Тема 8 ЖОЛДАРМЕН ЖҰМЫС ІСТЕУ

Тәжірибелік жұмыс №6

8.1 Тәжірибелік жұмыстың мақсатты

Жолдық айнымалыларды қолдану арқылы есептеу процесін құру және бағдарламалау бойынша практикалық дағдыларды алу.

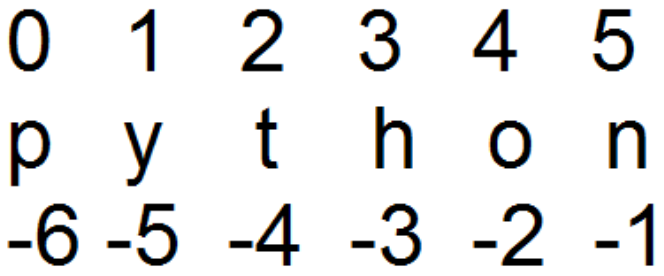
8.2 Теориялық мәлімдемелер

Айта кету керек, біз бұрын Python бағдарламалау тілінде жолдармен жұмыс жасаудың алғашқы дағдыларын алдық. Деректерді енгізуге шақыру немесе бағдарламада жауап беру, сондай-ақ жолды аудару, кесте, жолды пішімдеу сияқты сұрақтар қарастырылады. Бұл тақырыпта біз жолдармен жұмыс істеуге арналған функциялар мен әдістермен танысамыз.

Python-дағы жол-бұл **str** класының нысаны. Бұрын тақырыптарда сандық деректермен жұмыс істеу үшін типтерді келтіру функциялары қолданылды, мысалы, **int**, бұл келесі мысалда көрсетілгендей:

```
ind=int(input(" \n Введите индекс элемента "))
```

Суретте жолдар нөлден индекстелетіні көрсетілген, яғни егер **stroke="python"** операторы болса, онда жолдағы бірінші таңба нөлге тең болады және оған сілтеме **stroka[0]** ретінде жазылады. Сонымен қатар, Python-дағы жол элементтеріне, теріс индекстерді көрсету арқылы қол жеткізуге болады, мысалы, **print(stroka[-6])** операторы **p** таңбасын экранға шығарады.



Сурет 8.1– Жолдарды индекстеу

Бағдарламалау кезінде бір таңбаны екінші таңбамен салыстырмас бұрын оны 128 әріптік-цифрлық таңбаларды кодтауды қолдайтын ASCII (American Standard Code for Information Interchange ақпарат алмасудың американдық стандарттық коды) кестесі арқылы санға айналдыру қажет екені белгілі. Сонымен, жолдармен жұмыс жасау кезінде **"A"** символы **"a"**

символымен бірдей емес екенін білу керек. Жолдың ұзындығы тек компьютердің жедел жадының көлемімен шектеледі және жол элементіне жүгіну үшін оның индексін тік жақшада көрсету жеткілікті. Сол сияқты, тізім элементіне жүгіну бұрын қарастырылған.

Python-дағы жолдар өзгермейтін дәйектілік болып табылады және **for** операторымен циклды қолдана отырып өңделеді. Индекске жүгіну арқылы жол таңбасын өзгерту мүмкін емес екенін түсіну маңызды.

```
stroka="Python"
stroka[0]="p"  #недопустимый оператор
```

Жолды өңдеу кезінде оған төмендегі тізімде көрсетілгендей тікелей циклде жүгінуге болады.

```
stroka="python"
for i in stroka:
    print(i, end=" ")
```

Бұл кодтың нәтижесі **p y t h o n** сөзін экранға шығару болады.

Len () функциясы*.Жолдармен жұмыс істеу кезінде* **Len()** функциясы пайдалы болуы мүмкін, оның мақсаты жолдың ұзындығын анықтау болып табылады.

Ord() функциясы Тағы бір функция - **ord(),** оның синтаксисі **ord("таңба")** көрсетілген таңбаның кодын қайтарады.

Chr() функциясы **Chr(Сан)** синтаксисі болатын функцияның әрекеті **ord()** функциясына тікелей қарама-қарсы, атап айтқанда таңбаның ASCII коды бойынша таңбаны қайтару.

Upper() әдісі **.Stroka.upper()** әдіс синтаксисі. Жолдың барлық таңбаларын жоғарғы регистрге түрлендіреді

Lower() әдісі. **Stroka.lower()** әдіс синтаксисі.. Жолдың барлық таңбаларын төменгі регистрге түрлендіреді.

Swapcase()әдісі **. Stroka. swapcase()** әдіс синтаксисі. Кіші әріптермен жазылған жолдың барлық таңбаларын жоғарғы және керісінше түрлендіреді.

Capitalize()әдісі **. Stroka.capitalize()** әдіс синтаксисі. Жолдағы бірінші әріпті жоғарғы регистрге, ал қалғанын төменгі регистрге түрлендіреді.

Title() әдісі. **Stroka.title()** әдіс синтаксисы. Жолдағы барлық бірінші әріптерді жоғарғы регистрге, ал қалғандары төменгі регистрге түрлендіреді.

Startswith() әдісі **. Stroka.startswith(podstroka)** әдіс синтаксисі. **Stroka** жолының **podstroka**(жол үзіндісінен) көрсетілген ішкі жолдан басталатынын тексереді.

Endswith() әдісі **.**Әдіс синтаксисі: **stroka.endswith(podstroka). Stroka** жолының **podstroka** ішкі жолымен аяқталғанын тексереді.

Replace()әдісі **.**Әдіс синтаксисі: **stroka. replace (old, new),** мұнда **old**-ішкі жолды ауыстыру үшін, **new**-жаңа ішкі жол. Әдіс **stroka** жолында **old** ішкі жолды тауып алып **new** ішкі жолмен ауыстырады.

