

A long-exposure photograph of the Chicago skyline at night, viewed from the water. The city lights are reflected on the calm surface of the lake. On the left, a small lighthouse structure is visible on a pier. The sky is a deep blue, and the water is dark with some ripples.

第二回
23/06/19

ITs
プログラミング教室
Python基礎編

1. リスト操作
2. 論理演算子
3. 条件分岐
4. 繰り返し処理

演習0)

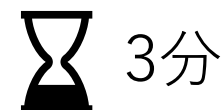
kawamura, aoki, saiを要素とするリストを作成してjohoという名前の変数に代入し、そのリストを出力してください。

次にfujiwaraをリストに追加し、aokiを削除してください。

最後にjohoリストを出力してください。

出力:

1. ['kawamura', 'aoki', 'sai']
2. ['kawamura', 'sai', 'fujiwara']



解答例

✓
0
秒



```
joho = ['aoki', 'sai', 'kawamura']  
print(joho)  
joho.append('fujiwara')  
joho.remove('aoki')  
print(joho)
```

```
['aoki', 'sai', 'kawamura']  
['sai', 'kawamura', 'fujiwara']
```



johoは後で使うからコードを実行しておこう！！

✓
0
秒



```
numbers = [4, 2, 3, -4, 8, 0, 9]
numbers.sort()
print(numbers)
```

[-4, 0, 2, 3, 4, 8, 9]

✓
0
秒



```
numbers = [4, 2, 3, -4, 8, 0, 9]
numbers.sort(reverse=True)
print(numbers)
```

[9, 8, 4, 3, 2, 0, -4]

リスト内の中身を並び替える場合は
.sort()を使う。
標準では昇順に並び替えられるけど、
降順にしたいときはreverse=True
にする。



✓
0
秒

```
languages = ["Japanese", "Chinese", "Indonesian", "English", "Arabic", "German"]  
languages.sort()  
print(languages)
```

['Arabic', 'Chinese', 'English', 'German', 'Indonesian', 'Japanese']

✓
0
秒

```
[7] languages = ["Japanese", "Chinese", "Indonesian", "English", "Arabic", "German"]  
languages.sort(reverse=True)  
print(languages)
```

['Japanese', 'Indonesian', 'German', 'English', 'Chinese', 'Arabic']

文字列(A~Z)が入っているときは
アルファベット順



✓
0
秒



```
numbers = [4, 2, 3, -4, 8, 0, 9]

sorted_nums = sorted(numbers)

print(numbers)
print(sorted_nums)
```

```
[4, 2, 3, -4, 8, 0, 9]
[-4, 0, 2, 3, 4, 8, 9]
```



ソートしたリストを別の変数に代入する場合は`sorted()`を使う。

このとき、もともとの変数`numbers`はソートされないから注意！！



✓
0
秒

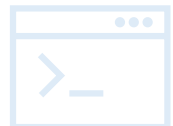


```
languages = ["Chinese", "English", "Indonesian", "Japanese"]  
languages.reverse()  
  
print(languages)
```

```
['Japanese', 'Indonesian', 'English', 'Chinese']
```



リストの中身を逆順にしたいときは
`reverse()`を使う。



演習1.1)

先ほど作成した**joho**というリストに対して、アルファベット逆順にしたものを出力してください。

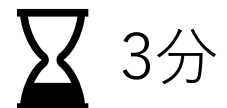
ただし、もとのリストの順番は保持するようにしてください。（代入する変数は**joho_sorted**とします。）

Joho , joho_sortedそれぞれのリストを出力してください。

出力:


```
['kawamura', 'sai', 'fujiwara']
```

```
['sai', 'kawamura', 'fujiwara']
```






解答例

✓
0
秒



```
joho_sorted = sorted(joho)
joho_sorted.reverse()
print(joho)
print(joho_sorted)
```

```
['kawamura', 'sai', 'fujiwara']
['sai', 'kawamura', 'fujiwara']
```



✓
0
秒

```
[34] joho_sorted = sorted(joho, reverse=True)
print(joho)
print(joho_sorted)
```

```
['kawamura', 'sai', 'fujiwara']
['sai', 'kawamura', 'fujiwara']
```



sorted()内でreverse=Trueを指定しても処理は同じ

0 1 2 3

✓
0
秒

```
[1] foods = ["サラダ", "ハンバーガー", "ポテト", "ナゲット"]  
     print(foods[1:3])
```

```
['ハンバーガー', 'ポテト']
```

リスト内の指定した範囲の要素を取得
したいときは`list[1:3]`のように記述する。

`[1:3]`は、要素の1番目から3番目の直前
までの範囲の要素を含む要素を取って
くる。



✓
0
秒 [2] foods = ["サラダ", "ハンバーガー", "ポテト", "ナゲット"]
print(foods[1:])

