Занятие 6

Множества

Что напечатает?

```
x = 10
print(0 < x < 100)
print(0 < x == 10)
print(10 == x == 10)
print(10 == x in [10])

print(False == (False in [False]))
print((False == False) in [False])
print(False == False in [False])
```

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

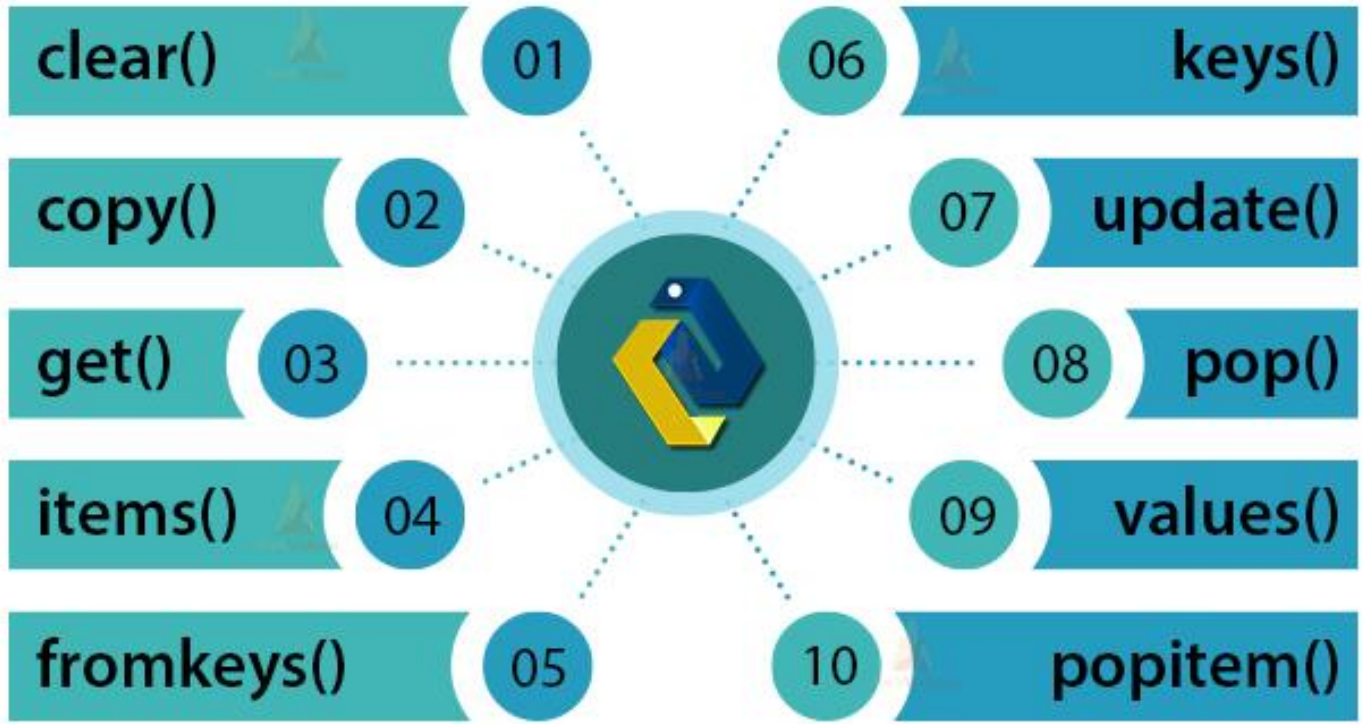
f[0] = 1, f[1] = 1

f[k] = f[k - 1] + f[k – 2]

Пример последовательности:

1, 1, 2, 3, 5, 8, 13, 21, …

Python Dictionary Methods



pop(key[, default])

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

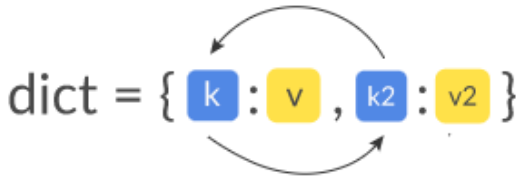
```
>>> x = {'one': 0, 'two': 20, 'three': 3}

>>> 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: 'ten'
```

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
```

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
```

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

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}**

Множества set

Множество — неупорядоченный набор элементов.

