Занятие 5

Примитивные типы

Начнем с ...

- 1. Разминка (Что напечатает?)
- 2. Обсуждение домашнего задания

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)
```

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:

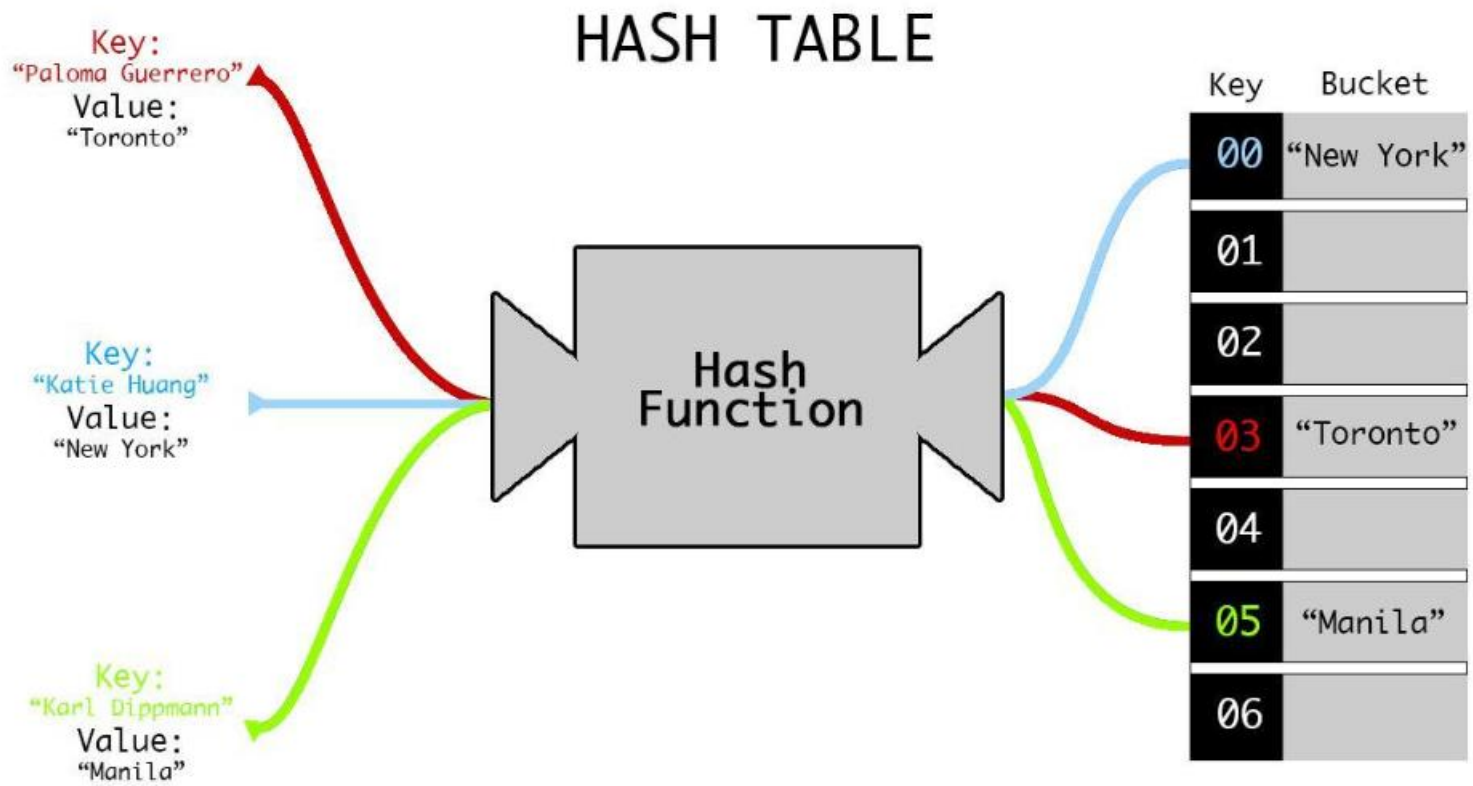
abc = {} или abc = {1:11, ‘2’:222, ‘xyz’:’xyz’}

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.

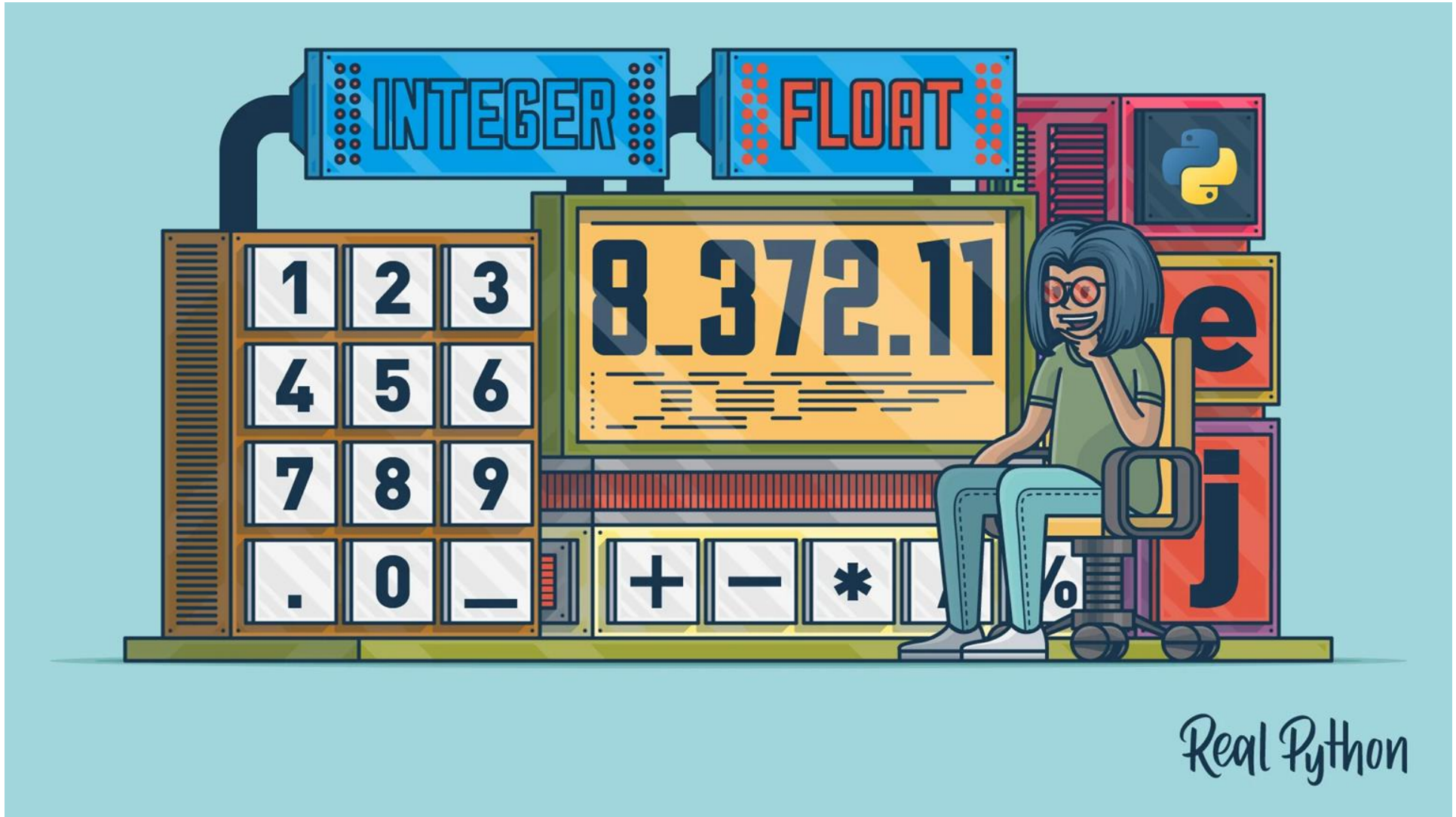
Затем в бесконечном цикле (while True) введите слово, и программа должна напечатать его синоним.

Окончание работы программы – ввод слова “stop”.

Примитивные типы

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

NUMBER



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

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.

Особенности чисел

- Можно работать с большими числами
- >>**x**=9999999999999999999999999999999999999999999
9999999999999999999999999999999999999999999999
9999999999999999999999999999999999999999999999
999
- Форма записи числа
- >> 9_999_999
- Если нужна + бесконечность (inf)
- >> 2e400 → inf
- Если нужна - бесконечность (-inf)
- >> -2e400 → -inf

Scientific notation

- Python использует нотацию E для отображения больших чисел с плавающей запятой
- >> 20000000000000000.0 → 2e+17
- >> 1e15 → 1000000000000000.0
- >> 1e16 → **?**
- >> 1e-4 → 0.0001
- >> 1e-5 → **?**
- >> 1e-0 → **?**

Немного о потери точности

- >> 3.14 * 10 → 31.400000000000002
- >> -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

Пример использования Decimal, Fraction

