

Занятие 5

Словари, примитивные типы

Что напечатает?

```
abc = {1:'1', '1':1}
```

```
for k, v in abc.items():
```

```
    print(k, v)
```

```
#-----
```

```
nums = [0, 1, 2, 3, 4, 5]
```

```
nums.append(nums[:])
```

```
print(len(nums))
```

```
#-----
```

```
data = {tuple(): 1}
```

```
print(data)
```

Задание 4-1

Напишите калькулятор (простой).

На вход подается строка, например:

1 + 2 или 5 – 3 или 3 * 4 или 10 / 2.

Вывод: сосчитать и напечатать результат операции.

Гарантируется, что два операнда и операция есть в каждой строке, и все они разделены пробелами.

Задача 4-2

Вводим натуральное число n .

Напечатайте спираль из чисел $1, 2, 3, \dots, n * n$

Например для $n = 4$:

```
1  2  3  4
12 13 14 5
11 16 15 6
10 9  8  7
```

Можно использовать словарь с двумя индексами $d[x, y]$

Задача 4-3

Ввод: 2 предложения, содержащие пробелы, знаки препинания.

Определить, являются ли эти предложения анаграммами (т.е. имеют одинаковый набор букв).

Игнорируем пробелы, знаки препинания, цифры и т.д.

Вывод: Если да, то True, если нет, то False

```
import string
print(*dir(string))
print(*string.ascii_letters)
print(*string.punctuation)
```

Задание

Дан кортеж (123, 234, 345, 456, 567, 678, 789, 890).

Вводится еще одно целое число больше 0.

Создайте новый кортеж из первого кортежа и введенного числа, чтобы в новом кортеже числа не убывали.

Коллекции

1. Строка (str) 'Hello world'
2. Список (list) [1, 100, 1, 'a', True]
3. Кортеж (tuple) (1, 100, 1, 'a', True)
4. **Словарь (dict) {1:1, 22:100, 123:1, 'a':'a', 5:True}**
5. Множество (set) {1, 100, 'a', True}

•

Словари – самое главное

Формирование словаря:

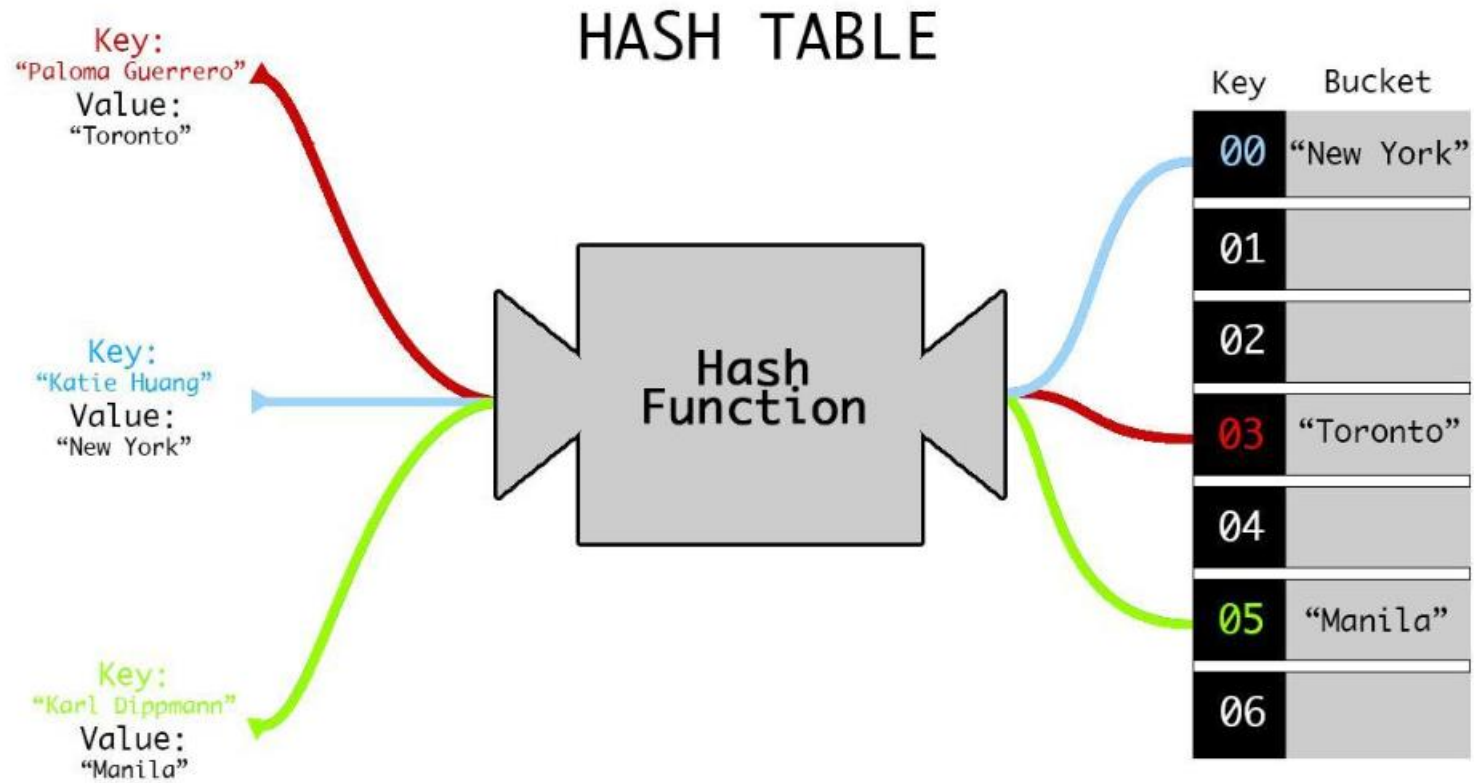
`abc = {}` или `abc = {1:11, '2':222, 'xyz':'xyz'}`