Rfind()әдісі **.**Әдіс синтаксисі: **stroke.rfind (podstr),** мұнда **podstr** - ішкі жол. Әдіс жолға соңғы ішкі жолдың кіру позициясын қайтарады. Егер **ішкі жол** табылмаса, онда **-1** мәні қайтарылады. **Podstr** параметрінен кейін бастапқы позицияны және ішкі жолды іздейтін жолдағы соңғы позицияларды көрсетуге болады. Бұл параметрлер міндетті емес, егер бастапқы позиция болмаса, іздеу жолдың басынан бастап жүзеге асырылады.

Find()әдісі **.**Әдіс синтаксисі: **stroke.flnd (podstr),** мұнда **podstr** - ішкі жол. Алдыңғы әдіске қарағанда, **find()** әдісі ішкі жолдың жолға бірінші кіру позициясын қайтарады. Егер **ішкі жол** табылмаса, онда **-1** мәні қайтарылады.

Count() әдісі.Әдіс синтаксисі: **stroke.count (podstr),** мұнда **podstr** - ішкі жол. Бұл әдіс **podstr** ішкі жолының пайда болу санын **stroka** жолына қайтарады.

Strip() әдісі Әдіс синтаксисі: **stroka.strip().** Бұл әдіс **stroka** жолындағы бастапқы және соңғы бос орындарды жояды.

Lstrip() и Rstrip()әдістері **.** Олардың синтаксисі мен әрекеті қарастырылған **strip ()** әдісіне ұқсас, айырмашылығы - **lstrip()** әдісі бастапқы жолдың басынан сол жақтағы бос орын таңбаларын жояды, ал **rstrip()** әдісі бастапқы жолдың соңында. **Strip(), strip()** және **lstrip()** әдістерін қолдана отырып, бос орындарды ғана емес, жоюға болатындығын ескерсек. Сонымен, егер әдістердің бірінің параметрі ретінде таңбаны немесе таңбалар тізбегін көрсетсеңіз, онда ол (олар) жойылады.

Split() әдісі. Әдіс синтаксисі: **stroka. split().** Әдіс жолды ішкі жолдарға бөледі және оларды тізімге қосады. Егер бөлгіш параметр ретінде болса, онда жол көрсетілген бөлгішке сәйкес ішкі жолдарға бөлінеді. Ол болмаған жағдайда бос орын бөлгіш болып саналады.

Join()әдісі **.**Синтаксис әдісі: **бөлгіш.join (spisok),** мұндағы **spisok** - тізім немесе кортеж. Жолды тізімге айналдырудың тағы бір әдісі - **list()** функциясын қолдану, ал кері әрекеті **]join()** әдісімен жүзеге асырылады. Мұндай түрлендірулер жол элементін индекс бойынша өзгерту үшін қажет, өйткені жолдар бастапқыда өзгермейді.

8.3 Тәжірибелік жұмысты орындауға арналған тапсырма мысалдары

Таңбалар санын анықтау.

Есеп 8.3.1. Пайдаланушы **stroka** бастапқы жолын және **simvol** таңбасын енгізеді. Берілген таңба бастапқы жолда қанша рет кездесетінін есептеңіз.

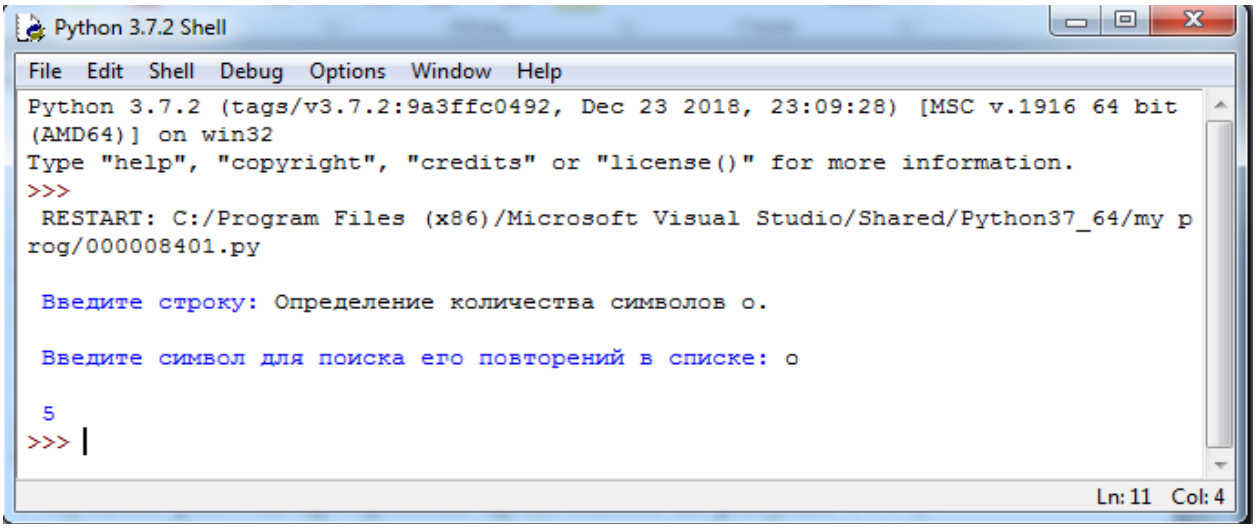
Шешімі. Бастапқыда **k** ұяшығын нөлдендіргеннен кейін, ол таңбалар есептегішінің рөлін атқарады, біз бастапқы жолдың барлық таңбаларын төменгі **lower()** әдісімен кіші әріптерге түрлендіреміз. Содан кейін циклде **spisok[i]** жолының ағымдағы символы ізделетін **simvol** салыстырылады. **I** цикл параметрі 0-ден(жолдың басы) жолдың соңына дейін өзгереді(бұл үшін **len (spisok)**

функциясының мәні бар **n** ұяшығы жауап береді). Жолдағы табылған таңбалар санын есептеуді **k+=1** операторы қамтамасыз етеді.