Каждый элемент в множестве уникален (т. е. повторяющихся элементов нет) и неизменяем.

```
>>> data_scientist = set(['Python','SQL','Pandas','Git'])
```

```
>> data_engineer = set(('Python','Java','Hadoop','SQL','Git’ ))
```

Задание множества

Что будет при дублировании значения ?

```
>>> data_scientist = set(['Python','R','R','SQL','Pandas','Git'])
>>> type(data_scientist)
<class 'set'>
```

Создайте множество из списка, в котором есть повторяющиеся элементы, и напечатайте его

Задание множества

Мы также можем создать множество с элементами разных типов. С неизменяемыми!

Например:

```
>>> mixed_set = {2.0, "Nicholas", (1, 2, 3)}
>>> print(mixed_set)
```

Что будет напечатано?

```
>>> mixed_set = {2.0, "Nicholas", (1, 2, 3), 1, 2, 3}
>>> print(mixed_set)
```

А теперь?

Создание пустого множества

- Пустой список lst = []
- Пустой словарь dct = {}

- Пустое множество sss = set()
- Если написать sss = {}, то будет пустой словарь!!!

Задание

Дан список lst = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]

Какова его длина?

Мы хотим узнать, сколько в списке уникальных элементов.

Преобразуем его во множество

```
mno = set(lst)
```

Какова его длина?

Давайте напечатаем множество:

```
print(mno)
```

Итерирование множества

```
months = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}

for m in months:
    print(m)
```

В каком порядке будут напечатаны эти месяцы?

```
# проверка на членство в множестве
print("May" in months)
```

Элементы во множествах хранятся в неупорядоченном виде, и к ним нельзя обратиться по индексу, использовать срезы.

Скорость

Операции с множествами быстрее, чем списки и кортежи, но памяти тратится намного больше

```
from time import time
number = 15000
my_set = set(range(number))
my_list = list(range(number))
my_tuple = tuple(range(number))
t = time()
for i in range(number):
    if i in my_list:
        pass
print(f"Операция со списком: {time() - t} секунд")
```

Операция со списком: 1.179133653640747 секунд

Операция с кортежем: 1.440788984298706 секунд

Операция со множеством: 0.0028142929077148438 секунд

Напишите программу, которая вычисляет времена выполнения схожих операций

Задание

Дано множество, состоящее из чисел

Его можно ввести одной строкой с пробелами между числами
с помощью оператора tes = set(map(int, input().split()))

Напечатайте среднее арифметическое введенных чисел.

Воспользуйтесь функциями sum и len

Важные методы

Функция	SET	LIST	DICT
Добавить новое значение	Add	Append	Setdefault
Копировать	Copy	Copy	Copy
Выдать значение и удалить его	Pop	Pop	Pop
Очистить	Clear	Clear	Clear
Добавить такой же объект	Union	Extend	Update
Длина	Len	Len	Len
Новый объект (функция)	Set()	List()	Dict()
Новый объект (скобки)	{1,2,3,}	[1,2,3, 1,2,3]	{1:11, 2:22, 3:33}

Добавление элементов

```
months = set(["Jan", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])


months.add("Feb")
print(months)
```

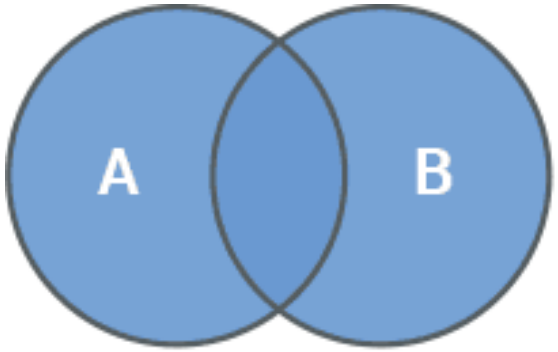
Что будет напечатано? На каком месте будет Feb?