Изменения значения: `abc[1] = 11111`

Новое значение: `abc[3] = 3456`

Проверка, есть ли ключ: `x in abc`

Хэш-таблицы – быстрый доступ



get(*key*[, *default*])

Метод dict.get() возвращает значение для ключа key, если ключ находится в словаре, если ключ отсутствует то вернет значение default.

Если значение default не задано и ключ key не найден, то метод вернет значение None.

Метод dict.get() никогда не вызывает исключение KeyError, как это происходит в операции получения значения словаря по ключу [dict[key]].

```
x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

```
x.get('two', 0) # 2
```

```
x.get('ten', 0) # 0
```

```
print(x)
```

setdefault(*key*[, *default*])

Похож на get, но есть отличие!

```
dct = {1:111, 2:222, 3:333}
```

```
print(dct.get(4, 0))
```

```
print(dct)
```

```
print(dct.setdefault(4, 0))
```

```
print(dct)
```

Что напечатает?

```
abc = {1:'111', 2:'222'}
```

```
print(abc.get(3, '333'))
```

```
print(abc)
```

```
print(abc.setdefault(4, '444'))
```

```
print(abc)
```

Задание

На вход подается число n . Затем на n строчках подается по паре слов – синонимов, например:

большой огромный

маленький небольшой

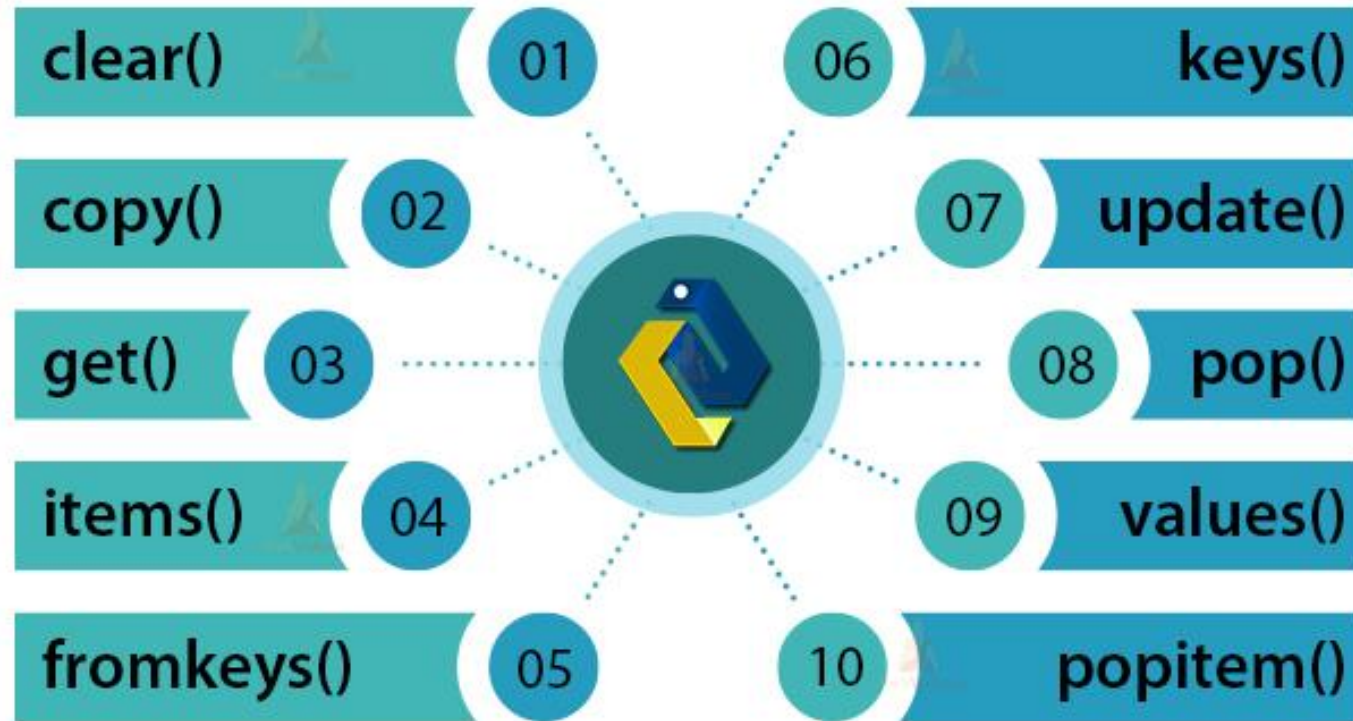
big large ...

Ваша задача составить словарь синонимов.

Затем в бесконечном цикле (`while True`) введите слово, и программа должна напечатать его синоним.