['ハンバーガー', 'ポテト', 'ナゲット']

[1:]は1番目から
それ以降全部

✓
0
秒 [3] foods = ["サラダ", "ハンバーガー", "ポテト", "ナゲット"]
print(foods[:3])

['サラダ', 'ハンバーガー', 'ポテト']

[:3]は0番目から
3番目の直前まで

✓
0
秒 [4] foods = ["サラダ", "ハンバーガー", "ポテト", "ナゲット"]
print(foods[-2:])

['ポテト', 'ナゲット']

リストの最後から
2番目以降

後ろから数えるときは1、2、3…のように数える



✓
0
秒



```
foods = ["サラダ", "ハンバーガー", "ポテト", "ナゲット"]  
menu = foods[:]  
print(menu)
```

```
['サラダ', 'ハンバーガー', 'ポテト', 'ナゲット']
```

リストの要素すべてを取得するときは
[:]と記述する。



```
✓ [10] list_a = [1, 2, 3, 4, 5]  
0  list_b = list_a  
秒  
  
list_b.remove(5)  
print(list_b)
```

```
[1, 2, 3, 4]
```

```
✓ [10] print(list_a)  
0  
秒
```

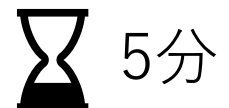
```
[1, 2, 3, 4]
```

リストを別の変数に代入してremove()すると、もともとのリストからも削除されてしまう！！



リスト `stationeries` の最初から3番目までの3つの要素を出力してください。さらに、`stationeries` と同等の `my_stationeries` というリストを生成し、その後、最後の要素を削除し、かつ、最初の要素として `stapler` を追加してください。最後に、`stationeries` と `my_stationeries` の内容を画面に出力してください。この際、これらのリストの内容は互いに異なることを確認してください。

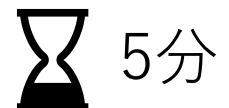
```
stationeries = ["pen", "pencil", "eraser", "ruler", "highlighter", "calculator",  
"sharpener", "compass", "cutter", "marker"]
```



解答例

```
stationeries = ["pen", "pencil", "eraser", "ruler", "highlighter", "calculator", "sharpener", "compass", "cutter", "marker"]
print(stationeries[:3])
my_stationeries = stationeries[:]
my_stationeries.pop()
my_stationeries.insert(0, 'stapler')
print(stationeries)
print(my_stationeries)
```

```
['pen', 'pencil', 'eraser']
['pen', 'pencil', 'eraser', 'ruler', 'highlighter', 'calculator', 'sharpener', 'compass', 'cutter', 'marker']
['stapler', 'pen', 'pencil', 'eraser', 'ruler', 'highlighter', 'calculator', 'sharpener', 'compass', 'cutter']
```



✓
0
秒



左と右が等しいならTrue
1 == 1



True

✓
0
秒

[2] 1 == 2

False

✓
0
秒



1 != 2

True

論理演算子の場合
帰ってくるものは
True,Falseどちらか

!= は異なるならTrue



```
▶ age = 19  
print(age < 21)  
print(age <= 21)  
print(age > 19)  
print(age >= 19)
```

```
⦿ True  
True  
False  
True
```

<, > の場合は比較する値を含まない。

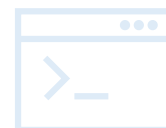
<=, >= の場合は比較するものを含む。



```
[ ] name = "さいはると"  
  
    name == "さいはると"
```

True

文字列の場合も比較できる。



✓
0
秒



```
age = 19
```

```
if age >= 20:
```

```
    print("お酒を飲めます")
```

```
else:
```

```
    print("お酒は飲めません")
```

お酒は飲めません

if 条件式:

条件を満たす場合の処理

else:

条件を満たさない場合の処理



✓
0
秒

```
[8] age = 10

if age < 7:
    print(' 料金は500円です')
elif age <= 12:
    print(' 料金は700円です')
else:
    print(' 料金は1200円です')
```

料金は700円です

複数の条件がある
場合はelifを使う

✓
0
秒

```
▶ drink = "coffee"
if drink != "soda":
    print("Hot or cold?")
```

Hot or cold?

文字列を用いても
条件式を記述できる。



✓
0
秒



```
# どっちもTrueならTrue  
print(1 == 1 and 2 == 3)
```

```
# どっちかがTrueならTrue  
print(1 == 1 or 2 == 3)
```

False

True

and は～かつ～

or は～または～



✓
0
秒

```
[12] drink = "beer"  
     age = 19  
  
     print(age < 20)  
     print(drink == "beer")  
     print(age < 20 and drink == "beer")  
  
     if age < 20 and drink == "beer":  
         print("You cannot order this drink.")
```

```
True  
True  
True  
You cannot order this drink.
```

論理演算子を用いてint型とstr型
両方比較できる



✓
0
秒

```
[13] order = ["french fries", "beer", "ice cream", "salad"]  
      print("beer" in order)  
  
      if "beer" in order:  
          print("We need to verify your age.")
```

True
We need to verify your age.

