Словари*

Обычные списки представляют собой набор пронумерованных элементов, то есть для обращения к какому-либо элементу списка необходимо указать его номер. Номер элемента в списке однозначно идентифицирует сам элемент.

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется ***словарем*** или ***ассоциативным массивом***. Соответствующая структура данных в языке Питон называется `dict` .

* Все изображения загружены с сайта Pixabay (https://pixabay.com) (Stunning free images & royalty free stock)

Словарь — неупорядоченная структура данных, которая позволяет хранить пары «ключ — значение». Доступ к значению осуществляется по ключу. В качестве ключа может выступать объект любого **неизменяемого** типа.

Примеры словарей (из жизни):

- собственно словарь. Орфографический, англо-русский, словарь синонимов и пр. Ключ - слово (строка), значение - толкование, варианты перевода, список синонимов.
- база ИНН или СНИЛС. Ключ - уникальный номер или комбинация символов
- база номеров бронирования авиабилетов. Ключ - шестисимвольный код, типа `I6YV3F` . Значение - список различных данных о пассажире, рейсе и т.п.
- база номеров автомобилей

Демонстрация

In [ ]:
```
# Создадим пустой словарь capitals
capitals = dict()   # {}

# Заполним его несколькими значениями
capitals['Russia'] = 'Moscow'
capitals['China'] = 'Beijing'
capitals['USA'] = 'Washington'

countries = ['Russia', 'France', 'USA', 'China']

for country in countries:
    # Для каждой страны из списка проверим,
    # есть ли она в словаре столиц
    if country in capitals:
        print(f'Столица страны {country}: {capitals[country]}')
    else:
        print(f'В базе нет страны с названием {country}')
```

In [ ]:
```
capitals
```

Способы создания словаря "вручную"

In [1]:
```
# 1 множество пар вида ключ:значение
capitals1 = {'Russia': 'Moscow', 'China': 'Beijing',
             'USA': 'Washington'}
print(capitals1)

# 2 список элементов вида имя = значение
capitals2 = dict(Russia = 'Moscow', China = 'Beijing',
                 USA = 'Washington')
print(capitals2)

# 3 список кортежей
capitals3 = dict([("Russia", "Moscow"), ("China", "Beijing"),
                  ("USA", "Washington")])
print(capitals3)

# 4 из двух списков
capitals4 = dict(zip(["Russia", "China", "USA"],
                     ["Moscow", "Beijing", "Washington"]))
print(capitals4)
```

```
{'Russia': 'Moscow', 'China': 'Beijing', 'USA': 'Washington'}
{'Russia': 'Moscow', 'China': 'Beijing', 'USA': 'Washington'}
{'Russia': 'Moscow', 'China': 'Beijing', 'USA': 'Washington'}
{'Russia': 'Moscow', 'China': 'Beijing', 'USA': 'Washington'}
```

zip – один из самых популярных вариантов

не путать с архиватором!

In [2]:
```
list(zip(["Russia", "China", "USA"],
                    ["Moscow", "Beijing", "Washington"]))
```

Out[2]:
```
[('Russia', 'Moscow'), ('China', 'Beijing'), ('USA', 'Washington')]
```

In [ ]:
```
help(zip)
```

Автоматизация создания словарей

In [3]:
```
# словарь на основе генератора
cubes =  {x: x**3 for x in range(1, 6)}
print(cubes)
```

```
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```

Работа с элементами словаря

In [4]:
```
# дальше будем использовать этот словарь
capitals = {'Russia': 'Moscow',
            'China': 'Beijing',
            'USA': 'Washington',
            'France':'Paris'}
```

Получение значения элемента по ключу

In [5]:
```
# Способ 1.
# x = d[key]
town = capitals['Russia']

# Способ 2.
# x = d.get(key)
# или
# x = d.get(key, value)
city = capitals.get('France')

print(town, city)
```

```
Moscow Paris
```

In [6]:
```
# Если в словаре нет такого ключа,
# возникнет ошибка
what = capitals['Armenia']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-6-68c7959b5c9a> in <module>
      1 # Если в словаре нет такого ключа,
      2 # возникнет ошибка
----> 3 what = capitals['Armenia']