Окончание работы программы – ввод слова “stop”.

Python Dictionary Methods



clear()

Метод производит удаление всех элементов из словаря.

```
>>> x = {'one': 0, 'two': 20, 'three': 3, 'four': 4}
```

```
>>> x.clear()
```

```
>>> x
```

```
# {}
```

copy()

Метод создает копию словаря.

```
>>> x = {'one': 0, 'two': 20, 'three': 3, 'four': 4}
```

```
>>> y = x.copy()
```

```
>>> y
```

```
{'one': 0, 'two': 20, 'three': 3, 'four': 4}
```


items()

Метод `dict.items()` возвращает новый список кортежей вида `(key, value)`, состоящий из элементов словаря.

```
>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

```
>>> items = x.items()
```

```
>>> items
```

```
dict_items([('one', 1), ('two', 2), ('three', 3), ('four', 4)])
```

```
for k, v in dict_items.items():
```

```
    print(k,v)
```

Итерирование словаря

dict = { **k** : **v** , **k2** : **v2** , **k3** : **v3** }



```
statistic_dict = {'b': 10, 'd': 30, 'e': 15, 'c': 14, 'a': 33}
```

```
for key, val in statistic_dict.items():  
    print(key)  
    print(val)
```

```
for key in statistic_dict:  
    print(key, statistic_dict[key])
```

Задание

Вводится строка, состоящая из чисел, разделенных пробелом.

Превращаем ее в список (`lst = list(map(int, input().split()))`)

Составьте и напечатайте словарь, ключи – числа из списка, а значения - списки из индексов, на которых эти числа стоят.