○ in □ は
□の中に○が
含まれているなら
True

✓
0
秒

```
▶ order = ["french fries", "beer", "ice cream", "salad"]  
   print("beer" not in order)
```

False

○ not in □は、inの逆で、
□の中に○が含まれてい
ないなら
True



以下のコードセルの二行目には0から80までの整数を発生させるコードが記述されています。この値はage（人の年齢）に格納されています。年齢が2歳に満たない場合は「baby」、2歳以上4歳未満は「toddler」、4歳以上で13歳未満は「kid」、13歳以上で20歳未満は「teenager」、20歳以上で65歳未満は「adult」、65歳以上は「elder」という文字列を出力してください。

✓
0
秒



```
import random  
age = random.randint(0, 80)  
print(age)
```

33



⌚ 5分

解答例

✓
0
秒



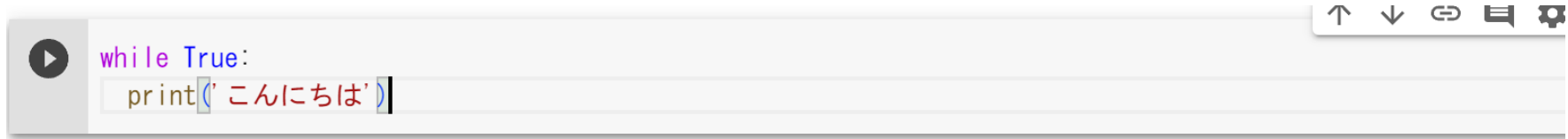
```
import random
age = random.randint(0, 80)
print(age)
if age < 2:
    print('baby')
elif age < 4:
    print('toddler')
elif age < 13:
    print('kid')
elif age < 20:
    print('teenager')
elif age < 65:
    print('adult')
else:
    print('elder')
```



21
adult



5分



```
while True:  
    print('こんにちは')
```

while 条件式:
 処理

と書くと、条件式がTrueの間はくり返し 上のやつだと、条件式の部分がTrueなので、常に繰り返しとなる

このように記述すると…

こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
…

と無限に出力される



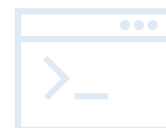
✓
0
秒



```
number_now = 0
while True:
    print(number_now)
    number_now += 1
    if number_now > 5:
        break
```

0
1
2
3
4
5

breakで繰り返し処理
を抜け出す。



✓
0
秒



```
number_now = 0

while number_now < 20:
    number_now += 1
    if number_now % 2 == 0:
        continue
    print(number_now)
```

1
3
5
7
9
11
13
15
17
19

continue -> それ以降の処理を行わずに繰り返し処理の先頭へ行く

number_nowが2で割り切れる
(偶数) の場合は出力せずに
continueしている。



✓
0
秒

```
[4] foods = ["salad", "hamburger", "french fries", "fried chicken"]  
for food in foods:  
    print(food)
```

salad
hamburger
french fries
fried chicken

for ○ in □

は□の中にある要素を一つずつ
取り出して○に代入する作業を
要素の数だけ繰り返す。

✓
0
秒



```
foods = ["salad", "hamburger", "french fries", "fried chicken"]  
for food in foods:  
    print("Ordering: {}".format(food))
```

Ordering: salad
Ordering: hamburger
Ordering: french fries
Ordering: fried chicken



✓
0
秒



```
foods = ["salad", "hamburger", "french fries", "fried chicken"]  
for food in foods:  
    print(food)  
print('終了')
```



```
salad  
hamburger  
french fries  
fried chicken  
終了
```

for文の処理が終わってからその下の処理が行われる。

for food in foods:

 print(food)

print('終了') のように繰り返し処理をするときは
インデント（空白）を空ける。



0
秒

```
[7] foods = ["salad", "hamburger", "french fries"]  
     print(foods)
```

```
File "<ipython-input-7-8dcffdd99636>", line 2  
    print(foods)  
    ^
```

IndentationError: unexpected indent

SEARCH STACK OVERFLOW

インデントが合っていないよ！！！！

0
秒

```
▶ foods = ["salad", "hamburger", "french fries"]  
  for food in foods  
    print("Ordering: {}".format(food))
```

```
File "<ipython-input-8-4bc716ff6e3a>", line 2  
    for food in foods  
    ^
```

SyntaxError: expected ':'

「:」を忘れてない??



