(account)):
      if count_acc == 4:
        account = account[:t]
        break
      elif account[t][-1] != account[t + 1][-1]:
        account[t].insert(0, count_acc)
        count_acc += 1
      else:
        account[t].insert(0, count_acc)

    count_bs = 1
    for t in range(len(b_s)):
      if count_bs == 4:
        b_s = b_s[:t]
        break
      elif b_s[t][-1] != b_s[t + 1][-1]:
        b_s[t].insert(0, count_bs)
        count_bs += 1
      else:
        b_s[t].insert(0, count_bs)
```

```
def table(sub):
        if sub == eng:
          doc.add_heading("English Toppers", 0)
        if sub == mat:
          doc.add_heading("Maths Toppers", 0)
        if sub == physics:
          doc.add_heading("Physics Toppers", 0)
        if sub == chemistry:
          doc.add_heading("Chemistry Toppers", 0)
        if sub == C_S:
          doc.add_heading("C.S Toppers", 0)
        if sub == ap_maths:
          doc.add_heading("App.Maths Toppers", 0)
        if sub == biology:
          doc.add_heading("Biology Toppers", 0)
        if sub == economics:
          doc.add_heading("Economics Toppers", 0)
        if sub == account:
          doc.add_heading("Accountancy
Toppers",0)
        if sub == b_s:
          doc.add_heading("B.S Toppers", 0)
```

```
        table = doc.add_table(rows=1, cols=3)
        table.autofit = True
        row = table.rows[0].cells
        row[0].text = 'Place'
        row[1].text = 'Name'
        row[2].text = 'Mark'

        for place, name, mark in sub:
          row = table.add_row().cells
          row[0].text = str(place)
          row[1].text = name
          row[2].text = str(int(mark))
        doc.add_page_break()
        doc.save(file[:-4] + '.docx')

    table(eng); table(mat); table(physics);
table(chemistry); table(C_S); table(ap_maths);
table(biology); table(economics);
table(account); table(b_s)
```

```
else:
    name_lst = []
    for i in x1:
      name_lst.append(i.strip())

    ds_lst = []
    bd_lst = []
    for i in final:
      if i[0] not in name_lst:
        ds_lst.append(i)
      elif i[0] in name_lst:
        bd_lst.append(i)

    def option(var, f_ext):
      eng = []
      mat = []
      physics = []
      chemistry = []
      C_S = []
      ap_maths = []
      biology = []
      economics = []
      account = []
```

```
b_s = []
        for a in final:
            if a[2].isdigit():
                eng.append(a[1:3])
            if a[3].isdigit():
                mat.append([a[1]] + [a[3]])
            if a[4].isdigit():
                physics.append([a[1]] + [a[4]])
            if a[5].isdigit():
                chemistry.append([a[1]] + [a[5]])
            if a[6].isdigit():
                C_S.append([a[1]] + [a[6]])
            if a[7].isdigit():
                ap_maths.append([a[1]] + [a[7]])
            if a[8].isdigit():
                biology.append([a[1]] + [a[8]])
            if a[9].isdigit():
                economics.append([a[1]] + [a[9]])
            if a[10].isdigit():
                account.append([a[1]] + [a[10]])
            if a[11].isdigit():
                b_s.append([a[1]] + [a[11]])
```

```
count_eng = 1
        for t in range(len(eng)):
            if count_eng == 4:
                eng = eng[:t]
                break
            elif eng[t][-1] != eng[t + 1][-1]:
                eng[t].insert(0, count_eng)
                count_eng += 1
            else:
                eng[t].insert(0, count_eng)

        count_mat = 1
        for t in range(len(mat)):
            if count_mat == 4:
                mat = mat[:t]
                break
            elif mat[t][-1] != mat[t + 1][-1]:
                mat[t].insert(0, count_mat)
                count_mat += 1
            else:
                mat[t].insert(0, count_mat)
```