Например:

Ввод: 1 1 1 22 1 1 22 33 4 5 1

Вывод: {1:[0, 1, 2, 4, 5, 10], 22:[3, 6], 33:[7], 4:[8], 5:[9]}

keys()

Метод dict.keys() возвращает список-представление всех ключей , содержащихся в словаре dict.

```
>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

```
>>> keys = x.keys()
```

```
>>> keys
```

```
dict_keys(['one', 'two', 'three', 'four'])
```

keys()

Список-представление ключей `dict_keys`, является динамичным объектом. Это значит, что все изменения, такие как удаление или добавление ключей в словаре сразу отражаются на этом представлении.

Производим операции со словарем 'x', а все

отражается на списке-представлении ``keys``

```
>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

```
>>> keys = x.keys()
```

```
>>> del x['one']
```

```
>>> keys
```

```
dict_keys(['two', 'three', 'four'])
```

```
>>> x
```

```
{'two': 2, 'three': 3, 'four': 4}
```

values()

Метод `dict.values()` возвращает новый список-представление всех значений `dict_values`, содержащихся в словаре `dict`.

Список-представление значений `dict_values`, является динамичным объектом.

Это значит, что все изменения, такие как удаление, изменение или добавление значений в словаре сразу отражаются на этом представлении.

```
>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

```
>>> values = x.values()
```

```
>>> values
```

```
# dict_values([1, 2, 3, 4])
```

fromkeys(iterable[, value])

Метод dict.fromkeys() встроенного класса dict() создает новый словарь с ключами из последовательности iterable и значениями, установленными в value.

```
>>> x = dict.fromkeys(['one', 'two', 'three', 'four'])
```

```
>>> x
```

```
{'one': None, 'two': None, 'three': None, 'four': None}
```

```
>>> x = dict.fromkeys(['one', 'two', 'three', 'four'], 0)
```

```
>>> x
```

```
{'one': 0, 'two': 0, 'three': 0, 'four': 0}
```

update() - объединение словарей


dict1 = { k1: v1 }
dict2 = { k2: v2 }

```
showcase_1 = {'Apple': 2.7, 'Grape': 3.5, 'Banana': 4.4}
```

```
showcase_2 = {'Orange': 1.9, 'Coconut': 10}
```

```
showcase_1.update(showcase_2)
```

```
print(showcase_1)
```

```
> {'Apple': 2.7, 'Grape': 3.5, 'Banana': 4.4, 'Orange': 1.9,  
  'Coconut': 10}
```


pop(key[, default])

Метод dict.pop() вернет значение ключа key, а также удалит его из словаря dict. Если ключ не найден, то вернет значение по умолчанию default.

```
>>> x = {'one': 0, 'two': 20, 'three': 3}
```

```
>>> x.pop('three')
```

```
3
```

```
>>> x
```

```
{'one': 0, 'two': 20}
```

```
>>> x.pop('three', 150)
```

```
150
```

```
>>> x.pop('three')
```

```
# Traceback (most recent call last):
```

```
#   File "<stdin>", line 1, in <module>
```

```
# KeyError: 'three'
```

popitem()

Метод dict.popitem() удалит и вернет двойной кортеж (key, value) из словаря dict. Пары возвращаются с конца словаря, в порядке **LIFO** (последним пришёл - первым ушёл)

```
>>> x = {'one': 0, 'two': 20, 'three': 3}
```

```
>>> x.popitem()
```

```
('four', 4)
```

```
>>> x.popitem()
```

```
('three', 3)
```

```
>>> x.popitem()
```

```
('two', 20)
```

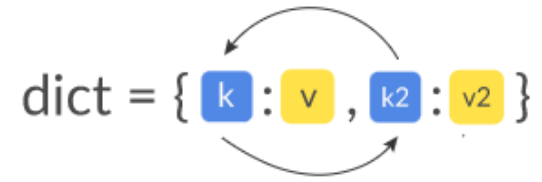
```
>>> x.popitem()
```