KeyError: 'Armenia'
```

In [7]:
```
# Метод get позволяет избегать этой ошибки
# второй параметр метода -
#   то значение,которое возвратится при отсутствии ключа
what = capitals.get('Armenia', 'ой, не знаю')
print(what)
```

```
ой, не знаю
```

In [8]:
```
# Проверка принадлежности ключа словарю:
# x in d
print('France' in capitals)
print('Armenia' in capitals)
```

```
True
False
```

Добавление и удаление элемента

In [9]:
```
# Добавление нового элемента в словарь:
# - обычное присваивание
# d[key] = value
capitals['Armenia'] = 'Yerevan'
print(capitals)
```

```
{'Russia': 'Moscow', 'China': 'Beijing', 'USA': 'Washington', 'France': 'Paris', 'Armenia': 'Yerevan'}
```

In [ ]:
```
# Удаление элемента из словаря
# Способ 1
# del d[key]

# Способ 2
# x = d.pop(key)
# или
# x = d.pop(key, value)
```

Перебор элементов словаря

In [10]:
```
# Перебор элементов словаря
# for key in d:
#     print(key, d[key])
for c in capitals:
    print(c, capitals[c])
```

```
Russia Moscow
China Beijing
USA Washington
France Paris
Armenia Yerevan
```

In [12]:
```
# Представления элементов словаря
# d.keys()
# d.values()
# d.items()
# пример проверки: val in A.values()
print(capitals.keys())
print(capitals.values())
print(capitals.items())
```

```
dict_keys(['Russia', 'China', 'USA', 'France', 'Armenia'])
dict_values(['Moscow', 'Beijing', 'Washington', 'Paris', 'Yerevan'])
dict_items([('Russia', 'Moscow'), ('China', 'Beijing'), ('USA', 'Washington'), ('France', 'Paris'), ('Armenia', 'Ye
revan')])
```

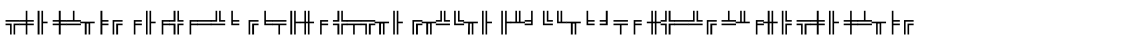
In [13]:
```
# Второй способ перебора элементов
# for key, val in d.items():
#     print(key, val)
for country, capital in capitals.items():
    print(f'{country}, столица: {capital}')
```

