С Е М И Н А Р 1

Базовые типы данных. Методы работы со строками, списками

ОРД, 2021

Ввод и вывод данных

Ввод

```
x = input() #строка
x = int(input()) #целое число
```

Вывод

```
print(x, sep = ‘ ‘, end =’ ‘)
>>> x = 2
>>> y = 1
>>> print(x, y, sep =‘_‘, end =’/‘)
2_1/
```

```
print(z)
print(m+n)
print(’Hello’)
```

Вводим два любых значения с клавиатуры. Программа должна перед и после каждого значения, которое вы ввели, ставить нижнее подчеркивание.

Ввод:

мне

Ок

Вывод:

Ответ:_мне_ок_

Типы данных

Узнать тип данных

```
>>> print(type(x))
```

Строки

Ввод с клавиатуры

```
x = input()
```

Присваивание значений

```
x = 'punk'
```

Базовые операции

Сложение строк

```
>>> x = 'punk'
>>> y = 'rock'
>>> print(x+y)
punkrock
```

```
>>> x = input()
>>> y = input()
>>> print(x+y)
5
7
57
```

Дублирование строки

```
>>> print('punk' * 3)
punkpunkpunk
```

```
>>> x = input()
>>> x = x * 3
>>> print(x)
5
555
```

Длина строки

```
>>> x = 'punk'
>>> len(x)
4
```

```
>>> len('555555')
6
```

Метод replace

```
>>> print('punkrock'.replace('k', 'K'))
punKrocK
```

Метод count

```
>>> print('5556785'.count('5'))
4
```

Переменная 1 = число, переменная 2 = слово, переменная 3 = знак препинания.

Ввод
90
stop
!

Вывод
stop90!

Ввод
90
stop
!

Вывод
9090!

```
>>> x = 'punkrock'
```

0	1	2	3	4	5	6	7
p	u	n	k	r	o	c	k
-8	-7	-6	-5	-4	-3	-2	-1

Индексы

Первый элемент

```
>>> print(x[0])
p
```

N-ый элемент

```
>>> print(x[5])
o
>>> print(x[-3])
o
```

Метод find

```
>>> print(x.find('k'))
4
```

Метод rfind

```
>>> print(x.rfind('k'))
7
```

Срезы

```
>>> print(x[2:5])
nkr
>>> print(x[3:-2])
kro
```

```
>>> print(x[:6])
punkro
>>> print(x[1:])
unkrock
>>> print(x[:])
punkrock
```

Шаг среза

```
>>> print(x[2::2])
nrc
```

Переворот

```
>>> print(x[::-1])
kcorknup
```

Вводим с клавиатуры значение строки длиной 9 символов. Вводим число от 0 до 8 с клавиатуры (это индекс).

Вывести символ соответствующий этому индексу.

Ввод:
ваовтомто
1

Вывод:
а

Сделать срез с 1 по 7

Ввод:
ваовтомто

Вывод:
аовтомт

Вывести каждый третий символ, начиная с 1

Ввод:
ваовтомто

Вывод:
атт

Числа

Натуральные
int

x + y	сложение
x - y	вычитание
x * y	умножение
x / y	деление
x // y	целая часть от деления
x % y	остаток от деления
x ** y	возведение в степень
abs(x)	модуль числа

Вещественные
float

round (x)	округление

Модуль math

```
>>> import math
```

math.pi	число ПИ
math.sqrt(x)	корень
math.ceil(x)	округление вверх
math.floor(x)	округление вниз
round(x)	"Банковское округление“ - округление к ближайшему чётному

Комплексные
complex

```
>>> x = complex(a, b)
>>> print (x)
(a+bj)
```

Вводим вещественное и целое число. Перемножаем их. Сколько раз в получившемся произведении повторяется число 5?

Ввод

0.25

7

Вывод

1

Метод count

```
>>> print('5556785'.count('5'))
4
#работает только со строками
```

Конвертация числа в строку

```
>>> x = 2 #число
>>> x = str(x) #перевод в строку
```

Списки

упорядоченные изменяемые коллекции объектов произвольных типов

Синтаксис

имя = [элемент1, …, элемент N]

имя = list[‘элемент1’]

Заполнение списка

```
>>> a = []
>>> for i in range(int(input())):
    a.append(int(input()))
>>> print(a)
```

```
>>> x = [1, ‘g’, ’z1’]
>>> print(x)
[1, ‘g’, ’z1’]
```

Добавляем элемент

```
>>> x = list(‘korov’)
>>> x.append(‘a’)
>>> print(x)
['k', 'o', 'r', 'o', 'v', 'a']
```

Расширение списка

```
>>> x = list(‘kor’)
>>> y = list(‘ova’)
>>> x.extend(y)
>>> print(x)
['k', 'o', 'r', 'o', 'v', 'a']
```

Удаление элемента

```
>>> x = list(‘korov’)
>>> x.remove(‘r’)
>>> print(x)
['k', 'o', 'o', 'v', 'a']
```

Очистка списка

```
>>> x = list(‘korov’)
>>> x.clear()
>>> print(x)
[]
```

К элементам списка можно обращать по индексу. **Нумерация с 0**

Создайте список содержащий информацию о планетах нашей Солнечной системы. (Меркурий, Венера, Земля, Марс, Юпитер, Сатурн, Уран, Нептун)

Задание 1

Вывести 5 планету/4 с конца планету Солнечной системы (двумя способами)

Вывод
Юпитер

Задание 2

Вывести планеты с 2 по 4

Вывод
Венера, Земля, Марс

Задание 3

Вывести планеты начиная с 5/4 последних планеты (двумя способами)

