00.文字の逆順

自分の最初の解決策：

```
Reversed_str="stressed"
Reversed_str_list=list(Reversed_str)
flag=True
index=0
while flag:
    imp=Reversed_str_list[index]
    Reversed_str_list[index]=Reversed_str_list[(len(Reversed_str)-index-1)]
    Reversed_str_list[len(Reversed_str)-index-1]=imp
    index=index+1
    if index ==len(Reversed_str)/2:
        break
Reversed_str=''.join(Reversed_str_list)
print(Reversed_str)
```

実行結果：

```
desserts
```

模範解答：

```
str="stressed"
new_str=str[::-1]
print(new_str)
```

実行結果：

```
desserts
```

まとめ：
最初は文字列をリストに変換し、index の位置の内容と（len（Reversed _ str）-index-1）の内容をループで入れ替えることで、文字列の逆順序の結果を達成したいと思っていました。運行結果から見ると、確かに所期の目標を達成した。しかし、模範解答に比べてプロセスが複雑すぎる。
模範解答は python 言語のスライス方法を使用している。
例えば：
>>>a= '0123456'
>>>a[1:5]
'1234'
このようなコードの意味は、選択文字列 1～5 文字。コードが a[:]の場合は文字列のコピーを意味します
>>>a[1:5:2]
'13'
これは、文字列 1～5 文字のスライスで、ステップサイズが 2 です。
[:::-1]は、最初から最後までの文字列全体を表し、ステップサイズは-1、つまり最後の文字を順番に選択します。これにより文字列の逆順が実現されます。

01.パタトクカシーー

自分の最初の解決策：

```
str="パタトクカシーー"
str2=""
for i in range(len(str)):
    if i%2==0:
        str2=str2+str[i]
print(str2)
```

実行結果：

```
パトカー
```

模範解答：
```
s = 'パタトクカシーー'
print (s[::2])
```

実行結果：
```
パトカー
```

まとめ：
最初は i の値で取り出すべき文字を判定したいと思っていましたが、i が偶数の場合は取り出されるべき文字です。しかし、前の問題から分かるように、スライス方法を再使用することができます。[:::2]は、最初から最後までの文字列全体を表し、ステップサイズは 2。

02.「パトカー」＋「タクシー」＝「パタトクカシーー」

自分の最初の解決策：

```
str="shoe"
str2="cold"
new_str=""
if len(str)==len(str2):
    for i in range(len(str)):
        new_str=new_str+str[i]
        new_str=new_str+str2[i]
if len(str)<len(str2):
    for i in range(len(str)):
        new_str=new_str+str[i]
        new_str=new_str+str2[i]
    for i in range(len(str),len(str2)):
        new_str=new_str+str2[i]
if len(str)>len(str2):
    for i in range(len(str2)):
        new_str=new_str+str[i]
        new_str=new_str+str2[i]
    for i in range(len(str2),len(str)):
        new_str=new_str+str[i]
print(new_str)
```

実行結果：
```
パタトクカシーー
```

模範解答：
```
text1="パトカー"
text2="タクシー"
new_text=[s1+s2 for s1,s2 in zip(text1,text2)]
new_text="".join(new_text)
print(new_text)
```

実行結果：
```
パタトクカシーー
```

まとめ：
模範解答では、zip 関数が使用されています。zip 関数は、一連の反復可能なオブジェクトをパラメータとして受け入れ、オブジェクト内の対応する要素を 1 つの tuple（タプル）にパッケージ化し、それらの tuple（タプル）から 1 つの list（リスト）を構成して戻すことができます。リストを返すので、””.join 関数を呼び出して文字列に変換する必要があります。

03.円周率

解答：

```
PI="Now I need a drink, alcoholic of course, after the heavy lectures involving quantum mechanics."
normalized_str=PI.replace(",", "").replace(".", "")
words=normalized_str.split(" ")
print(words)
pi_list = [len(w) for w in words]
print(pi_list)
```

実行結果：

```
[3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9]
```

まとめ：

normalized_str=PI.replace(",", "").replace(".", "")

replace 関数を使用すると、文字列の", "."を削除することができます。

```
In [87]: PI="Now I need a drink, alcoholic of course, after the heavy
lectures involving quantum mechanics."
    ...: normalized_str=PI.replace(",", "").replace(".", "")

In [88]: normalized_str
Out[88]: 'Now I need a drink alcoholic of course after the heavy lectures
involving quantum mechanics'
```

normalized_str.split(" ") ； split 関数を使用して文字列を空白で区切る

```
In [91]: runfile('D:/学习文件/自然语言处理100练/untitled0.py', wdir='D:/学习
文件/自然语言处理100练')
['Now', 'I', 'need', 'a', 'drink', 'alcoholic', 'of', 'course', 'after',
'the', 'heavy', 'lectures', 'involving', 'quantum', 'mechanics']
```

split 関数は、他の文字を境界として文字列を分割することもできます，例えば：

```
In [93]: txt = "Google#Runoob#Taobao#Facebook"
    ...: x = txt.split("#")
    ...: print(x)
['Google', 'Runoob', 'Taobao', 'Facebook']
```

04.元素記号

解答：

```
Element="Hi He Lied Because Boron Could Not Oxidize Fluorine. New Nations Might Also Sign Peace Security Clause. Arthur King Can."
normalized_str=Element. replace(".", "")
words=normalized_str.split(" ")

Map_char=[1,5,6,7,8,9,15,16,10]

Element_result={}#define a set named Element_result

for i,word in    enumerate(words):
        if i+1 in Map_char:
            Element_result[i]=word[0]
        else:
            Element_result[i]=word[:2]
print(Element_result)
```