```
Russia, столица: Moscow
China, столица: Beijing
USA, столица: Washington
France, столица: Paris
Armenia, столица: Yerevan
```

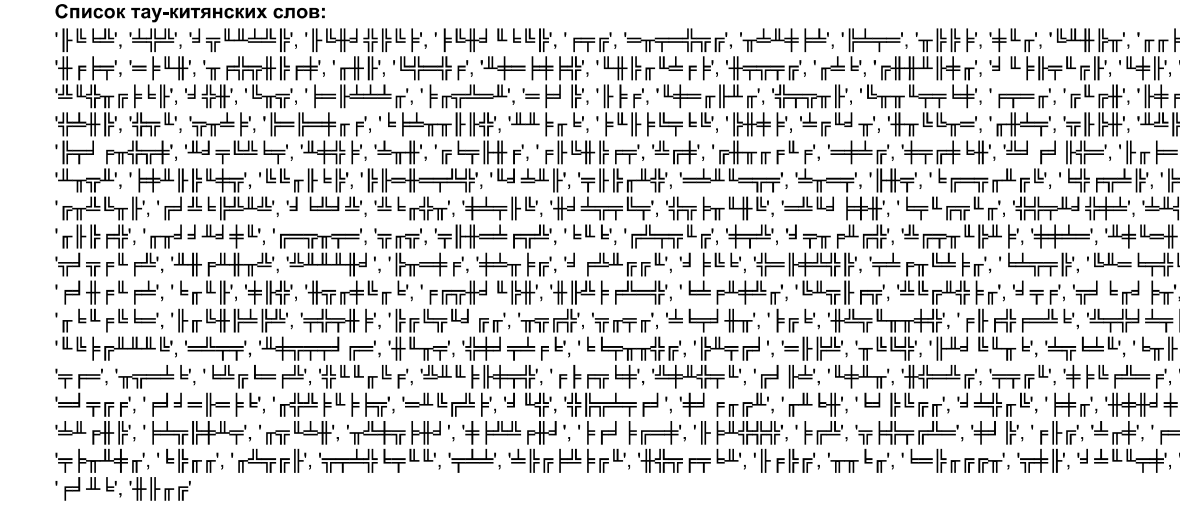
Задачи по теме "Словари"

Задача из контрольной

Знаменитый межгалактический преступник Джим де Гриз встал на путь исправления и хочет украсть из сейфа одного из криминальных авторитетов звездной системы Тау Кита план покорения Вселенной. Чтобы открыть сейф нужно ввести пароль – фразу из дюжины слов на русском языке (узнав об этом, Джим сразу догадался, из какой части планеты Земля происходят предки этого преступного авторитета). У криминального авторитета плохая память, и чтобы не забыть пароль, он написал его прямо на сейфе – только на таукитянском языке. Джим не знает ни русского, ни таукитянского (он даже букв таких не знает!), зато прекрасно умеет пользоваться галактик-нетом, в котором он сумел найти два файла: в одном - небольшой список русских слов, в другом - список их переводов на язык Тау Кита. Поскольку времени на взлом сейфа будет очень мало, нужно подготовиться к операции заранее. Разработайте программу, которая из заданных списков создает таукитянско-русский словарь, и потренируйтесь взламывать код – объясните Джиму, какое русское предложение он должен набрать, если на двери сейфа нацарапано



Список тау-китянских слов:



Список их переводов на русский:

'усовершенствование', 'клапан', 'дрессировщик', 'марс', 'камера', 'фильм', 'размер', 'верба', 'зрение', 'обед', 'слух', 'волчок', 'еда', 'даль', 'помочь', 'мышка', 'ошибка', 'штаб', 'аккуратненький', 'пиво', 'тренировка', 'шкода', 'наглядный', 'честь', 'помеха', 'лесоповал', 'кошмарный', 'клевета', 'спешка', 'резец', 'приписка', 'артериальный', 'круговорот', 'высота', 'возможность', 'гурьба', 'вербовка', 'мальчишка', 'масса', 'идиллия', 'гель', 'застенок', 'линия', 'гимнастика', 'проволочный', 'конкуренция', 'сторожевой', 'мотивация', 'дезинформация', 'мозг', 'тенор', 'победа', 'моторный', 'экзотический', 'боец', 'приветствие', 'пистолет', 'фауна', 'отдел', 'состав', 'квалификация', 'галета', 'беспощадно', 'княжество', 'обстоятельство', 'лодка', 'слог', 'лира', 'фирма', 'вагон', 'сцена', 'дисплей', 'кошка', 'принцип', 'стебелек', 'неспособный', 'кинозвезда', 'добивание', 'агония', 'образование', 'зоопарк', 'разрез', 'иконостас', 'протяженность', 'запад', 'дрема', 'пакость', 'ключ', 'аттракцион', 'пристанище', 'человечество', 'акваланг', 'визитер', 'подачка', 'существо', 'наблюдение', 'бактериологический', 'погибель', 'сечение', 'организация', 'ботаника', 'капут', 'босяк', 'беда', 'слово', 'противник', 'табурет', 'империалист', 'спазм', 'вяз', 'сигарета', 'улучшение', 'страничка', 'труба', 'теневой', 'прием', 'сыск', 'кувырок', 'принуждение', 'председатель', 'большой', 'рытвина', 'символика', 'кормилица', 'завоевание', 'кочка', 'место', 'зверинец', 'биополе', 'одуванчик', 'черт', 'услужливый', 'и', 'годовщина', 'билет', 'боеголовка', 'значение', 'лицо', 'старание', 'тело', 'насильно', 'сухонький', 'лесенка', 'горазд', 'виновник', 'гипофиз', 'будни', 'игра', 'пчела', 'вождение', 'транспарант', 'пыль', 'оптимист', 'очевидность', 'бацилла', 'успех', 'чем', 'преобразователь', 'апельсиновый', 'исключительность', 'культура', 'совестный', 'острие', 'вода', 'заспанный', 'прикрытие', 'фраза', 'наводчик', 'разумно', 'правоохранительный', 'планетолет', 'гадание', 'один', 'полый', 'запасник', 'направление', 'романтика', 'сват', 'картонка', 'смех', 'путаница', 'нашивка', 'дно', 'минерал', 'житель', 'способ', 'кросс', 'персональный', 'выздоровление', 'всплеск', 'весна', 'провинциал', 'организм', 'осознание', 'веточка', 'добрый', 'корзинка', 'период', 'зигзаг', 'население'