Удаление элемента из множеств

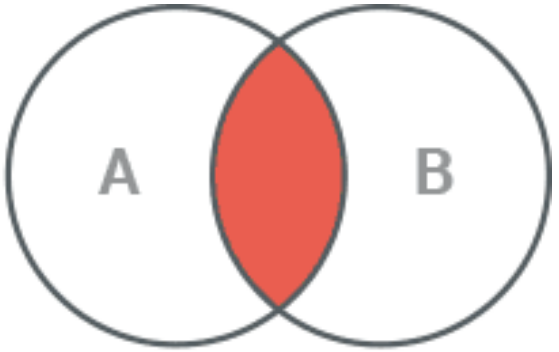
```
>>>num_set = {1, 2, 3, 4, 5, 6}
>>>num_set.discard(3)
>>>print(num_set)
{1, 2, 4, 5, 6}
```

Метод num_set.remove(7) аналогичный, но вызовет ошибку при отсутствии элемента.

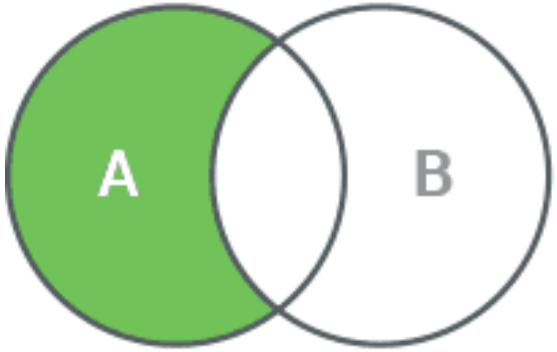
Из теории множеств



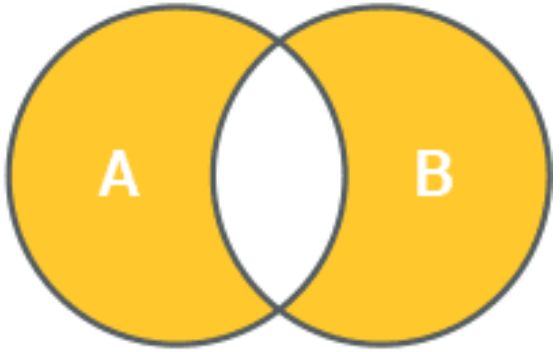
Union



Intersection



Difference



Symmetric Difference

Объединение множеств

- >>> months_a = set(["Jan", "Feb", "March", "Apr", "May", "June"])
- >>> months_b = set(["July", "Aug", "Sep", "Oct", "Nov", "Dec"])
-
- >>> all_months = months_a.**union**(months_b)
- print(all_months)
- {'Oct', 'Jan', 'Nov', 'May', 'Aug', 'Feb', 'Sep', 'March', 'Apr', 'Dec', 'June', 'July'}

Задание

Напишите программу, которая получает на вход строку,
и определяет, является ли строка панграммой
(т.е. содержатся ли в ней все **33** буквы русского алфавита).

union() или оператор |

Объединение может состоять из более чем двух множеств

x = {1, 2, **9**}

y = {4, 5, 6}

z = {7, 8, **9**}

output = x.**union**(y, z)

print(output)

{1, 2, 9, 4, 5, 6, 7, 8}

print(x | y | z)

Пересечение множеств

x = {1, 2, 3}

y = {4, 3, 6}

z = x.**intersection(**y)

print(z) #

x = {1, 2, **3**}

y = {4, **3**, 6}

print(x & y)

3

Разница между множествами

```
set_a = {1, 2, 3, 4, 5}
set_b = {4, 5, 6, 7, 8}

diff_set = set_a.difference(set_b)
print(diff_set)
{1, 2, 3}

print(set_a - set_b)
```

Симметричная разница

set_a = {1, 2, 3, 4, 5}

set_b = {4, 5, 6, 7, 8}

symm_diff = set_a.**symmetric_difference**(set_b)

print(symm_diff)

{1, 2, 3, 6, 7, 8}

print(set_a ^ set_b)

Сравнение множеств

Чтобы проверить, является ли множество A дочерним от B, мы можем выполнить следующее.

```
months_a = set(["Jan", "Feb", "March", "Apr", "May", "June"])

months_b = set(["Jan", "Feb", "March", "Apr", "May", "June", "July", "Aug", "Sep", "Oct", "Nov", "Dec"])


# Чтобы проверить, является ли множество B подмножеством A

subset_check = months_a.issubset(months_b)


# Чтобы проверить, является ли множество A родительским множеством

superset_check = months_b.issuperset(months_a)


print(subset_check)

print(superset_check)
```

Задание

Напишите программу, которая получает **n** слов, и вычисляет количество **уникальных** символов во всех словах.

