Lists. Operators and Functions on Lists.

Нека да разгледаме задачата с книгите.

1. Всяка книга се моделира чрез променлива.
2. Ако искаме да добавим нова книга, трябва да добавим нова променлива.
3. За да знаем крайният брой на книгите, трябва ръчно да увеличаваме брояч.
4. На задачата с оптималните покупки, прилагаме алгоритъм "на око"

Нека да разгледаме задачата с най-голямото и най-малкото число от 3 цифри.

1. Всяка цифра е променлива
2. Имаме между 6 и 12 if-а
3. Ако цифрите станат 4, ние ще се откажем.

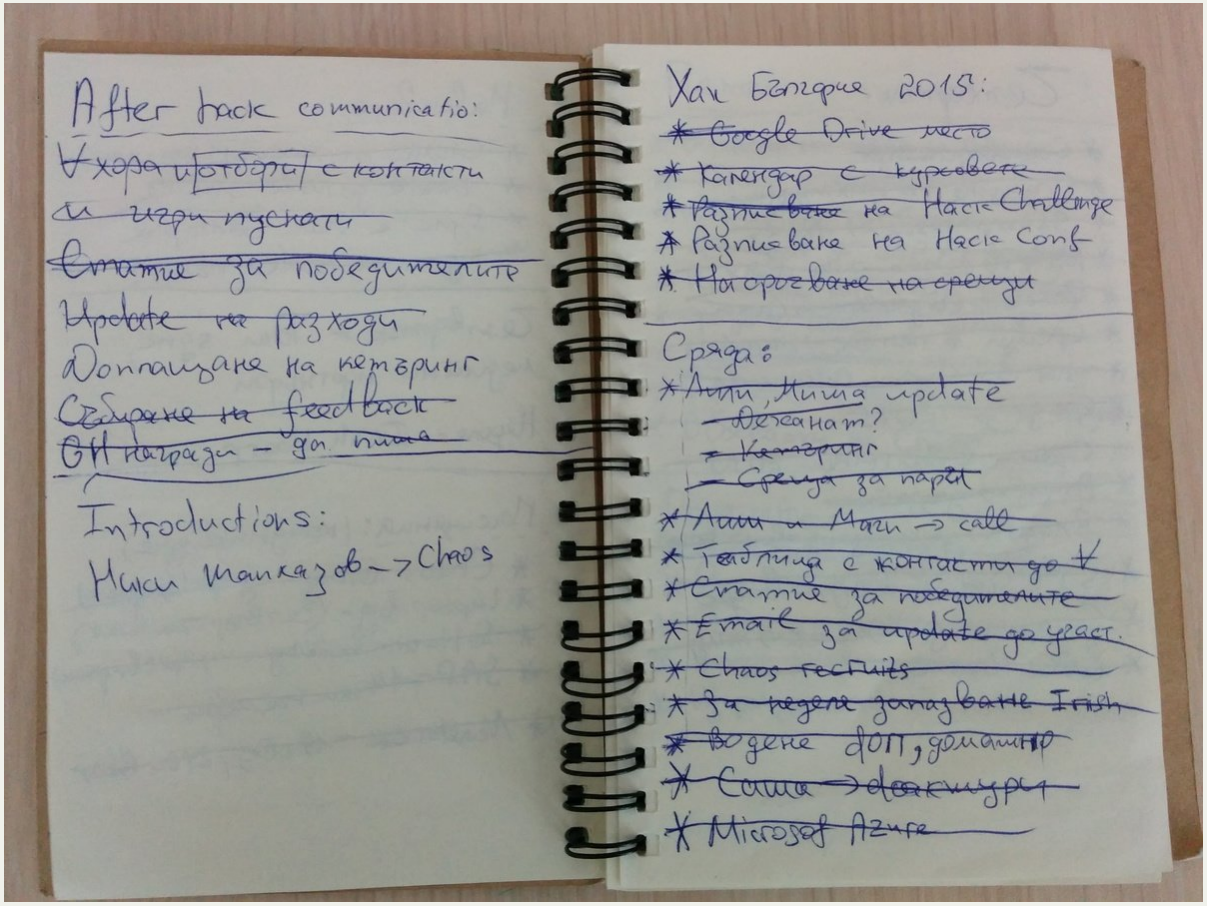
```
person1 = "Ivan"
person2 = "Maria"
person3 = "Ivo"
person4 = "Tony"
```

1. Колко човека има?
2. Как добавяме нов човек към групата?
3. Как премахваме нов човек от групата?
4. Как създаваме нова група от хора, които са прочетени от потребителския вход?

