**Creating strings.py**

Ebben a feladatban olyan feladatot kell megoldanunk, ahol egy standard inputról érkező szöveg karaktereiből elő kell állítsuk azok összes permutációját, valamint meg kell adjuk a permutációk számát!

Kezdjük a programunkat azzal, hogy kérjük be a feldolgozandó adatokat!
Szükségünk lesz egy stringre, melyet jelöljünk `s`-sel:

```
s = input()
```

Ennek az s változónak a tartalmát adjuk át egy függvénynek, amit a következő lépésben írunk meg. Ez a függvény fogja elvégezni számunkra a munka érdemi részét.

Hozzuk létre a függvényt:

```
def letrehoz(szoveg):
```

Amennyiben ez a függvény bemenő értékként üres stringet kap, ne adjon vissza semmit, mivel nincs mit permutálni:

```
if len(szoveg) == 0:
    return ['']
```

Létrehozunk egy `tomb` nevű változót, mely egy lista elemeit tartalmazza majd, azaz a permutációkat. Természetesen, ez a változó az `if` ágon kívül kap helyet.

```
tomb = []
```

Egy ciklus segítségével menjünk végig a függvény által kapott `szoveg` változón, ami egy string. Ennek a stringnek van hossza (`len()` fgv. adja meg), ezért végig tudunk menni az első karkaterétől az utolsóig.

```
for i in range(len(szoveg)):
```

Raktározzuk el az `i` változó szerinti aktuális betűt:

```
        aktualis = szoveg[i]
```

A szöveg maradékát pedig tegyük el egy maradek nevű változóba.
A szöveg maradéka a `szoveg` változó `i`-ig tartó részéből (úgy, hogy az aktuális betű nem része már ennek), valamint a `szoveg` változó `i`-edik eleme utáni rész. Ennek a kettőnek az összefűzött része lesz a maradék szövegrész:

```
      maradek = szoveg[:i] + szoveg[i+1:]
```

Egy újabb ciklussal (az előző `for`-on belül!) `x` értékre újra meghívjuk a `letrehoz` nevű függvényünket, de itt már az eredeti szövegrész maradékát adjuk át neki, azaz a maradék változó tartalmát.

Az a fentiekhez hasonlóan addig meghívásra kerül `letrehoz` fgv., míg el nem fogy a (`maradek`) szöveg.

```
        for x in letrehoz(maradek):
            tomb.append(aktualis + x)
```

Ezzel a rekurzióval létrejön a maradék karakterlánc minden permutációja, melyeket hozzáfűzünk a `tomb` változóban tárolt társaihoz úgy, hogy első karkaterének az `aktualis` változóban tárolt betűt tesszük.

A rekurzív meghívások visszaépülését követően a létrehoz függvényünk visszatér a `tomb` változóban tárolt listával.

```
    return tomb
```

Az általunk létrehozott függényen kívül, az `osszes` nevű változóba tároljuk el a `letrehoz` függvénytől visszakapott listát, mely egy ismétlődéseket nem tartalmazó (`set`), rendezett lista(`sorted`).

```
minden = letrehoz(s)
osszes = sorted(set(minden))
```

Írjuk ki első kimeneti sorba a permutációk számát, ami az `osszes` változó hossza.

```
print(len(osszes))
```

Végezetül írjuk ki az egyedi elemeket az `osszes` nevű változóból.

```
for egyedi in osszes:
    print(egyedi)
```

Lássuk a teljes programunkat egyben:

```
def letrehoz(szoveg):

    if len(szoveg) == 0
    tomb = []

    for i in range(len(szoveg
        aktualis = szoveg[i]
        maradek = szoveg[:i] + szoveg[i+1:]

        for x in letrehoz(maradek):
            tomb.append(aktualis + x)
    return tomb

s = input()
minden = letrehoz(s)
osszes = sorted(set(minden))

print(len(osszes))
for egyedi in osszes:
    print(egyedi)
```