Төменде есепті шешуге жауап беретін бағдарлама коды берілген.

```
k=0
stroka=input("\n Введите строку: ")
stroka1=stroka.lower()
spisok=list(stroka1)
simvol=input("\n Введите символ для поиска его повторений в списке: ")
n=len(spisok)
for i in range(0, n):
    if spisok[i]==simvol:
        k+=1
print("\n", k)
```

Бағдарламаның нәтижесі 8.2-ші суретте көрсетілген



Сурет 8.2 – Жолдағы таңбалар санын анықтау бойынша бағдарлама

Жолдағы таңбаларды ауыстыру.

Есеп 8.3.2. Пайдаланушы **stroka** бастапқы жолын және **simvol** таңбасын енгізеді. Бастапқы жолдағы бос орындарды көрсетілген таңбамен ауыстырыңыз.

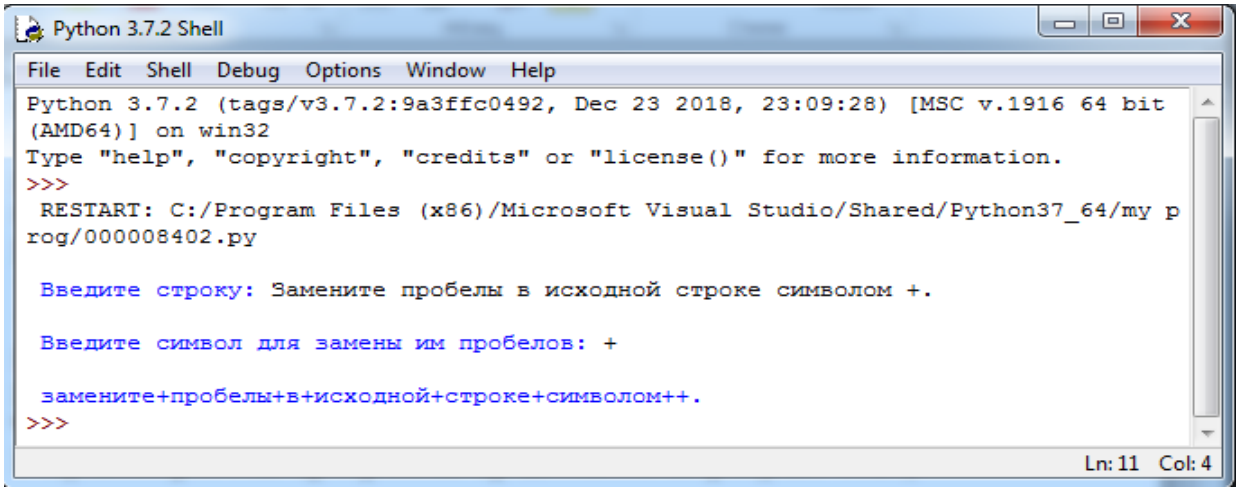
Шешімі. Join() әдісінің жұмысын түсіндірген кезде осындай есеп қарастырылған. Листингте код көрсетілген, өңделетін жол бекітілмеген, бірақ пернетақтадан енгізілген. Оны өңдеу әдістері өзгеріссіз қалды. Мұндай есепті **replace()** әдісін қолдана отырып шешуге болады.

Төменде есепті шешуге жауап беретін бағдарлама коды берілген.

```
stroka=input("\n Введите строку: ")
stroka1=stroka.lower()
spisok=list(stroka1)
simvol=input("\n Введите символ для замены им пробелов: ")
```

```
n=len(stroka1)
for i in range(0, n):
    if spisok[i]==" ":
        spisok[i]=simvol
stroka1="".join(spisok)
print("\n", stroka1)
```

Бағдарлама жұмысының нәтижесі 8.3-ші суретте көрсетілген.



Сурет 8.3 -Бастапқы жолдағы бос орындарды + белгісімен ауыстыру бағдарламасының нәтижесі

Жолдағы таңбаларды жою.

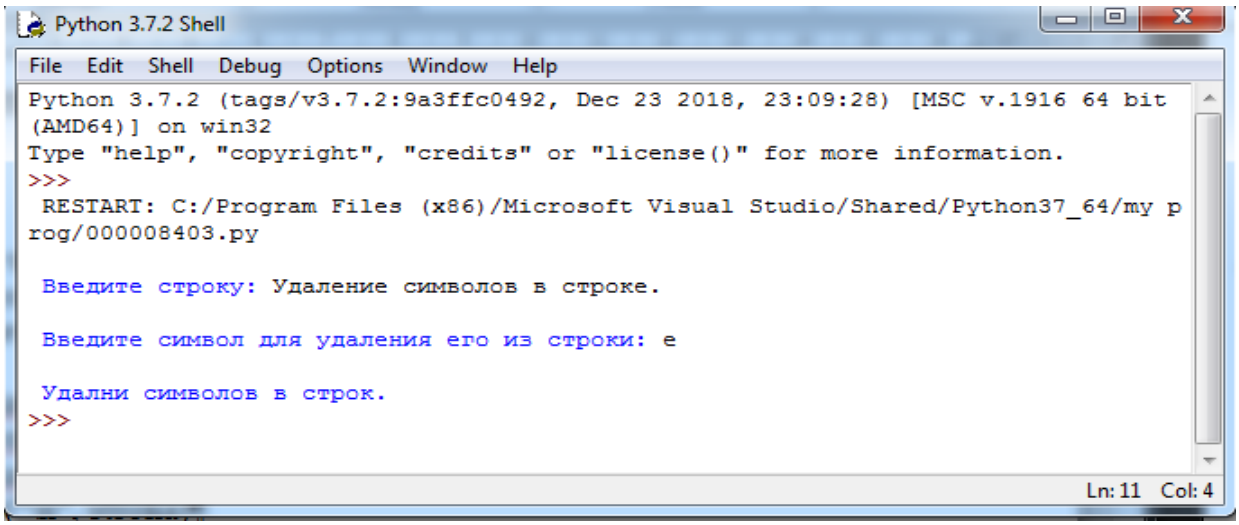
Есеп 8.3.3. Пайдаланушы **stroka** бастапқы жолын және **simvol** таңбасын енгізеді. Бастапқы жолда көрсетілген таңбаны жойыңыз.