実行結果：

```
{0: 'H', 1: 'He', 2: 'Li', 3: 'Be', 4: 'B', 5: 'C', 6: 'N', 7: 'O', 8: 'F', 9: 'N', 10: 'Na', 11: 'Mi', 12: 'Al', 13: 'Si', 14: 'P', 15: 'S', 16: 'Cl', 17: 'Ar', 18: 'Ki', 19: 'Ca'}
```

まとめ：

enumerate()関数は、リスト、タプル、文字列などのトラバース可能なデータ・オブジェクトをインデックス・シーケンスに結合するために使用されます。また、データとデータ・カーソルをリストするために使用されます。

例えば：

```
seasons = ['Spring', 'Summer', 'Fall', 'Winter']

list(enumerate(seasons))

# output

[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
```

05.n-gram

解答：

```
def ngram(S, n):
    r = []
    for i in range(len(S) - n + 1):
        r.append(S[i:i+n])#The process range is [i,i+n)
    return r
s = 'I am an NLPer'
print (ngram(s.split(),2))
print (ngram(s,2))
```

実行結果：

```
単語:
[['I', 'am'], ['am', 'an'], ['an', 'NLPer']]
文字:
['I ', ' a', 'am', 'm ', ' a', 'an', 'n ', ' N', 'NL', 'LP', 'Pe', 'er']
```

まとめ：
　　解答にスライスを用いた方法では、関数を呼び出す際に入力したパラメータタイプ（リストまたは文字列）を選択することで、結果が単語 bi-gram か文字 bi-gram かを制御する

06.集合

自分の最初の解決策：

```
def ngram(S, n):
    r = []
    for i in range(len(S) - n + 1):
        r.append(S[i:i+n])#The process range is [i,i+n)
    return r

SetX=set(ngram('paraparaparadise',2))
SetY=set(ngram('paragraph',2))
Intersection_Set=SetX.intersection(SetY)
print(Intersection_Set)#積集合
Union_Set=SetX.union(SetY)
print(Union_Set)#和集合
Difference_Set=SetX.difference(SetY)
print(Difference_Set)#差集合
print('se' in SetX)
print('se' in SetY)
```

模範解答：

```
def ngram(S, n):
    r = []
    for i in range(len(S) - n + 1):
        r.append(S[i:i+n])
    return r
s1 = 'paraparaparadise'
s2 = 'paragraph'
st1 = set(ngram(s1, 2))
st2 = set(ngram(s2, 2))
print(st1 | st2)
print(st1 & st2)
print(st1 - st2)
print('se' in st1)
print('se' in st2)
```

実行結果：

```
{'ar', 'ap', 'ra', 'pa'}
{'se', 'ar', 'ag', 'ra', 'pa', 'di', 'gr', 'ap', 'ad', 'ph', 'is'}
{'se', 'di', 'ad', 'is'}
True
False
{'se', 'ar', 'ag', 'ra', 'pa', 'di', 'gr', 'ap', 'ad', 'ph', 'is'}
{'ar', 'ap', 'ra', 'pa'}
{'se', 'di', 'ad', 'is'}
True
False
```

まとめ：
結果的には、どちらの方法も予想される効果を実現することができます。したがって、和集合を求めるには st1 | st2，SetX.intersection(SetY)、交集合を求めるには st1 & st2，　SetX.union(SetY)、差集合を用いるには st1 - st2 SetX.difference(SetY)。

07. テンプレートによる文生成

解答：

```
def temperature(x,y,z):
    return str(x)+'時の'+str(y)+'は'+str(z)
x = 12
y = '気温'
z = 22.4
print (temperature(x,y,z))
```

実行結果：

```
In [10]: runfile('D:/学习文件/自然语言处理100练/untitled0.py', wdir='D:/学习文件/自然语言处理100练')
12時の気温は22.4
```

まとめ：

関数では、文字列を返す必要があり、入力したパラメータタイプが不確定な場合、パラメータを使用するときに文字列形式に強制的に変換する必要があることに注意してください（str（））。

08. 暗号文

解答：

```
def cipher(str):
    new = []
    for i in str:
        if 97 <= ord(i) <= 122:
            i = chr(219 - ord(i))
        new.append(i)
    return ''.join(new)
s = 'I am an NLPer'
print(cipher(s))
```

実行結果：

```
In [12]: runfile('D:/学习文件/自然语言处理100练/untitled0.py', wdir='D:/学习文件/自然语言处理100练')
I zn zm NLPvi
```

まとめ：

ord()関数は Python のライブラリ関数で、与えられた文字値から数値値を取得するために使用され、文字を受け入れ、文字を整数に変換するために使用される整数、すなわち a の ASCII 値を取得するために使用される整数を返します。

chr（）は range（256）内の（つまり 0～255）整数をパラメータとし、ASCII 符号値に対応する文字を返します。

09. Typoglycemia

解答：

```
import random
s = 'I couldn't believe that I could actually understand what I was reading : the phenomenal power of the human mind .'
ans = []
text = s.split()
for word in text:
    if (len(word)>4):
        mid = list(word[1:-1])
        random.shuffle(mid)
        word = word[0] + ''.join(mid) + word[-1]
        ans.append(word)
    else:
```

```
            ans.append(word)
print (' '.join(ans))
```

実行結果：

```
In [18]: runfile('D:/学习文件/自然语言处理100练/untitled0.py', wdir='D:/学习文件/自然语言处理100练')
I clnu'odt bvieele that I colud aluctlay undtrsnaed what I was rdnaieg : the ponamehenl power of the hmuan mind .
```

まとめ：

random.shuffle()は、リスト内の要素を順序を乱すために使用されます。この方法を使用すると、新しいリストは生成されませんが、元のリストの順序を乱すだけです。