```
In [14]: # Текст программы, осуществляющей перевод
tau = ['╟╚╩', '╩╬╝', '╛╦╙╩╩╟', '╟╠╬╬╟╙╞', '╞╚╠╙╘╚╟', '╤╓', '╪╥╤╪╥╓', '╥╧╨╪╠', '╟╩╤', '╥╫╞╞', '╪╙╓', '╚╩╫╞╥',
rus = ['усовершенствование', 'клапан', 'дрессировщик', 'марс', 'камера', 'фильм', 'размер', 'верба', 'зрение', 'обед
d = dict(zip(tau, rus))
note = '╥╫ ╪╥╞╓ ╓╟╫╤╧╘ ╓╧╟╞╒ ╫╥╥╟ ╓╥╧╙╥╟ ╟╝ ╙╙╥╘ ╛╥╒ ╟╠╧╓ ╧╜╓╫╟ ╥╫ ╪╥╞╓'
for tw in note.split():
    print(d[tw], end=' ')
```

```
добрый слово и пистолет возможность добивание горазд большой чем один добрый слово
```

А теперь введите перевод!

Учтите, что стиль мышления таукитян сильно отличается от земного; у них в языке совсем нет глаголов, почти нет предлогов и наречий, полностью отсутствуют падежи и склонения. К счастью, слова друг от друга они отделяют пробелами. Постарайтесь перевести зашифрованную фразу не механически: хорошо бы понять ее смысл и записать ее в виде связного русского предложения – и с наречиями, и с глаголами. А может быть, Вы ее уже слышали раньше? Судьба вселенной в Ваших руках!

перевод:

Добрым словом и пистолетом вы можете добиться гораздо большего, чем одним только добрым словом

Задачи с сайта [ПитонТьютор (http://pythontutor.ru)](http://pythontutor.ru)

Задача 1. Номер появления слова

В единственной строке записан текст. Для каждого слова из данного текста подсчитайте, сколько раз оно встречалось в этом тексте ранее.

Словом считается последовательность непробельных символов идущих подряд, слова разделены одним или большим числом пробелов или символами конца строки.

```
In [ ]: # Дано
simple_text = "one two three two two one"
patterer = "макака коалу в какао макала, \n\
коала какао лениво лакала, \n\
макака макала, коала икала, \n\
устала макака - какао не стало."
#print(patterer)
# Решение 1
d = {}
for word in simple_text.split():
    if word in d:
        print(d[word], end = ' ')
        d[word] += 1
    else:
        print(0, end = ' ')
        d[word] = 1

# Решение 2
d = dict()
for word in patterer.split():
    print(d.get(word, 0), end = ' ')
    d[word] = d.get(word, 0) + 1
```

Задача 2. Словарь синонимов

Дан словарь, состоящий из пар слов. Каждое слово является синонимом к парному ему слову. Все слова в словаре различны. Для слова из словаря, записанного в последней строке, определите его синоним.

Пример ввода	Образец вывода
3	
Hello Hi	
Bye Goodbye	
List Array	
Goodbye	Bye

In [15]:
```
# Решение
n = int(input("n:"))
d = {}
for i in range(n):
    s1, s2 = input('2 synonims: ').split()
    d[s1] = s2
    d[s2] = s1
```

```
n:3
2 synonims: hi hello
2 synonims: bye goodbye
2 synonims: list array
```

In [17]:
```
word = input('word:')
print(d[word])
```