```
>>> 0.1 + 0.1 + 0.1 – 0.3
5.551115123125783e-17
```

Для высокой точности следует использовать другие объекты (например decimal и fraction)

```
>>> from decimal import Decimal
>>> q=w=e=r=t=y=u=i=o=p=Decimal('0.1')
>>> q+w+e+r+t+y+u+i+o+p
Decimal('1.0')
```

```
from fractions import Fraction
a = Fraction(2, 3)
b = Fraction(1, 3)
a + b
```

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).

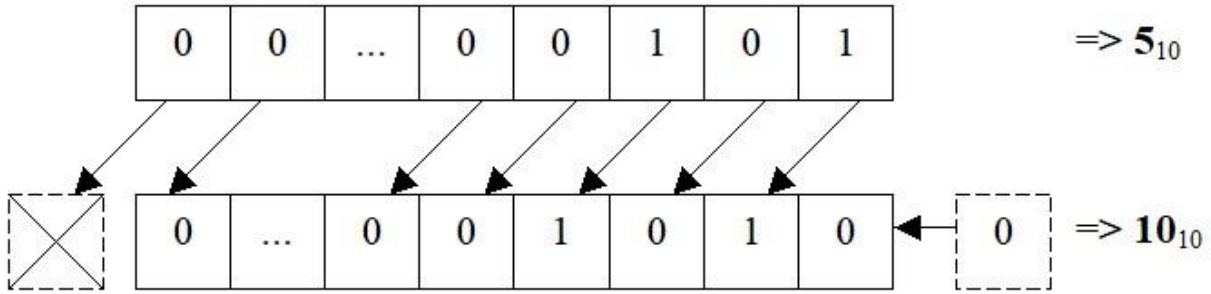
- >>n = 4 + 3j → (4+3j)

Битовые операторы

- ~ битовый оператор НЕТ (инверсия, наивысший приоритет);
- <<, >> – операторы сдвига влево или сдвига вправо на заданное количество бит;
- & битовый оператор И
- ^ битовое исключающее ИЛИ
- | битовый оператор ИЛИ.

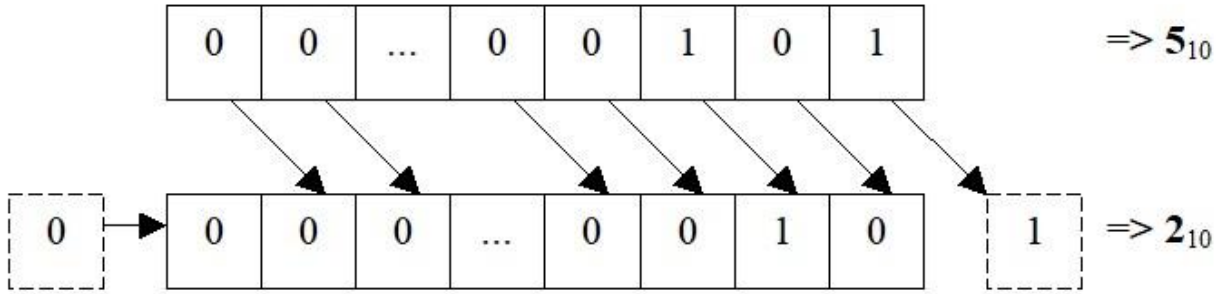
Пример: Операторы сдвига влево <<, вправо >>

