Python 問題

＜文字列の置き換え＞

`replace()`を使わない方法で文字列を置き換えることを考える。

まず、必要な文字列を探すために `find()`を使う。

`find()`メソッドは、文字列において特定の部分文字列が最初に現れる位置を見つける。指定された部分文字列が見つかった場合、その**開始インデックス**を返す。もし部分文字列が文字列に存在しない場合は、「-1」 を返す。

```
文法:
  str.find(sub[, start[, end]])
```

・str：検索を行う文字列。
・sub：文字列内で検索する部分文字列。
・start：検索を開始するインデックス（省略可）。指定しない場合は 0 から検索を開始。
・end：検索を終了するインデックス（省略可）。指定しない場合は文字列の末尾まで検索。

例：

```
text = ”Hello world”
index = text.find(”world”)
print(index)　#  出力: 6
```

この場合には、文字は以下のように配列に入る。インデックス番号は先頭が「0」。
したがって、「w」の文字はインデックス番号「6」となる。

H	e	l	l	o		w	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

◆部分抽出

リスト abc[]の 0～5 までを切り出すには、

abc[0,6]

と記述する。この場合、2 つ目のインデックスよりひとつ少ない数になる。

この「0：6」の「0」は、リストの先頭からの場合、省略可能。

abc[:6]

```
#問題 1
st = "今日は 3 時間勉強した"
ix = st.ア("3")
print(イ+"5"+st[ix+1:len(st)])
```

正解：ア：find

イ：st[0:ix]

イは、開始の要素を省略して st[:ix]でも可

◆補足　replace()を使うと

文字列内の特定の部分文字列を別の文字列に置き換える。この場合、新しい文字列を返し、元の文字列は変更されない。

文法:

str.replace(old, new[, count])

- str：置き換えを行う文字列。
- old：置き換える対象の部分文字列
- new：old を置き換える新しい文字列
- count：置き換えを行う回数（省略可）。指定しない場合は、すべての old が置き換えらる

```
text = "Hello world"
replaced_text = text.replace("world", "Python")
print(replaced_text)  # 出力: Hello Python
```

＜choice を使ってランダムな文字列生成＞

乱数を発生させるには random モジュールを使う。Python では random モジュールは標準ライブラリのひとつであるので import で読み込む。

0 から 9 まで、ランダムな数の生成コード：

```
import random

number = random.randint(0, 9)
print(number)
```

randint()は関数名の通り「int 数」を返します。
では、小数点を含む数を返すには？と考えますが・・・。少数を返す関数は、

uniform(a, b)

です。予想と全く違った形です。

その他にも、ある範囲で乱数を返す randrange()もあります。

randrange（最初の数、最後の数、ステップ数）

◆choice：ランダムに要素を取り出す

choice は、与えられた「シーケンス（リスト、タプルなど)」からランダムに要素を選択して返す。choice 関数は、random モジュールの一部であるので「import random」する必要がある。

```
import random

sequence = [1, 2, 3, 4, 5]
element = random.choice(sequence)
print(element)
```

◆string.digits

Python で文字を扱う string モジュール。したがって、使う前に「import string」をする必要がある。この中で、string.digits は、0 から 9 までの数字を返します。

他にも、

・string.ascii_letters：大文字小文字のアルファベットを返す
・string.punctuation：ascii に含まれる記号を返す

!”#$%&’()*+,-./:;<=>?@[¥]^_`{|}~

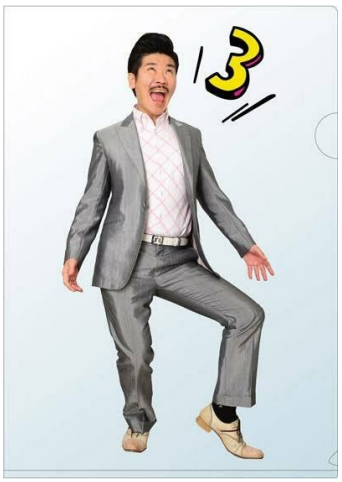
4 桁の数字をランダムに作成

```
#問題 2
from random import choice
import ア

nums = string.digits
print(イ (nums) + イ(nums) + イ(nums) +イ(nums))
```

正解：ア：string
　　イ：choice

＜分岐と繰り返し＞



「世界のナベアツ」2004 年頃、「3 の倍数と 3 が付く数字のときだけアホになります」と言うネタで大ブレイク。現在は、なんと、落語家に転身している。

この「3 倍数で」というのは元々海外にあった遊び。3 の倍数の時「fizz」といい、5 の倍数の時「buzz」という遊び。

◆for 文（in range()）

```
for i in range(1,16):
```

・for：for ループを示し、i はその中の変数

・in range():文字通り「範囲の中で」と言う意味。() には、開始値と終了値が設定されるが、終了値は「未満」(less than) の値であることに注意。したがって、ここでは「1 から 15」までになる。

＜応用＞for i in rage で文字を扱う場合

range 関数自体は数値を扱うので、そのままでは文字を扱うことができない。この場合、文字列の長さに対して range のインデックスで文字にアクセスする手法をとる。

```
s = "Hello"
for i in range(len(s)):
    print(s[i])
```

ここでは、変数 s に格納された長さ文だけ、出力をくりかえす。print 文なので、結果は書く文字改行されて表示される。

```
#問題 3
for i in ア(1, 16):
    if:  イ
        if(i % 5 ==0):
            print('fizz-buzz')
        else:
            print('fizz')
    elif i % 5 ==0:
        print('buzz')
    else:
        print(i)
```

ちなみに、改行されないためには end=’’と前の文字列の最後に追加される。

正解：ア：range

イ：i % 3==0