Вывод
Юпитер, Сатурн, Уран, Нептун

Задание 4

Заменить название 4 планеты СС на «Сникерс»

Задание 5

Заменить название первых 3 планет следующим образом: Мер, Вен, Зем

Задание 6

Вернуть первым 4 планетам исходные названия

Задание 7

Посчитать количество планет в списке с помощью функции len

Задание 8

Отсортировать планеты СС в алфавитном порядке с помощью метода sorted

Задание 9

Добавить в наш список планет еще Плутон, с помощью метода append

Задание 10

Удалить Плутон из списка с помощью метода pop (удаляет последний элемент в списке), либо с помощью remove

Задание 11

Найти какой по счету Сатурн в солнечной системе с помощью метода index

Кортежи

неизменяемые списки

Синтаксис

имя = tuple(элемент 1, …, элемент N)

имя = (элемент 1, …, элемент N)

имя = элемент 1,

Зачем они нужны?

- *Защищен от изменений*
- *Меньший размер чем у списка*
- *Использовать как ключ для словаря*

```
>>> a = tuple('hello’)
>>> print(a)
('h', 'e', 'l', 'l’, 'o')
```

```
>>> a = (‘s’, ’m’)
>>> a
('s’, ’m’)
```

```
>>> a = 's’,
>>> a
('s',)
```

Множества

Синтаксис

```
>>> a = set()  # пустое множество

>>> a = set('hello')
>>> a
{'h', 'o', 'l', 'e'}
# видим, что порядок нарушен
```

Не дайте фигурным скобкам себя обмануть!

```
>>> b = {}  # это уже не множество, а словарь, про него – позже!
```

Множество в Python - "контейнер", содержащий **не повторяющиеся** элементы **в случайном порядке.**

- Можно очень быстро проверять принадлежность элемента множеству
- С множествами можно выполнять множество операций: находить объединение, пересечение, разность
- Множества удобно использовать для удаления повторяющихся элементов:

```
>>> words = ['gimme', 'gimme', 'gimme', 'a', 'man', 'after', 'midnight']
>>> set(words)
{'man', 'a', 'midnight', 'after', 'gimme'}
```

```
>>> a = {1, 2, 3}
>>> a.add('mississippi')
>>> a
{1, 2, 3, 'mississippi'}
```

```
>>> A = {1, 2, 3}
>>> B = {3, 2, 3, 1, 1, 1}
>>> print(A==B)
True
```

```
>>> a = {i ** 2 for i in range(10)}
>>> a
{0, 1, 4, 81, 64, 9, 16, 49, 25, 36}
```

Операции с множествами

С множествами в питоне можно выполнять обычные для математики операции над множествами.

A | B **A.union(B)**	Возвращает множество, являющееся объединением множеств `A` и `B`.
A |= B **A.update(B)**	Добавляет в множество `A` все элементы из множества `B`.
A & B **A.intersection(B)**	Возвращает множество, являющееся пересечением множеств `A` и `B`.
A &= B **A.intersection_update(B)**	Оставляет в множестве `A` только те элементы, которые есть в множестве `B`.
A - B **A.difference(B)**	Возвращает разность множеств `A` и `B` (элементы, входящие в `A`, но не входящие в `B`).
A -= B **A.difference_update(B)**	Удаляет из множества `A` все элементы, входящие в `B`.
A ^ B **A.symmetric_difference(B)**	Возвращает симметрическую разность множеств `A` и `B` (элементы, входящие в `A` или в `B`, но не в оба из них одновременно).

Словари

неупорядоченный набор данных произвольного типа с доступом по ключу

Синтаксис

создание

имя = {ключ 1:значение 1, …, ключ N: значение N}

имя = dict(‘ключ 1’ = значение 1, …, ‘ключ N’ = значение N)

добавление элемента

имя[новый ключ] = значение

удаление элемента

del имя[ключ]

вызов словаря

имя

Зачем они нужны?

- *Для подсчета числа каких-то объектов. В этом случае нужно завести словарь, в котором ключами являются объекты, а значениями — их количество.*
- *Для хранение каких-либо данных, связанных с объектом. Ключи — объекты, значения — связанные с ними данные.*
- *Установка соответствия между объектами (например, “родитель—потомок”). Ключ — объект, значение — соответствующий ему объект.*
- *Если нужен обычный массив, но максимальное значение индекса элемента очень велико, и при этом будут использоваться не все возможные индексы (так называемый “разреженный массив”), то можно использовать ассоциативный массив для экономии памяти.*

Методы словарей	
имя.clear()	Очищает словарь
имя.copy()	Возвращает копию словаря
имя.items()	Возвращает пары (ключ, значение)
имя.keys()	Возвращает ключи
имя.values()	Возвращает значения в словаре
имя.update({ключ; значение})	Обновляет словарь, добавляя пары.

Перебор словаря

```
>>> a = {1: ‘one’, 2: ‘two’, 3: ‘three’}
>>> for key, value in a.items():
print(key, ‘:’, value)
1 : one
2 : two
3 : three
```

```
>>> a = {1: ‘one’, 2: ‘two’, 3: ‘three’}
>>> for key in a.keys():
print(key)
1
2
3
```

```
>>> a = {1: ‘one’, 2: ‘two’, 3: ‘three’}
>>> for val in a.values():
print(val)
one
two
three
```

Создайте словарь, где ключ – это римская цифра, а значение арабская. Для цифр от 1 до 3.

Задание 1

Выведете цифру 2 из словаря

Задание 2

Добавьте в словарь число 11, с ключом «XI»

Задание 3

Замените значение под ключом «I» на слово «один»