Това е много често срещан проблем.

1. Как **групираме** дадени стойности в общ контейнер?
2. Как да знаем **бройката** на тези стойности?
3. Как да **добавим** или **премахнем** тези стойности?
4. Как да **обходим всяка** една от тези стойности?

В реалния живот ползваме списъци.



Списъците обединяват неща, **които имат смисъл да са заедно**, в общ контейнер

Примерни списъци

1. Списък от всички курсисти в Хак България.
2. Списък от всички курсисти в Програмиране 0.
3. Списък от всички курсисти, които са в присъствената група на Програмиране 0.
4. Списък от неща за купуване за месеца за залата.

В Python, имаме списъци.

```
books = ["Pro Git", "Code Complete"]
people = ["Maria", "Ivan"]
even_numbers = [2, 6, 10, 28]
```

1. Синтаксисът: [елементи, ..., елементи]
2. Имаме празен списък - []
3. Използваме го, за да **групираме елементи**, които може да имат нещо общо помежду им.

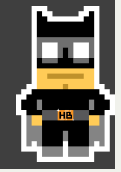
Списъкът:

1. Може да участва в изрази.
2. Представлява **контейнер за стойности.**
3. Има тип. Типът е **списък от типът** на стойностите, които държи вътре.
4. **В един списък може да имаме стойности само от един тип.**

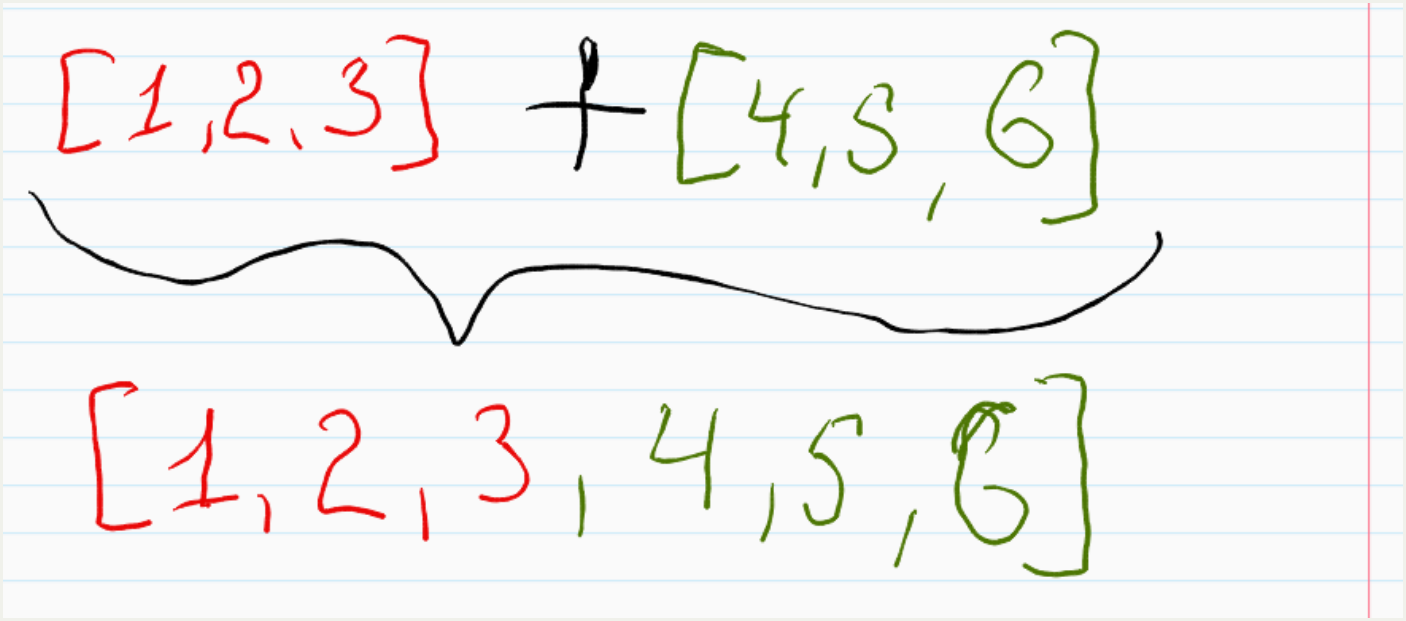
```
>>> []
[]
>>> [1,2,3]
[1, 2, 3]
>>> [True, False]
[True, False]
>>> ["Python", "Django"]
['Python', 'Django']
```