0
秒



```
animals = ["dog", "cat", "hamster"]  
for animal in animals:  
print("I like {}".format(animal))
```

File "<ipython-input-9-3dc13a297765>", line 3

```
    print("I like {}".format(animal))  
    ^
```

IndentationError: expected an indented block after 'for' statement on line 2

SEARCH STACK OVERFLOW

インデントのつけ忘れは特にミスしやすい！！



✓
0
秒

```
[10] list(range(1, 5))
```

```
[1, 2, 3, 4]
```

range()は整数の順の生成ができる

✓
0
秒

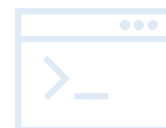


```
for number in range(1, 5):  
    print(number)
```



```
1  
2  
3  
4
```

range(始める値,指定した値の直前)





```
num_list1 = list(range(0, 51, 2))  
num_list2 = list(range(0, 51, 5))
```

```
print(num_list1)  
print(num_list2)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50]  
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

range(始める値,指定した値の直前,値の増加量) も指定できる



✓
0
秒

```
[13] cubes = []  
     for num in range(10):  
         cube_num = num ** 3  
         cubes.append(cube_num)  
     print(cubes)
```

[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]

✓
0
秒

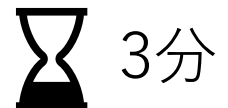
```
▶ cubes = [num ** 3 for num in range(10)]  
print(cubes)
```

[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]

上と下どちらも処理は同じ。
下の記述を
リスト内包表記という。



for および range() を利用して，1から100までの整数を合計して出力するためのプログラムを書いてください．



解答例

✓
0
秒

```
[15] sum = 0
      for num in range(1, 101):
          sum += num
      print(sum)
```

5050

range()に値を一つだけ入れた場合指定した値の直前まで整数を出力する。



```
sum = 0
```

```
for i in range(101):
    sum += i
```

```
print(sum)
```



5050



for および range() を利用して，1以上の偶数を小さい順に50個出力するプログラムを書いてください．数値は1行に1個ずつ出力してください．



解答例

✓
0
秒



```
for i in range(1, 101):  
    if (i % 2) == 0:  
        print(i)
```



2
4
6
8
10
12
14
16
18
20
22
24
26
28

インデントに注意！！



✓
0
秒



```
alcohol_list = ["beer", "wine", "sake"]

age = 12
orders = ["soda", "beer", "pasta", "chicken nugget", "wine"]

for item in orders:
    if item in alcohol_list:
        print("{} is not added to your order".format(item.title()))
    else:
        print("Adding {} to your order.".format(item))

print("Order completed.")
```

```
Adding soda to your order.
Beer is not added to your order
Adding pasta to your order.
Adding chicken nugget to your order.
Wine is not added to your order
Order completed.
```



以下のコードセルには **current_usres** と **new_users** という2つのリストが定義されています. **new_users** の3つの要素のそれぞれが, **current_users** に含まれている場合は

Your username is not available.,

含まれていない場合は

You can use this string.

の文字列を表示するプログラムを完成させてください.

このときに, 英字の大文字と小文字は区別しないでください.

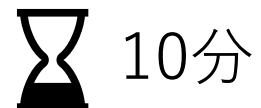
例えば, John と JOHN は同じ文字列とみなしてください.

```
current_users = ["Yamada", "Suzuki", "Matsui", "Ochiai", "Nagashima", "Harimoto"]
new_users = ["SUZUKI", "Nagashima", "Maeda"]
```

大文字と小文字の区別をどうすればよいか
自分で調べてみよう！！



ヒント:どっちのリストも小文字または
大文字に統一できれば…?



10分

解答例

```
current_users = ["Yamada", "Suzuki", "Matsui", "Ochiai", "Nagashima", "Harimoto"]
new_users = ["SUZUKI", "Nagashima", "Maeda"]

current_users = [current_user.lower() for current_user in current_users]
new_users = [new_user.lower() for new_user in new_users]

for new_user in new_users:
    if new_user in current_users:
        print("Your username is not available.")
    else:
        print("You can use this string.")
```

Your username is not available.
Your username is not available.
You can use this string.



lower()は文字列を小文字に変換する。

- (1) 0から100までの整数を出力するプログラムを作ってください。ただし、3の倍数の時はその数字の代わりにfizz, 5の倍数の時はbazz, 両方の倍数の時はfizzbazz と出力してください。
- (2) 整数のみが格納されている適当なリストに対し、そのリストを小さい順に並び変えた新しいリストを作成して出力してください。ただし、.sortやsorted, は使わないでください。（「選択ソート アルゴリズム」で検索）
- (3) 任意の自然数を素因数分解して得られた因数をリストで出力してください。