```
# Traceback (most recent call last):
```

```
#   File "<stdin>", line 1, in <module>
```

```
# KeyError: 'popitem(): dictionary is empty'
```

Сортировка словаря



```
statistic_dict = {'b': 10, 'd': 30, 'e': 15, 'c': 14, 'a': 33}
```

```
for key in sorted(statistic_dict):  
    print(key)
```

a

b

c

d

e

Задание

На вход подается список продаж в формате: Товар Количество.

Например: Яблоко 10

Груша 5

Яблоко 5

0

Окончание ввода – число 0.

Составьте и напечатайте сводный отчет о продажах, например:

Груша 5

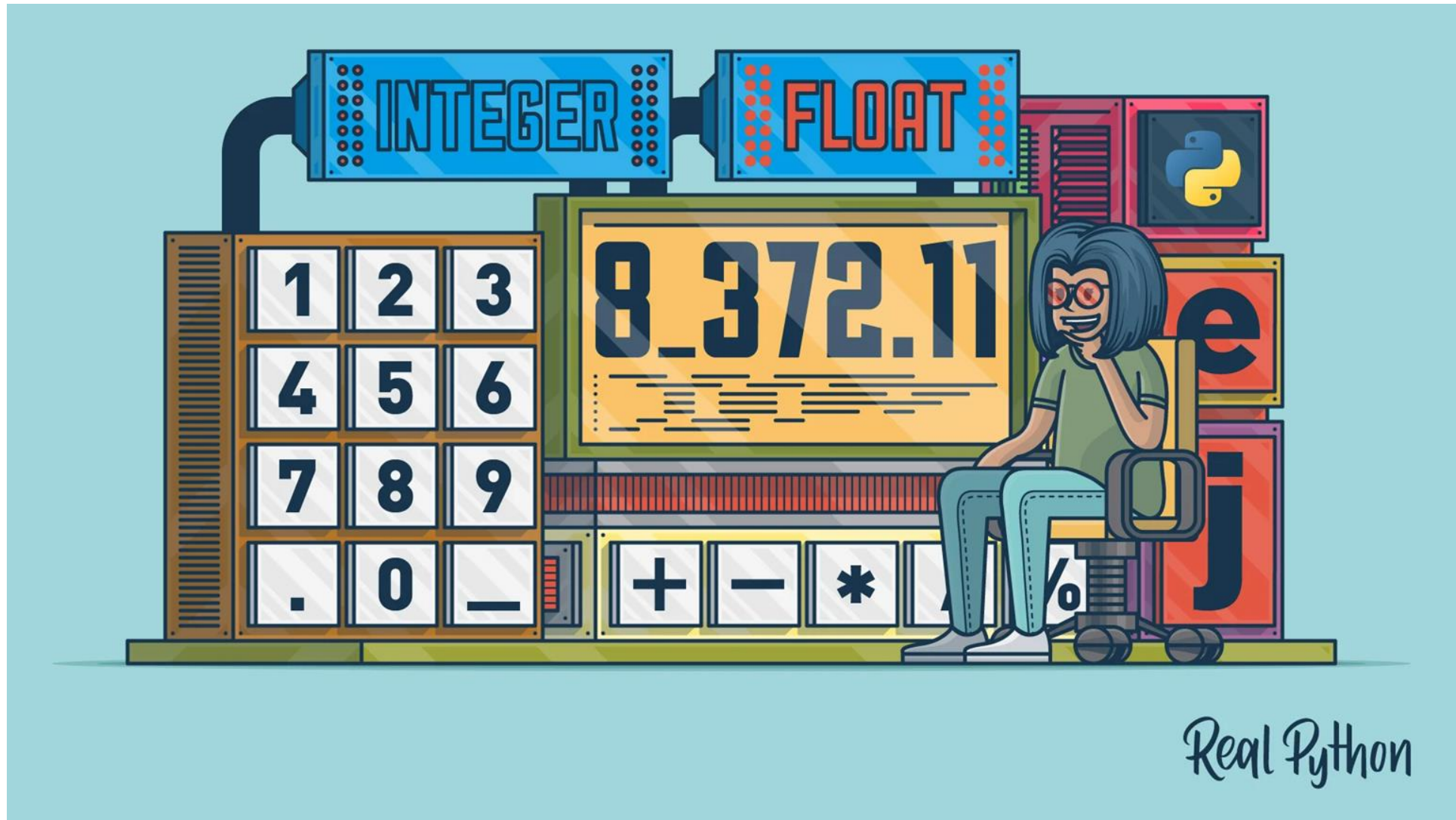
Яблоко 15

отсортированный по товару в алфавитном порядке.

Примитивные типы

- Целые числа (int)
- Числа с плавающей запятой (float)
- Комплексные числа (complex)
- Логический (bool)
- NoneType

NUMBER



Real Python

Операторы в Python для работы с числами

Операторы в Python для работы с числами:

- + сложение
- вычитание
- * умножение
- / деление
- // целочисленное деление
- % деление с остатком
- ** возведение в степень

`abs(x)` - модуль числа

`divmod(x, y)` - пара (`x // y`, `x % y`)

`pow(x, y[, z])` x^y по модулю (если модуль задан)

Что напечатает?