Шешімі. Әдіс **replace()** әдісін қолдануға негізделген. Біз жойғымыз келетін таңбаны енгіземіз, содан кейін оны **replace()** әдісінде параметр ретінде қолданамыз. Әдісдегі екінші параметр бос тырнақшаларға тең деп аламыз.

Төменде есепті шешуге жауап беретін бағдарлама коды берілген.

```
stroka=input("\n Введите строку: ")
simvol=input("\n Введите символ для удаления его из строки: ")
stroka=stroka.replace(simvol, "")
print("\n", stroka)
```

Бағдарлама жұмысының нәтижесі 8.4-ші суретте көрсетілген.



Сурет 8.4- Бастапқы жолда енгізілген символды жою бойынша бағдарлама жұмысының нәтижесі

Жолға таңба қою.

Есеп 8.3.4. Пайдаланушы бастапқы **stroka** жолын енгізеді. Пайдаланушы пернетақтадан енгізетін **simvol** таңбасынан кейін бастапқы жолға **simvol1** таңбасын қосу керек.

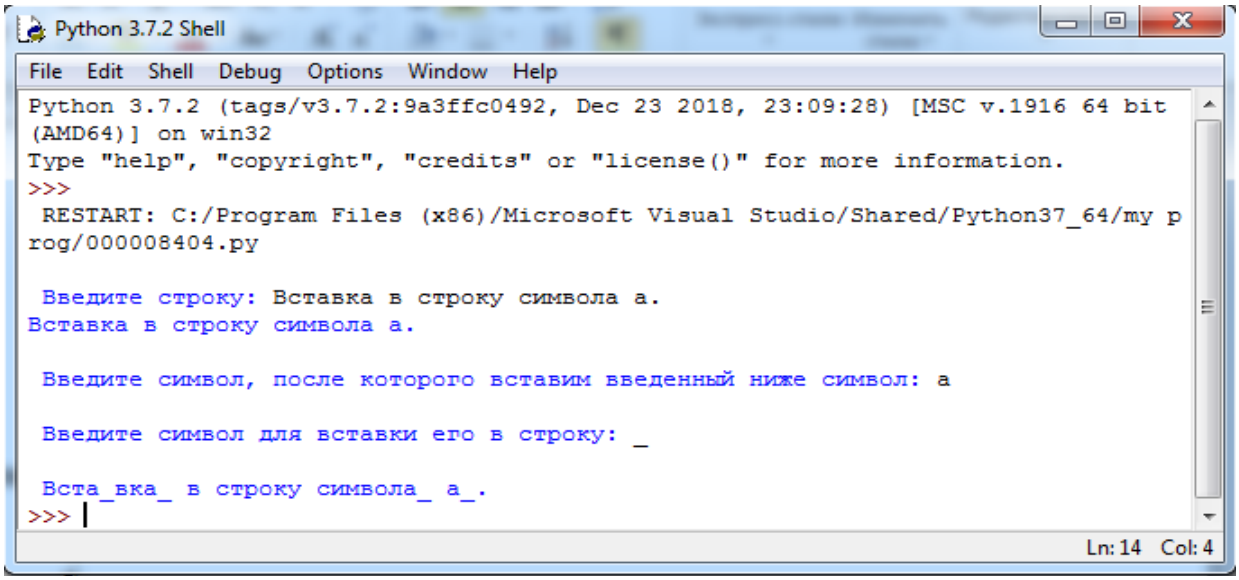
Шешімі. Берілген позицияға ішкі жолды енгізу **spisok.insert (i+l, simvol1),** операторының көмегімен жүзеге асырылады біз **list** функциясын қолдана отырып, жолды тізімге айналдырдық. **I** параметрін **for** операторымен циклде өзгерту жол бойынша жылжуды қамтамасыз етеді. Егер тізімнің келесі элементі табылған элементке тең болса **(if spisok[i]== simvol),** онда **insert** әдісі қолданылады. Бағдарламаның соңында, **join** әдісін қолдана отырып, кері түрлендіруді жасаймыз: тізім экранға шығарылатын жолға айналады. Жолдармен жұмыс істеу кезінде таңбаның жоғарғы немесе төменгі регистрде қандай күйде екенін ескеру қажет екенін есте ұстаған жөн.

Төменде есепті шешуге жауап беретін бағдарлама коды берілген.

```
stroka=input("\n Введите строку: ")
print(stroka)
spisok=list(stroka)
simvol=input("\n Введите символ, после которого вставим введенный ниже символ: ")
simvol1=input("\n Введите символ для вставки его в строку: ")
n=len(stroka)
k=0
for i in range(0, n):
    if spisok[i]==simvol:
        k+=1
for i in range(0, n+k):
    if spisok[i]==simvol:
        spisok.insert(i+1, simvol1)
```

```
stroka="".join(spisok)
print("\n", stroka)
```

Бағдарлама жұмысының нәтижесі 8.5-ші суретте көрсетілген.



