


本系统的测试样例全部都写在程序中，所以无输入。调用本程序，只需要运行(atom_count.py)

 atom_count.py

文件即可。

程序测试要求如下：

```
atom_count("He") == 1
atom_count("H2") == 2
atom_count("H2SO4") == 7
atom_count("CH3COOH") == 8
atom_count("NaCl") == 2
atom_count("C60H60") == 120
```


程序输出结果如下：

```
C:\Users\pc\AppData\Local\Programs\Py
atomcount(He)=1
atomcount(H2)=2
atomcount(H2SO4)=7
atomcount(CH3COOH)=8
atomcount(NaCl)=2
atomcount(C60H60)=120

Process finished with exit code 0
```

程序主要思路：

本程序,先在(calclex.py)

 calclex.py

文件中定义了分词规则。

```
tokens = (
    'NUMBER', # 识别分子式后的数量
    'SYMBOL', # 识别原子
)

# Regular expression rules for simple tokens
t_SYMBOL = (# 正则表达式识别化学元素
    r"C[laroudsemf]?|Os?|N[eaibdpos]?|S[icernbmg]?|P[drmtboau]?|"
    r"H[efogas]?|A[lrsgutcm]|B[eraik]?|Dy|E[urs]|F[erm]?|G[aed]|"
    r"I[nr]?|Kr?|L[iaur]|M[gnodt]|R[buhenaf]|T[icebmah]|"
    r"U|V|W|Xe|Yb?|Z[nr]")

def t_NUMBER(t):
    r'\d+'
    t.value = int(t.value)
    return t
```

然后再在 atom_count.py 使用 yacc 库对文法识别：

文法为:

分子式 : 更细分的分子式 + 分子
 | 分子

识别分子

分子 : 原子 + 数量
 | 原子

```
# 识别分子式
# 分子式 : 更细分的分子式 + 分子
#           | 分子
def p_species_list(p):
    '''species_list : species_list species
                    | species'''
    if(len(p)==3):p[0]=p[1]+p[2]
    else: p[0]=p[1]






# 识别分子
# 分子 : 原子 + 数量
#           | 原子
def p_species(p):
    '''species : SYMBOL NUMBER
                | SYMBOL'''
    if(len(p)==3):p[0]=p[2]
    else:p[0]=1
```

最后调用函数

```
def atom_count(ATOM):
    atomcount=parser.parse(ATOM)
    print(ATOM+" "+str(atomcount))
    return atomcount
```

最后得到输出。

注: (文件中其他文件均是调用完 atom_count.py, 自动生成的, 所以无关, 只需要将 (atom_count.py) 和(calclex.py)程序放在一个目录下运行即可。)

 __pycache__
 atom_count.py
 calclex.py
 parser.out
 parsetab.py