```
word:goodbye
bye
```

Задача 3. Выборы в США

Как известно, в США президент выбирается не прямым голосованием, а путем двухуровневого голосования. Сначала проводятся выборы в каждом штате, и определяется победитель выборов в данном штате. Затем проводятся государственные выборы: на этих выборах каждый штат имеет определенное число голосов — число выборщиков от этого штата. На практике, все выборщики от штата голосуют в соответствии с результатами голосования внутри штата, то есть на заключительной стадии выборов в голосовании участвуют штаты, имеющие различное число голосов.

В первой строке дано количество записей. Далее, каждая запись содержит фамилию кандидата и число голосов, отданных за него в одном из штатов. Подведите итоги выборов: для каждого из участника голосования определите число отданных за него голосов. Участников нужно выводить в алфавитном порядке.

Пример ввода	Образец вывода
5	
Trump 14	Biden 35
Biden 26	Jorgensen 1
Jorgensen 1	Trump 34
Biden 9	
Trump 20	

In [ ]:
```
# Решение
d = {}
n = int(input('n = '))
for i in range(n):
    s = input('data: ')
    name, votes = s.split()
    d[name] = d.get(name, 0) + int(votes)
print(d)
```

In [ ]:
```
for key in sorted(d.keys()):
    print(key, d[key])
```

Задача 4. Самое частое слово

Дан текст: в первой строке задано число строк, далее идут сами строки. Выведите слово, которое в этом тексте встречается чаще всего. Если таких слов несколько, выведите то, которое меньше в лексикографическом порядке.

В качестве теста попробуйте использовать первые 5-7 строк из Дзен Питона.

In [18]:
```
# Дзен Питона
import this
```

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

In [19]:
```
# Решение
d = {}
n = int(input('n: '))
for i in range(n):
    s = input('> ')
    for w in s.split():
        d[w] = d.get(w, 0) + 1
mx = max(d.values())
a = [word for word in d if d[word]==mx]
print(min(a))
```

```
n: 5
> Beautiful is better than ugly
> Explicit is better than implicit
> Simple is better than complex
> Complex is better than complicated
> Flat is better than nested
better
```

In [ ]:
```
Beautiful is better than ugly
Explicit is better than implicit
Simple is better than complex
Complex is better than complicated
Flat is better than nested
```

In [21]:
```
a
```

Out[21]:
```
['is', 'better', 'than']
```

Задача 5. Частотный анализ

Дан текст: в первой строке записано количество строк в тексте, а затем сами строки. Выведите все слова, встречающиеся в тексте, по одному на каждую строку. Слова должны быть отсортированы по убыванию их количества появления в тексте, а при одинаковой частоте появления — в лексикографическом порядке.

Указание. После того, как вы создадите словарь всех слов, вам захочется отсортировать его по частоте встречаемости слова. Желаемого можно добиться, если создать список, элементами которого будут кортежи из двух элементов: частота встречаемости слова и само слово. Например,

```
[(2, 'hi'), (1, 'what'), (3, 'is')]
```

Тогда стандартная сортировка будет сортировать список кортежей, при этом кортежи сравниваются по первому элементу, а если они равны — то по второму. Это почти то, что требуется в задаче.

In [ ]:
```
d = dict()
n = int(input('n: '))
for i in range(n):
    s = input('> ')
    for w in s.split():
        d[w] = d.get(w, 0) + 1
a = [(-d[w], w) for w in d]   # обратите внимание на минус!
a.sort()

for n, word in a:
    print(word)

# for w in a:
#     print(w[1])
```

In [ ]:
```
a
```

Задачи с сайта [LeetCode (https://leetcode.com)]

Задача 1. Сумма двух.
Дан список целых чисел `nums` и целое число `target` — целевое значение. Нужно вычислить индексы двух элементов списка, сумма которых равна этому целевому значению.

Еще раз:
найти индексы `i` и `j`, такие что `i!=j` и `nums[i] + nums[j] == target`.

При решении нужно считать, что в списке есть только одна такая пара элементов.

In [22]:
```
# очевидное плохое правильное решение
def sum_two(nums, target):
    n = len(nums)
    for i in range(n-1):
        for j in range(i+1, n):
            if nums[i]+nums[j]==target:
                return [i, j]
```

In [23]:
```
nums = [2,7,11,15]
target = 9
assert sum_two(nums, target)==[0, 1], 'Тест1 не удался'

nums = [3,2,4]
target = 6
assert sum_two(nums, target)==[1,2], 'Тест2 не удался'

nums = [3,3]
target = 6
assert sum_two(nums, target)==[0,1], 'Тест3 не удался'
```

In [31]:
```
# тест проверки на скорость работы алгоритма
from random import randint
from time import time
n = 10**5
nums = [randint(1, n) for _ in range(n)]
nums[1000] = -5
nums[n - 1000] = -5
target = -10
t = time()
assert sum_two(nums, target)==[1000,n-1000], 'Тест4 не удался'
dt = time()-t
print(dt)
assert dt<0.1, 'Слишком долго'
```