Сурет 8.5 – Бастапқы жолға " а "символынан кейін енгізілген" _ " символын енгізу бойынша бағдарлама жұмысының нәтижесі

Топқа жататын таңбаны талдау.

Есеп 8.3.5. Пайдаланушы бастапқы **stroka** жолын енгізеді. Бастапқы жолдағы **symbol** және **simvol1** таңбаларының санын анықтаңыз. Пайдаланушы пернетақтадан **simvol1** және **simvol** таңбаларын енгізеді.

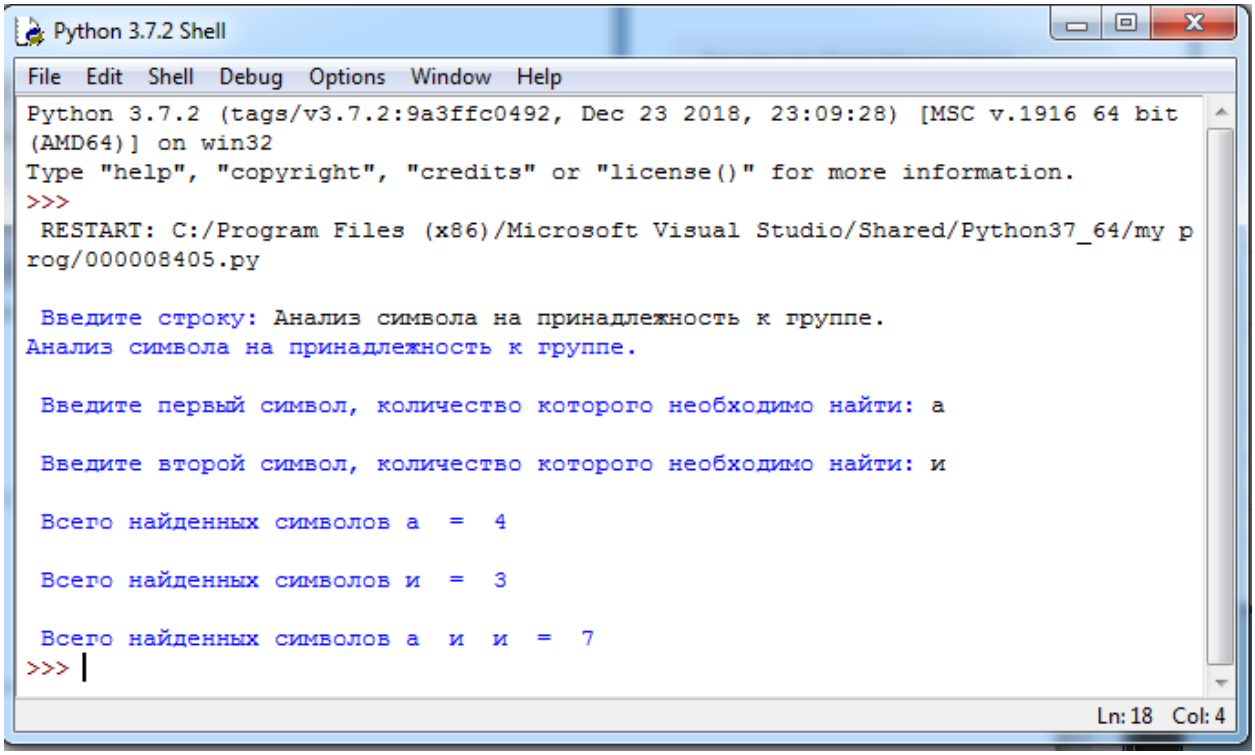
Шешімі. Бұл бағдарламада таңбалар жолын тізімге түрлендіріп, бастапқы жолда санын анықтағымыз келетін екі таңбаны енгіземіз. **For** операторымен циклде біз тізімнің әр элементін пайдаланушы енгізген таңбалармен салыстырамыз, егер шарт ақиқат болса, есептегіш бірге артады.

Төменде есептің шешуге жауап беретін бағдарлама коды берілген.

```
stroka=input("\n Введите строку: ")
print(stroka)
spisok=list(stroka)
simvol=input("\n Введите первый символ, количество которого необходимо найти: ")
simvol1=input("\n Введите второй символ, количество которого необходимо найти: ")
k1=0
k2=0
k=0
n=len(stroka)
for i in range(0, n):
    if spisok[i]==simvol:
```

```
            k+=1
            k1+=1
        if spisok[i]==simvol1:
            k+=1
            k2+=1
    print("\n Всего найденных символов", simvol," = ", k1)
    print("\n Всего найденных символов", simvol1," = ", k2)
    print("\n Всего найденных символов", simvol, " и ", simvol1, " = ", k)
```

Бағдарлама жұмысының нәтижесі 8.6-шы суретте көрсетілген.



Сурет 8.6 – Топқа жататын таңбаларды талдау бойынша бағдарлама жұмысының нәтижесі

Жол айналымы.

