

CMPE 230 HW 2 REPORT

Students: Olcayto Türker - 2016400180, Sertay Akpınar – 2016400075

In this project, we developed a Python program that crawls Bogazici University's OBIKAS registration pages and extracts course offering information.

We used pandas and sys library in this project. We used for loop to get every program in programs list. We defined a “func” function which takes the year in first argument, the year in second argument and the semester in first argument. In this function, we created the html and read it for each departments and desired semesters. We used try/except to handle the empty page cases. Pandas is used for reading the html.

```
for x in range(len(dfs[3][2])):
    if dfs[3][2][x] != 'LAB' and dfs[3][2][x] != 'P.S.' and dfs[3][2][x] != 'Name':
        courseName.append(dfs[3][2][x])

for x in range(len(dfs[3][0])):
    if dfs[3][0][x] != 'Code.Sec' and dfs[3][5][x] != 'STAFF' and dfs[3][5][x] != 'STAFF STAFF' and dfs[3][2][x] != 'LAB' and dfs[3][2][x] != 'P.S.':
        code = str(dfs[3][0][x])
        courseCode.append(code[:-3])

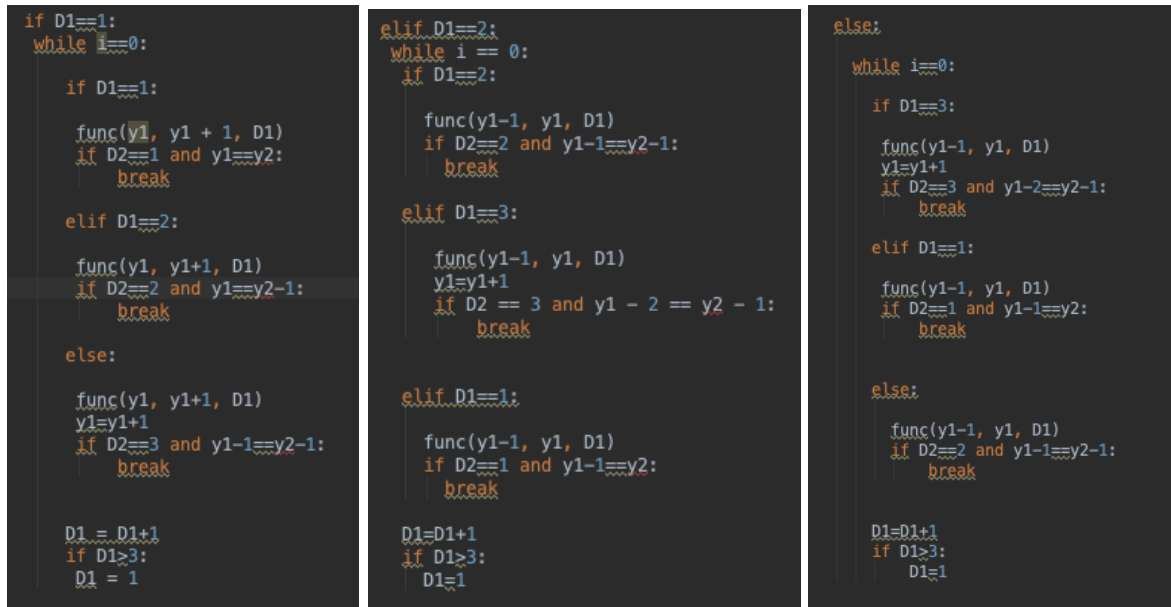
for x in range(len(dfs[3][5])):
    if dfs[3][5][x] != 'STAFF' and dfs[3][5][x] != 'STAFF STAFF' and dfs[3][5][x] != 'Instr.' and dfs[3][2][x] != 'LAB' and dfs[3][2][x] != 'P.S.':
        instructors.append(dfs[3][5][x])
```

In the picture above, we used for loop to get every data(courseName, courseCode, instructors) from the web-site. After that, we keep all courseCodes, instructors and courseNames in different lists. Set is used for removing the duplicates in these lists.

U represents the number of all undergraduate courses and G represents the number of all graduate courses and I represents the number of all instructors. We calculate this variables in “func”.

“termcourses” list takes “courseCode” lists as elements. In that case, we can keep all of the terms and its courses in termcourses. “allUG”, “allG” and “allI” keeps the datas for every semester.(Each semester is one element)