Frozenset – неизменяемые множества

```
fro = frozenset([1,2,3])
print(fro)
fro.add(4) - ?
```

Зато frozenset может входить в set, в отличие от set

Функции

Понятие функции

Функция в **Python** - объект, принимающий аргументы, реализующий какие-то законченные действия и возвращающий результат.

Вход в функцию - это передача ей аргументов - данных, полученных во внешней части программы.

Тело функции - получив данные, функция должна их как-то обработать: выполнить некоторые действия, вычислить какое-то значение.

Выход из функции - значение, вычисленное блоком кода данной функции и передаваемое во внешнюю часть программы.

Входные данные называют параметрами, а выходные - возвращаемым значением.

Впрочем, функция может и не принимать никаких параметров.

Что принимает в качестве параметров и что возвращает функция в результате своей работы, определяет программист.

Пример:

```
#Определение функции:
def summ(x, y):
    result = x + y
    return result

#вызов функции
a = 100
b = 50
answer = summ(a, b)

print(answer)
150
```

Роль функции в программировании

1. Сокращение кода

Код, который повторяется можно перенести в функцию и использовать её тогда,

когда нужно выполнить код, который находится внутри этой функции.

2. Логическое разделение программы

Мы можем выделить определённое сложное действие (например перемножение матриц) в отдельную функцию, чтобы оно не мешалось в коде, даже если используем её один раз за всё время выполнения программы.

3. Проще тестировать

4. Более эффективная организация труда команды разработчиков

Можно разрабатывать проект большим количеством программистов, каждый из которых отвечает за свои функции

Определение функции

```
# объявление функции my_function()

def my_function([параметр1, параметр2,..]):
    # тело функции

    # возвращаемое значение
    return result  # необязательно

# вызов функции
my_function([аргумент1 ,аргумент2,...] )
```

или

```
result = my_function([аргумент1 ,аргумент2,...] )

type(my_function)
<class 'function'>   - еще один тип в Python
```

Пример функции

Перевод градусов по шкале Фаренгейта в градусы по шкале Цельсия осуществляется по формуле C = 5/9*(F - 32).

Напишем функцию, которая осуществляет перевод:

```
def convert_to_celsius(temp):
    result = (5 / 9) * (temp - 32)
    return result


x = convert_to_celsius(32)
print(x)
```

Задание

Напишите функцию, которая переводит градусы по Цельсию в Фаренгейты

Выведите формулу

Сформируйте функцию convert_ to_ f(temp)

Проверьте ее на различных значениях: 0, 5, 10

Задание 6-1

Написать программу, которая переводит строку римских цифр в десятичное число.

Римские цифры:

I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000, IV= 4, IX = 9, XC = 90, XL = 40, CD = 400, CM = 900

Пример:

MMXXIII = 2023, MMXXIV = 2024, MCMXVII = 1917, MXMLX1 = 1961, MM = 2000, MDXXXLXII = 1862

Подсказка. Можно использовать функцию ‘abcde’.startswith(‘ab’), которая выдает True, если строка ‘abcde’ начинается с ‘ab’

Задача 6-2

Напишите программу, которая:

- Получает на вход две строки, в которых перечисляются книги, прочитанные двумя учениками.
- Выводит количество книг, которые прочитали **оба** ученика.

Пример ввода:

Война и мир, Над пропастью во ржи, Мастер и Маргарита, Идиот

Евгений Онегин, Идиот, Мастер и Маргарита, Война и мир

Ответ: 3

Задача 6-3

Напишите программу, которая принимает на вход строку из символов и печатает три строки:

одну строку из букв,

вторую из цифр,

третью из прочих символов.

Все строки состоят из уникальных символов.

Например:

Ввод: ab18.,cab=561:xz:

Вывод:

a b c x z

1 8 5 6

. , = :