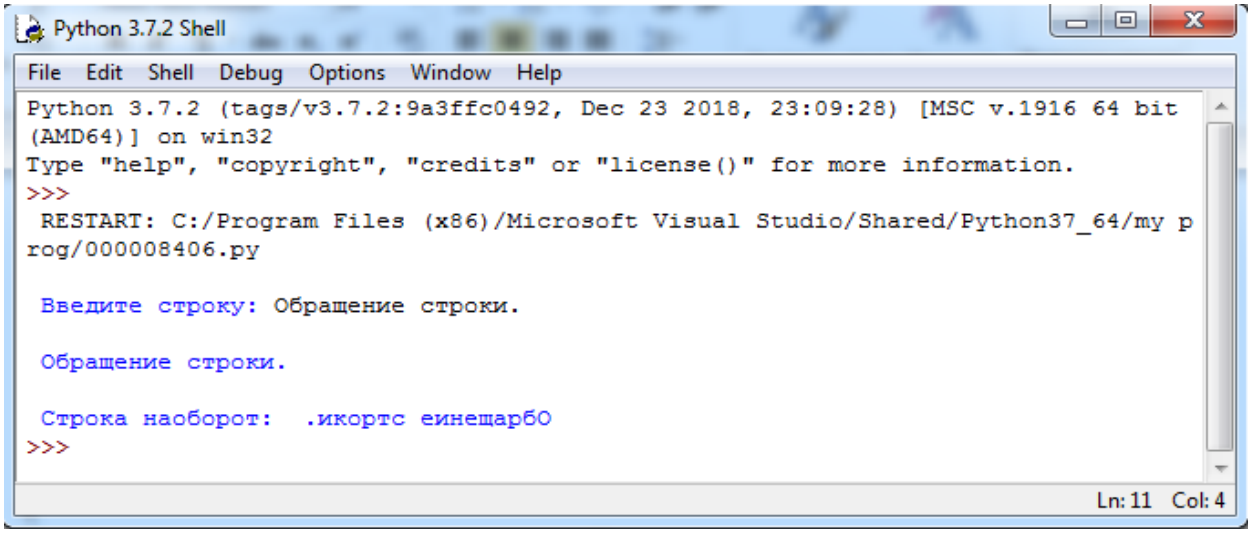
Есеп 8.3.6. Пайдаланушы бастапқы **stroka** жолын енгізеді. Жолды аудару керек, яғни таңбаларды кері ретпен жазу керек (соңғысы бірінші болады және керісінше).

Шешімі. Бұл есепті шешкен кезде тізімдерді өңдеу үшін **reverse** әдісін қолдануға болады, бірақ бұл бағдарламаның алгоритмі берілген жолдан таңбаларды кезекпен таңдауға және оларды жаңа жолдың басына жылжытуға негізделген. **Tmp** көмекші жолы бастапқыда бос болады. Циклды ұйымдастыру арқылы бастапқы жолдағы таңбалардың біріншісінен соңғысына дейін қарастырамыз. Олардың әрқайсысы жинақталған жолдың басына қосылады, оператор **tmp=stroka [i]+tmp** арқылы.

Төменде есепті шешуге жауап беретін бағдарлама коды берілген.

```
stroka=input("\n Введите строку: ")
print("\n", stroka)
tmp=""
n=len(stroka)
for i in range(0, n):
    tmp=stroka[i]+tmp
print("\n Строка наоборот: ", tmp)
```

Бағдарлама жұмысының нәтижесі 8.7-ші суретте көрсетілген.



Сурет 8.7 – Бағдарлама жұмысының нәтижесі

Алфавиттік таңдау.

Есеп 8.3.7. Алдын-ала белгіленген алфавит бар. Пайдаланушы бастапқы **stroka** жолын енгізеді. Алфавитте қолданылатын таңбаларды жолдан таңдап, оларды экранға шығарыңыз.

Шешімі . Бұл есепті шешкен кезде біз Python-да "тұрақты" сияқты ұғыммен танысамыз. Зерттелетін бағдарламалау тілінде *бас әріптермен* терілген айнымалы **тұрақты** деп аталады. Python-да тұрақтыларды қолдану ерекшелігі бар: көптеген бағдарламалау тілдерінен айырмашылығы, тұрақтыларды өзгертуге болады. Сондықтан тұрақты құрғаннан кейін пайдаланушы оның өзгермейтіндігін бақылауы керек.

Сонымен, бізде **ALFAVIT** тұрақтысында сақталатын алфавит деп аталатын белгілі бір таңбалар жиынтығы бар. Пайдаланушы жолды енгізеді және жолдың әр таңбасы үшін (**for** letter **in** stroka:) біз шартты тексереміз. Егер таңба алфавитте болса (**if** letter **in** ALFAVIT:), онда **tmp** айнымалысында біз **tmp+=letter** операторы арқылы алынған таңбаларды жинақтаймыз.

Төменде есепті шешуге жауап беретін бағдарлама коды берілген.

```
ALFAVIT="abcdgh"
stroka=input("\n Введите строку: ")
```

```
print("\n", stroka)
tmp=""
for letter in stroka:
    if letter in ALFAVIT:
        tmp+=letter
print(tmp)
```

Бағдарлама жұмысының нәтижесі 8.8-ші суретте көрсетілген.