```
print(round(1.1))
```

```
print(round(1.5))
```

```
print(round(2.5))
```

```
for k in [4, 3, 2, 1, 0]:
```

```
    print(round(12.3456, k))
```

```
print(15 // 6)
```

```
print(15 % 6)
```

```
print(divmod(15, 6))
```


Ввод числа типа float

```
f = float(input())  
print(type(f))
```

```
f = 1.23  
print(f + 1, f * 2, f ** 2, abs(f), round(f))
```

`round(x, n)` – округление `n` знаков после запятой, по умолчанию `n = 0`

```
a = float('inf')  
print(a)  
a > 1
```

Системы счисления

Десятичная

>>> 7 → int

>>> 3.14 → float

Двоичная

>>> 0b0010 → int

Восьмеричная

>>> 0o07 → int

Шестнадцатиричная

>>> 0x0F → int

Вопрос: какие системы счисления мы используем в повседневной жизни?

Что напечатает?

```
print(0b10 + 0b10)
```

```
print(bin(4))
```

```
print(0o07 + 0o07)
```

```
print(oct(14))
```

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

```
    print(hex(i))
```

Функции преобразования чисел

`int(x)` - преобразование к целому числу в десятичной системе счисления.

`bin(x)` - преобразование целого числа в двоичную строку.

`hex(x)` - преобразование целого числа в шестнадцатеричную строку.

`oct(x)` - преобразование целого числа в восьмеричную строку.

`int(str, n)` – преобразование строки цифр из n-ичной системы счисления в десятичную