Оператори върху списъци – сливане на 2 списъка.

```
>>> a = []
>>> a
[]
>>> a + [1]
[1]
>>> a = a + [1]
>>> a
[1]
```



a + b работи и ако a и b са списъци.



Добавяне на елементи в начало и край на списък.

```
# Добавяме в началото
# [1, 2, 3]

a = [2, 3]
a = [1] + a

# Добавяме в края
# [2, 3, 1]

b = [2, 3]
b = b + [1]
```

Функция, която взима дължината на списък - len

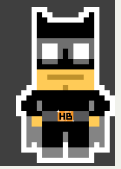
```
>>> a = [1,2,3,4]
>>> b = []
>>> len(a)
4
>>> len(b)
0
```

Списъците са **индексирани**. Всеки елемент има своят индекс.

1. **Индексите се броят от 0.**
2. Може да кажем "дай ми 3тия елемент" - това ще върне стойност.
3. Може да кажем "промени 3тия елемент с нова стойност"

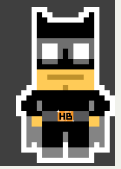
Дай ми елементът на 3то място

```
>>> a = [1,2,3,4]
>>> a[0]
1
>>> a[1]
2
>>> a[2]
3
>>> a[3]
4
```



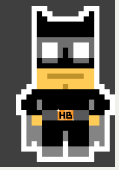
Промени елементът на 3то място.

```
>>> a = [1,2,3,4]
>>> a[0] = 10
>>> a
[10, 2, 3, 4]
>>> a[2] = 20
>>> a
[10, 2, 20, 4]
```



С in може да проверим дали елемент се намира в списък.

```
>>> a = [1, 2, 3]
>>> 1 in a
True
>>> 5 in a
False
>>> b = ["Python", "Django"]
>>> "Python" in b
True
>>> "Rails" in b
False
```



Задачи – Решете задачите от 1-Basic-List-Operations.

Обхождане на списъци – да минем през всеки елемент.

```
books = ["Pragmatic Thinking and Learning",
         "Code Complete"]

for book in books:
    print(book)
```

Същото, но с while.

```
books = ["Pragmatic Thinking and Learning",
         "Code Complete"]

start_index = 0
end_index = len(books)

while start_index < end_index:
    book = books[start_index]
    print(book)

    start_index += 1
```

Обхождания на списъци.

1. С for ходим по елементите. Не се интересуваме от индекси.
2. С while ходим по индексите и сами си взимаме елемента на този индекс.
3. В различни ситуации ще ни трябват различни неща.

Функция, която генерира списък от поредни числа.

```
# range.py

numbers = range(1, 10)

for number in numbers:
    print(number)
```

range(a, b), където a и b са integers

1. Създава списък от числата между **a и b - 1**
2. **Интервалът изглежда [a, b)**
3. Такъв списък искате да го обхождате с for

Сумата между 1 и n

```
# sum_of_n.py

n = input("Enter number: ")
n = int(n)

numbers = range(1, n + 1)

total_sum = 0

for number in numbers:
    total_sum += number

print(total_sum)
```

Задачи – Решете задачите от 2-List-Problems