```
16.280023097991943

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-31-8119a6295071> in <module>
     11 dt = time()-t
     12 print(dt)
---> 13 assert dt<0.1, 'Слишком долго'

AssertionError: Слишком долго
```

In [25]:
```
# сначала поговорим про enumerate
a = [3, 5, 7, 11, 0, 5]

for i in range(len(a)):
    print(i, '-', a[i], end = ' ')
print('\n--------------')
for i, x in enumerate(a):
    print(i, '-', x, end=' ')
```

```
0 - 3 1 - 5 2 - 7 3 - 11 4 - 0 5 - 5
--------------
0 - 3 1 - 5 2 - 7 3 - 11 4 - 0 5 - 5
```

In [26]:
```
# правильное хорошее решение
def sum_two_good(nums, target):
    d = {}
    for i, x in enumerate(nums):
        if x in d:
            return [d[x], i]
        else:
            d[target-x] = i
```

In [27]:
```
nums = [2,7,11,15]
target = 9
assert sum_two_good(nums, target)==[0, 1], 'Тест1 не удался'

nums = [3,2,4]
target = 6
assert sum_two_good(nums, target)==[1,2], 'Тест2 не удался'

nums = [3,3]
target = 6
assert sum_two_good(nums, target)==[0,1], 'Тест3 не удался'
```

In [29]:
```
# тест проверки на скорость работы алгоритма
from random import randint
from time import time
n = 10**5
nums = [randint(1, n) for _ in range(n)]
nums[1000] = -5
nums[n - 1000] = -5
target = -10
t = time()
assert sum_two_good(nums, target)==[1000,n-1000], 'Тест4 не удался'
dt1 = time()-t
print(dt1)
assert dt1<1, 'Слишком долго'
```

```
0.05000019073486328
```

In [32]:
```
dt/dt1
```

Out[32]:
```
325.599219897385
```

Дополнительно!!!.

На экзамене этого не будет.

Тем, кому не нужно — не нужно.

Найдено на форумах

Задача А. В тексте программы несколько переменных получили значения. Например,

```
n = 1
pi = 3.14
s = 'hello'
```

Требуется создать словарь, ключами которого были бы имена переменных, а значениями – значения этих переменных. В приведенном примере должен получиться словарь

```
{'n': 1, 's': 'hello', 'pi': 3.14}
```

Решение должно быть универсальным, т.е. не зависеть от имен использованных переменных и присвоенных им значений.

Задача Б. (Не хочу пользоваться списками). Можно ли автоматически (в цикле) присвоить переменным `x1`, `x2`, `x3`, `x4`, `x5` значения 10, 20, 30, 40 и 50?

Идея для решения

Функция `globals()` выдает словарь глобальных переменных (ключ – имя переменной). Функция `locals()` возвращает словарь только локальных переменных. Пример ниже.

Предупреждение: В Ipython Notebook получится слишком много вывода. Лучше выполнить в Python Shell. Или хотя бы перезагрузить ядро.

С другой стороны, если выполнить прямо здесь, станет понятнее, что такое `In[1]` и т.д.

In [ ]:
```
x, y = 5, 10
def test():
    y, z = 33, 44
    print('globals:', globals())
    print('locals:', locals())
test()
```

In [ ]:
```
# Решение задачи Б
for i in range(1,6):
    globals()["x"+str(i)] = 10*i
print(x1, x2, x3, x4, x5)
```

In [ ]:
```
# Решение задачи А
main_dict = set(locals().keys())
#-----------------------------------
nnn = 1
ppi = 3.14
qq = 234455
ss = 'hello'
#-----------------------------------
cur_dict = set(locals().keys())
d = {}
for key in cur_dict - main_dict - {'main_dict'}:
    d[key]=locals()[key]
print(d)
```

In [ ]:
```

```