Before we coded the if statements, we added a for loop because this process repeats for all programs. Arguments are checked by if statements and equalizes the argument to a number(1,2 or 3) according to the semester here.



```

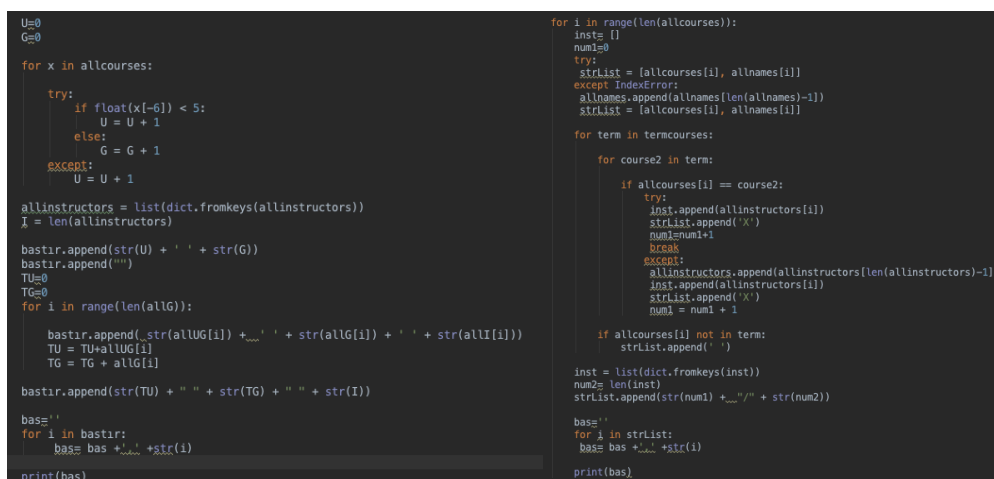
if D1==1:
    while i==0:
        if D1==1:
            func(y1, y1 + 1, D1)
            if D2==1 and y1==y2:
                break
        elif D1==2:
            func(y1, y1+1, D1)
            if D2==2 and y1==y2-1:
                break
        else:
            func(y1, y1+1, D1)
            y1=y1+1
            if D2==3 and y1-1==y2-1:
                break
        D1 = D1+1
        if D1>3:
            D1 = 1

elif D1==2:
    while i == 0:
        if D1==2:
            func(y1-1, y1, D1)
            if D2==2 and y1-1==y2-1:
                break
        elif D1==3:
            func(y1-1, y1, D1)
            y1=y1+1
            if D2 == 3 and y1 - 2 == y2 - 1:
                break
        elif D1==1:
            func(y1-1, y1, D1)
            if D2==1 and y1-1==y2:
                break
        D1=D1+1
        if D1>3:
            D1=1

else:
    while i==0:
        if D1==3:
            func(y1-1, y1, D1)
            y1=y1+1
            if D2==3 and y1-2==y2-1:
                break
        elif D1==1:
            func(y1-1, y1, D1)
            if D2==1 and y1-1==y2:
                break
        else:
            func(y1-1, y1, D1)
            if D2==2 and y1-1==y2-1:
                break
        D1=D1+1
        if D1>3:
            D1=1

```

The code in above picture, is actually the whole process. We check the semesters by if functions, and call the “func” with the desired parameters. If the summer term is called, it increments the year. If the process should end, we break the while loop.



```

U=0
G=0

for x in allcourses:
    try:
        if float(x[-6]) < 5:
            U = U + 1
        else:
            G = G + 1
    except:
        U = U + 1

allinstructors = list(dict.fromkeys(allinstructors))
I = len(allinstructors)

bastir.append(str(U) + ' ' + str(G))
bastir.append(' ')
TU=0
TG=0
for i in range(len(allG)):
    bastir.append(str(allUG[i]) + '...' + str(allG[i]) + ' ' + str(allI[i]))
    TU = TU+allUG[i]
    TG = TG + allG[i]

bastir.append(str(TU) + " " + str(TG) + " " + str(I))

bas= ' '
for i in bastir:
    bas= bas + '...' + str(i)

print(bas)

for i in range(len(allcourses)):
    inst= []
    num1=0
    try:
        strList = [allcourses[i], allnames[i]]
    except IndexError:
        allnames.append(allnames[len(allnames)-1])
        strList = [allcourses[i], allnames[i]]

    for term in termcourses:
        for course2 in term:
            if allcourses[i] == course2:
                try:
                    inst.append(allinstructors[i])
                    strList.append('X')
                    num1=num1+1
                except:
                    allinstructors.append(allinstructors[len(allinstructors)-1])
                    inst.append(allinstructors[i])
                    strList.append('X')
                    num1 = num1 + 1

            if allcourses[i] not in term:
                strList.append(' ')

    inst = list(dict.fromkeys(inst))
    num2= len(inst)
    strList.append(str(num1) + '...' + str(num2))

    bas= ' '
    for i in strList:
        bas= bas + '...' + str(i)

    print(bas)

```

The code in first above picture, we initialize U and G to zero to obtain the next U’s and G’s for every semester. Try/except is used for handling possible index errors. “Bastir” is a stringlist represents a row in the table.

The length of allG is equals to number of semesters. TU represents total undergrad. Courses and it updates itself by adding the ith index of allUG. It appends the total datas(TU, TI, TG) because we have to print these on the table too. “bas” is used for printing the elements of “bastır”. This is for printing the first line.

In the second picture, we used try/except again for handling the possible index errors. The nested loops gets required value for every courses in “termcourses”. If the course is opened in the semester, then it print “X” and break the inside for loop. If the for loop ends without any break commands that means the course does not overlap in the term’s courses, so print space. After that process we set the instructors and get it’s length. Then we append the required datas to the stringlist, in that case it corresponds total offerings. This process is for printing each course in each department. In the end, we give some arguments to try our code and we get the correct results in a csv. format.