```
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’)

Логический тип (Boolean)

```
x = True
y = False
print(x)
print(y)
print(str(x))
print(str(y))
print(int(True))
print(True + True)
```

Операторы сравнения

- "==" (равно)
  ">=" (больше или равно)
  "<=" (меньше или равно)
  "!=" (не равно)
  "<" (меньше)
  ">" (больше)
- Примечание: Когда мы хотим стравнить что две переменные равны то мы делаем так:
- >> weight_one = 100
- >> weight_two = 100
- >> weight_one == weight_two → true
- >> weight_one != 90 → true
- >> weight_one = weight_two ← не правильно !!

2 полезные функции символов

ord(s) – код символа s

Напечатайте:

ord(‘a’)

ord(‘z’)

ord(‘a’)

ord(‘я’)

ord(‘ё’)

А затем chr() от любых чисел, например:

```
for i in range(1102, 1110):
    print(i, chr(i))
```

Что напечатает: print(chr(ord(‘ы’)))

Задание

Определите коды больших латинских букв от A до Z,

напечатайте в цикле пары (буква и ее код).

Используйте функции chr и ord, например, определить код A можно с помощью ord(‘A’)

Сравнение строк при помощи == и !=

- >>>language = 'chinese'
- >>>print(language == 'chinese') → True
- >>>print(language != 'chinese') → False

- >>> 'chinese' > 'italiano'
- Ответ: **?**

Логические операторы

- **AND** - логическое И
- **OR** - логическое ИЛИ
- **NOT** - логическое отрицание
- **IN** - возвращает истину, если элемент присутствует в последовательности, иначе ложь.
- **NOT IN** - возвращает истину если элемента нет в последовательности.
- **IS** - проверка идентичности объекта

Таблица истинности

NOT

x	x'
0	1
1	0

AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Python - Logical Operators

- not

x	not x
False	True
True	False

- and

x	y	x and y
False	False	False
False	True	False
True	False	False
True	True	True

- or

x	y	x or y
False	False	False
False	True	True
True	False	True
True	True	True

http://inderpsingh.blogspot.com/

Применение логических операторов

- x = 10
- y = 20
- if x > 0 and y > 0:
- 	print('Положительные числа')
- if x > 0 or y > 0:
- 	print('Хотя бы одно положительное’)
- if 0 < x < 100:
  - print(“В интервале от 0 до 100”)
- if x > 0 or y / 0:
- 	print('Что будет?’)
- if x > 0 and y / 0:
- 	print(‘А теперь?')

Задание

- Определите, является ли введенный год високосным.
- Год является високосным, если его номер кратен 4, но не кратен 100, или если он кратен 400.

Таблица приоритетов операций

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

Вывод

- Чтобы не запутаться в приоритетах операций ставьте в выражении круглые скобки ()

- # Тестирование порядка выполнения выражения(слева направо)
- print(4 * 7 % 3)
- # Результат: 1

- print(2 * (10 % 5))
- # Результат: 0

None

Если надо создать переменную, но непонятно, что ей присвоить, то можно присвоить None, например, нельзя использовать 0.

Можно проверить, что ей не было ничего не присвоено, например:

```
a = None
if a == None:   # лучше писать if a is None
        print(“Ничего нет”)
else:
        print(f”Значение a = {a}”)
```

Например:

```
abc = {1:11, 2:22}
x = abc.get(3)
x is None
```

Задание

Введено слово (латинские буквы в нижнем регистре).

Перетасуйте его буквы, чтобы гласные и согласные шли по очереди. Если это невозможно, то выдайте “Impossible!”

Гласными будем считать только a, e, i, o, u. Остальные – согласные.

Например:

apple - >papel

idea -> Impossible!

sorted -> Impossible!

idiot -> idito