```
count_phy = 1
        if len(physics) != 0:
            for t in range(len(physics)):
                if t == len(physics) - 1:
                    if physics[t][-1] != physics[t - 1][-1]:
                        physics[t].insert(0, count_phy)
                    elif count_phy != 1:
                        physics[t].insert(0, count_phy - 1)
                    else:
                        physics[t].insert(0, count_phy)
                elif count_phy == 4:
                    physics = physics[:t]
                    break
                elif physics[t][-1] != physics[t + 1][-1]:
                    physics[t].insert(0, count_phy)
                    count_phy += 1
                else:
                    physics[t].insert(0, count_phy)
        else:
            pass

        count_che = 1
```

```
if len(chemistry) != 0:
        for t in range(len(chemistry)):
          if t == len(chemistry) - 1:
            if chemistry[t][-1] != chemistry[t - 1][-1]:
              chemistry[t].insert(0, count_che)
            elif count_che != 1:
              chemistry[t].insert(0, count_che - 1)
            else:
              chemistry[t].insert(0, count_che)
          elif count_che == 4:
            chemistry = chemistry[:t]
            break
          elif chemistry[t][-1] != chemistry[t + 1][-1]:
            chemistry[t].insert(0, count_che)
            count_che += 1
          else:
            chemistry[t].insert(0, count_che)
      else:
        pass

      count_cs = 1
```

```
for t in range(len(C_S)):
    if count_cs == 4:
        C_S = C_S[:t]
        break
    elif C_S[t][-1] != C_S[t + 1][-1]:
        C_S[t].insert(0, count_cs)
        count_cs += 1
    else:
        C_S[t].insert(0, count_cs)

count_ap_mat = 1
if len(ap_maths) != 0:
    for t in range(len(ap_maths)):
        if t == len(ap_maths) - 1:
            if ap_maths[t][-1] != ap_maths[t - 1][-1]:
                ap_maths[t].insert(0, count_ap_mat)
            elif count_ap_mat != 1:
                ap_maths[t].insert(0, count_ap_mat - 1)
            else:
                ap_maths[t].insert(0, count_ap_mat)
        elif count_ap_mat == 4:
            ap_maths = ap_maths[:t]
```

```
break
            elif ap_maths[t][-1] != ap_maths[t + 1][-1]:
              ap_maths[t].insert(0, count_ap_mat)
              count_ap_mat += 1
            else:
              ap_maths[t].insert(0, count_ap_mat)
        else:
          pass

        count_bio = 1
        for t in range(len(biology)):
          if count_bio == 4:
            biology = biology[:t]
            break
          elif biology[t][-1] != biology[t + 1][-1]:
            biology[t].insert(0, count_bio)
            count_bio += 1
          else:
            biology[t].insert(0, count_bio)

        count_eco = 1
        if len(economics) != 0:
```

```
break
            elif ap_maths[t][-1] != ap_maths[t + 1][-1]:
              ap_maths[t].insert(0, count_ap_mat)
              count_ap_mat += 1
            else:
              ap_maths[t].insert(0, count_ap_mat)
        else:
          pass

        count_bio = 1
        for t in range(len(biology)):
          if count_bio == 4:
            biology = biology[:t]
            break
          elif biology[t][-1] != biology[t + 1][-1]:
            biology[t].insert(0, count_bio)
            count_bio += 1
          else:
            biology[t].insert(0, count_bio)

        count_eco = 1
        if len(economics) != 0:
```

```
for t in range(len(economics)):
            if t == len(economics) - 1:
              if economics[t][-1] != economics[t - 1][-1]:
                economics[t].insert(0, count_eco)
              elif count_eco != 1:
                economics[t].insert(0, count_eco - 1)
              else:
                economics[t].insert(0, count_eco)
            elif count_eco == 4:
              economics = economics[:t]
              break
            elif economics[t][-1] != economics[t + 1][-1]:
              economics[t].insert(0, count_eco)
              count_eco += 1
            else:
              economics[t].insert(0, count_eco)
        else:
          pass