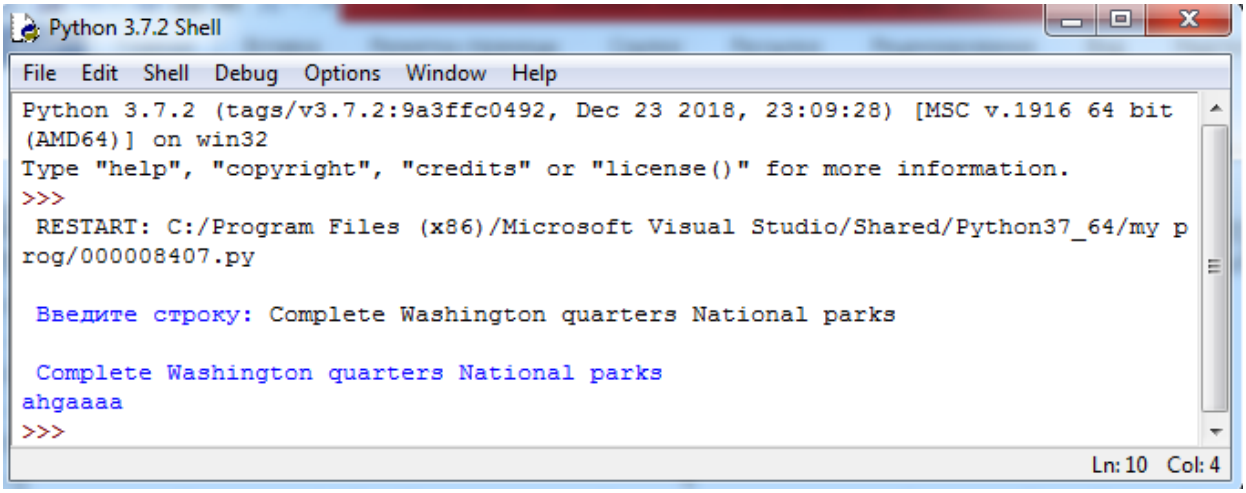


Рисунок 8.8 – Әріптік іріктеу бағдарламасының нәтижесі

Жол кесінділері .

Есеп 8.3.8. Бастапқы жолдан берілген бастапқы және соңғы мәндер арасында орналасқан таңбаларды алыңыз.

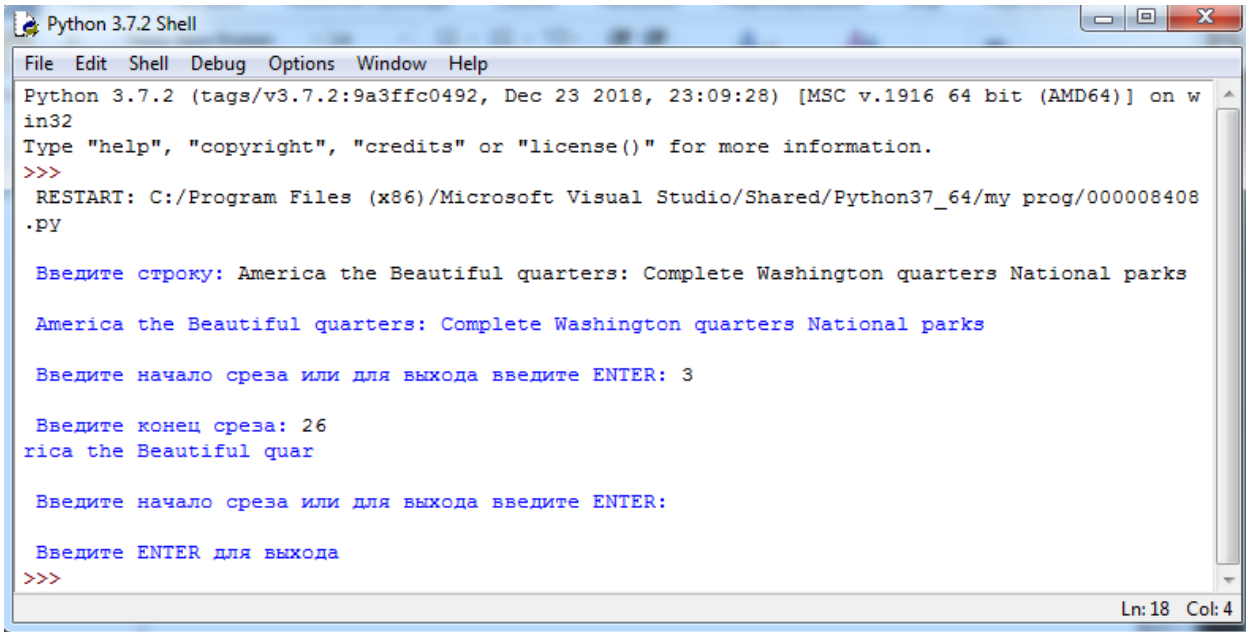
Шешімі . Кортеждермен жұмыс жасау тақырыбында "кесінді" ұғымымен таныстық онда кортеждің кесілуі элементтердің алдын-ала белгіленген бастапқы (a) және соңғы (b) позицияларының арасында орналасқан кортеж элементтерін шығару нәтижесінде алынған деп айтылды. Жол кесінділерінің жұмыс істеу механизмі кортеж кесінділерінең жұмыс істеу принциптеріне өте ұқсас. Кесінділерді жолдарға қолдана отырып, біз олардан кез-келген таңбаны немесе қатар орналасқан таңбалар тізбегін таңдай аламыз. Мұны істеу үшін бізге кесінділердің шекараларын көрсететін бастапқы және соңғы позициялары қажет.

Төменде есепті шешуге жауап беретін бағдарлама коды берілген.

```
stroka=input("\n Введите строку: ")
print("\n", stroka)
flag=None
while flag!="":
    flag=input("\n Введите начало среза или для выхода введите ENTER: ")
    if flag:
        flag=int(flag)
        kon=input("\n Введите конец среза: ")
        kon=int(kon)
```

```
        print(stroka[flag:kon])
input("\n Введите ENTER для выхода ")
```

Бағдарлама жұмысының нәтижесі 8.9-шы суретте көрсетілген.



Сурет 8.9 – Жолды кесінділердің бағдарламасының нәтижесі

8.4 Тәжірибелік жұмысты орындауға арналған тапсырмалар

Өз бетінше шешуге арналған тапсырмаларды және нұсқаға сәйкес жеке бір тапсырманы орындаңыз.

8.4.1 Өз бетімен шешуге арналған есептер .

8.4.1.1. Пернетақтадан енгізілген мәтінді шифрлайтын бағдарлама құрыңыз. Шифрлау процесі келесідей: он сан пернетақтадан енгізілген әр таңбаның ондық кодынан алынады. Алу нәтижесінде шыққан мән компьютер экранында көрсетілетін басқа таңбаның ондық коды ретінде көрсетіледі.