```
print(int('10', 2))
```

```
print(int('10', 8))
```

```
print(int('10', 16))
```

```
# Что напечатает?
```

Задание

Введена длительность некоторого процесса в секундах.

Напечатайте, сколько это часов, минут и секунд.

Т.е. переведите введенное число в 60 – ричную систему))).

Если число часов будет больше 24, то сосчитайте сколько суток, часов, минут, секунд.

Scientific notation

Python использует нотацию E для отображения больших чисел с плавающей запятой

>> 2000000000000000000000.0 → 2e+17

>> 1e15 → 1000000000000000000.0

>> 1e16 → ?

>> 1e-4 → 0.0001

>> 1e-5 → ?

>> 1e-0 → ?

Особенности чисел

Можно работать с большими числами

[illegible]

Форма записи числа

>> 9 999 999

Если нужна + бесконечность (inf)

>> 2e400 \rightarrow inf

Если нужна - бесконечность (-inf)

>> -2e400 \rightarrow -inf

Немного о потере точности

>> 3.14 * 10 → 31.40000000000000002

>> -4 // 3 → -2

>> 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 +
0.1

0.9999999999999999

Комплексные числа

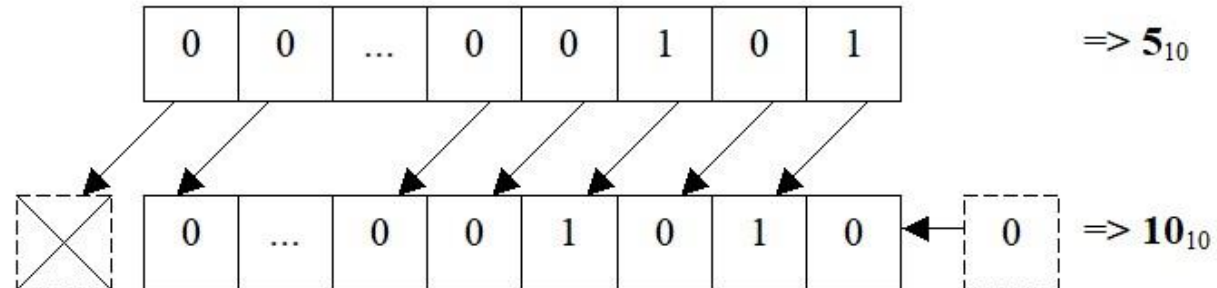
- Комплексное число — это любое число в форме $a + bj$, где a и b — действительные числа, а $j*j = -1$.
- Каждое комплексное число $(a + bj)$ имеет действительную часть (a) и мнимую часть (b) .
- $\gg n = 4 + 3j \rightarrow (4+3j)$

Битовые операторы

- \sim битовый оператор НЕТ (инверсия, наивысший приоритет);
- \ll, \gg – операторы сдвига влево или сдвига вправо на заданное количество бит;
- $\&$ битовый оператор И
- \wedge битовое исключающее ИЛИ
- $|$ битовый оператор ИЛИ.

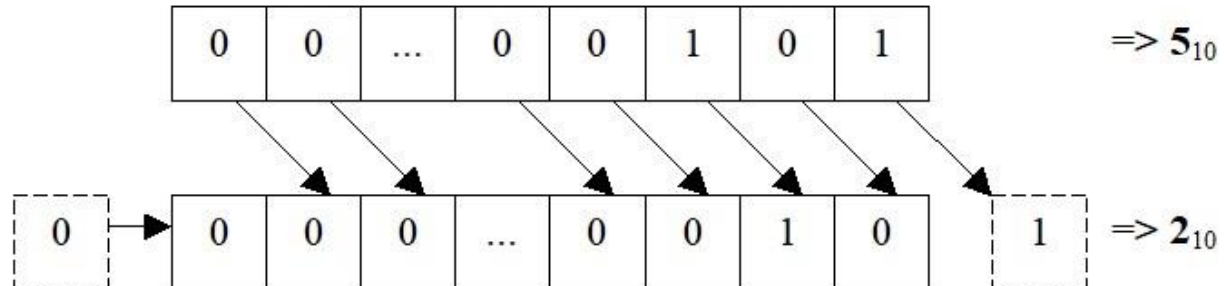
Пример: Операторы сдвига влево <<, вправо >>

x = 5
y = 5 << 1 # y = 10



a)

x = 5
y = x >> 1 # y = 2



b)

Задание

Введите число $n > 0$.

Напечатайте 'Простое', если оно делится только на себя и на 1.

В противном случае напечатайте 'Составное')

Задача 5-1

Ввести число n .

Напечатать треугольник Паскаля.

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

И т.д. n – номер последней строки.

Задача 5-2

1. Ввести число. Напечатать все его делители.

Например: 12

Вывод: 1 2 3 4 6 12

2. Более сложный вариант, напечатать только его простые делители и их степени.

Например: 12 $(12 = (2 ** 2) * (3 ** 1))$

Вывод:

2 – 2

3 – 1

Задание 5-3

Напечатайте ряд чисел Фибоначчи до введенного номера n

$$f[0] = 1, f[1] = 1$$

$$f[k] = f[k - 1] + f[k - 2]$$

Пример последовательности:

1, 1, 2, 3, 5, 8, 13, 21, ...