        count_acc = 1
        if len(account) != 0:
```

```
for t in range(len(account)):
            if t == len(account) - 1:
              if account[t][-1] != account[t - 1][-1]:
                account[t].insert(0, count_acc)
              elif count_acc != 1:
                account[t].insert(0, count_acc - 1)
              else:
                account[t].insert(0, count_acc)
            elif count_acc == 4:
              account = account[:t]
              break
            elif account[t][-1] != account[t + 1][-1]:
              account[t].insert(0, count_acc)
              count_acc += 1
            else:
              account[t].insert(0, count_acc)
        else:
          pass

        count_bs = 1
        if len(b_s) != 0:
```

```
for t in range(len(b_s)):
            if t == len(b_s) - 1:
              if b_s[t][-1] != b_s[t - 1][-1]:
                b_s[t].insert(0, count_bs)
              elif count_bs != 1:
                b_s[t].insert(0, count_bs - 1)
              else:
                b_s[t].insert(0, count_bs)
            elif count_bs == 4:
              b_s = b_s[:t]
              break
            elif b_s[t][-1] != b_s[t + 1][-1]:
              b_s[t].insert(0, count_bs)
              count_bs += 1
            else:
              b_s[t].insert(0, count_bs)
        else:
          pass

        def table(sub,f_ext):
          if sub == eng:
            f_ext.add_heading("English Toppers", 0)
```

```
if sub == mat:
   f_ext.add_heading("Maths Toppers", 0)
if sub == physics:
   f_ext.add_heading("Physics Toppers", 0)
if sub == chemistry:
   f_ext.add_heading("Chemistry Toppers", 0)
if sub == C_S:
   f_ext.add_heading("C.S Toppers", 0)
if sub == ap_maths:
   f_ext.add_heading("App.Maths Toppers", 0)
if sub == biology:
   f_ext.add_heading("Biology Toppers", 0)
if sub == economics:
   f_ext.add_heading("Economics Toppers", 0)
if sub == account:
   f_ext.add_heading("Accountancy Toppers", 0)
if sub == b_s:
   f_ext.add_heading("B.S Toppers", 0)

table = f_ext.add_table(rows=1, cols=3)
row = table.rows[0].cells
row[0].text = 'Place'
row[1].text = 'Name'
```

```
row[2].text = 'Mark'

        for place, name, mark in sub:
            row = table.add_row().cells
            row[0].text = str(place)
            row[1].text = name
            row[2].text = str(int(mark))
        f_ext.add_page_break()
        if f_ext == doc:
            f_ext.save(file[:-4] + '(DS).docx')
        elif f_ext == doc1:
            f_ext.save(file[:-4] + '(BD).docx')

    if len(eng) != 0:
        table(eng, f_ext)
    if len(mat) != 0:
        table(mat, f_ext)
    if len(physics) != 0:
        table(physics,f_ext)
    if len(chemistry) != 0:
        table(chemistry,f_ext)
    if len(C_S) != 0:
        table(C_S,f_ext)
```

```
if len(ap_maths) != 0:
            table(ap_maths,f_ext)
        if len(biology) != 0:
            table(biology,f_ext)
        if len(economics) != 0:
            table(economics,f_ext)
        if len(account) != 0:
            table(account,f_ext)
        if len(b_s) != 0:
            table(b_s,f_ext)
    print(len(ds_lst))
    print(len(bd_lst))
    option(ds_lst, doc)
    option(bd_lst, doc1)
```

I/O SCREEN DESIGN

Conclusion

An user friendly environment is created for the users.The concept of loops,modules and functions were clearly understood. This project also enables to learn lots of new things.Difficulty in python was cleared
after completion of the project.

BIBLIOGRAPHY

BOOKS:

- Computer science with python textbook for class XII -Sumitha arora -Published by : DHANPAT RAI & CO. (Pvt.) Ltd.

- Computer science with python textbook for class XI -Sumitha arora -Published by : DHANPAT RAI & CO. (Pvt.) Ltd.

LINKS:

- www.Google.com
- www.youtube.com