8.4.1.2. Қатар орналасқан «р» әріпінен тұратын ең ұзын тізбекті санау керек. Барлық леп белгілерін нүктелермен ауыстыру арқылы оны түрлендіріңіз

8.4.2 Жеке тапсырма нұсқа бойынша.

Нұсқа №	**Тапсырма**
1	Пернетақтадан енгізілген таңбалар тізбегі екілік сандар жүйесінде жазылған бүтін сан екенін тексеретін бағдарламаны жасаңыз

2	Пернетақтадан енгізілген жолдағы сөздердің орташа ұзындығын есептейтін бағдарламаны жасаңыз.
3	Сөз берілген . Оның палиндром екенін анықтаңыз (екі бағытта да бірдей оқылатын сөз, мысалы, "потоп").
4	Таңбалар жолы берілген. Жолдағы ең ұзын сөзді және бірдей ұзындықтағы сөздердің санын анықтаңыз.
5	Таңбалар жолы берілген. Ондық санның жазбасы болып табылатын сөздердің санын анықтаңыз.
6	Таңбалар жолы берілген. Одан барлық бос жолдарды жойыңыз.
7	Таңбалар жолы берілген. Сөз берілген. Жолдан осы сөзді жойыңыз .
8	Таңбалар жолы берілген. Бірінші және екінші нүктелер арасындағы ішкі жолды таңдаңыз.
9	Таңбалар жолы берілген. Ең қысқа және ең ұзын сөздердің ұзындығын анықтаңыз.
10	Таңбалар жолы берілген. Бір әріптен қанша сөз басталып, аяқталатынын анықтаңыз.
11	Таңбалар жолы берілген. Қанша сөз екі "с" әрпінен тұратынын анықтаңыз.
12	Таңбалар жолы берілген. Оның дұрыс жақшаның өрнегі екенін анықтаңыз. Тек дөңгелек жақшаларды қарастырыңыз.
13	Таңбалар жолы берілген. Қанша сөз үш "е" әрпінен тұратынын анықтаңыз.
14	Жолда тек сандар бар. Алда тұрған барлық нөльдерді жойыңыз
15	Таңбалар жолы берілген. Жолдағы тыныс белгілерінің санын есептеңіз.
16	Таңбалар жолы берілген. Жолдан барлық үтірлерді жойыңыз .
17	Таңбалар жолы берілген. Кейбір сөз берілген. Әр бос орыннан кейін оны қойыңыз .
18	Таңбалар жолы берілген. Жолда кездесетін сандардың қосындысын табыңыз.
19	Таңбалар жолы берілген. Жолдан барлық сандарды жойыңыз .
20	Таңбалар жолы берілген. Жолдан ең ұзын сөзді алып тастаңыз.
21	Таңбалар жолы берілген. Әр тыныс белгісінен кейін жолға бос орын қойыңыз.

22	Таңбалар жолы берілген. Бастапқы жол сөздерінен тұратын керісінше жазылған жолды қалыптастырыңыз.
23	Сандар мен латын әріптерінен жол берілген. Осы жолда қай әріптер көп екенін анықтаңыз, дауысты дыбыстар(A,E,I,O,I) немесе дауыссыз дыбыстар .
24	Берілген жолдан оның тырнақшада орналасқан бөліктерін алып тастаңыз (тырнақшалармен бірге).
25	Сөз берілген. А әріпін О әріпімен ауыстырыңыз. Егер бұл сөзде А әрпі болмаса, онда тиісті хабарламаны шығарыңыз.

8.5 Тәжірибелік жұмыс бойынша есеп дайындауға қойылатын талаптар

Тәжірибелік жұмыс туралы есепті дайындау кезінде келесі құрылым мен элементтердің кезектілігі ұсынылады:

- бастапқы бет;
- зертханалық жұмыстың атауы;
- зертханалық жұмыстың мақсаты;
- зертханалық жұмыстарға арналған жеке тапсырма (нұсқаларға сәйкес);
- жеке тапсырманы орындау туралы қысқаша түсініктемелер және есепті шешу алгоритмінің блок-схемасы;
- жеке тапсырмаға қажетті бағдарламалық код; - бағдарламаның нәтижелері; - тұжырымдар.

Зертханалық жұмыстарға арналған жеке тапсырма оқытушыдан алынған жеке тапсырманың толық мәтінін, жеке тапсырманы орындау алгоритмінің сипаттамасын және есепті шығарудың алгоритмінің блок-схемасын қамтиды.

Жеке тапсырмаға қажетті бағдарлама кодында студент әзірлеген бағдарлама кодының толық мәтіні болады.

Бағдарлама жұмысының нәтижелерінде әдетте бағдарламаның терезелерінің көшірмелері болады(скриншот).

8.6 Зертханалық жұмысты қорғауға арналған бақылау сұрақтары

8.6.1. Әріптік-цифрлық таңбалардың кодталуын қолдайтын кодтау қалай аталады? Оның құрылымы туралы айтыңыз.

8.6.2. Таңбалармен жұмыс істеудің негізгі функцияларын атап өтіңіз. Мысалдар келтіріңіз.

8.6.3. Жол таңбаларын әртүрлі пернетақта регистрлеріне түрлендіруге мүмкіндік беретін жолдармен жұмыс істеу әдістерін атаңыз..

8.6.4. Қандай әдіс жолды ішкі жолдарға бөлуге мүмкіндік береді? Оның синтаксисін жазыңыз.

8.6.5. Жолды тізімге түрлендіруге қандай әдіс жауап береді? Оның синтаксисін жазыңыз.

8.6.6. Негізгі жол алгоритмдеріне мысалдар келтіріңіз.

8.6.7. Жолды қалай кескіндеуге болады?