Занятие 6

Множества

Функции

Начнем с ...

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

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])
```

Отладчик в PyCharm

- Точка останова
- Значения переменных в точке останова
- Продвижение вперед до следующей точки останова – F9
- Продвижение вперед на следующую строку кода – F8

```
lst = list(map(int, input('--->').split()))
total = 0
for i in range(len(lst)):
    total += lst[i + 1] - lst[i]
    print(i, total)

print(f"Сумма = {total}")
```

Числа до 256 и немного больше

явное указание

```
a = 255
b = 255
print(a == b, a is b, 255)

a = 256 # 256
b = 256 # 256
print(a == b, a is b, 256)

a = 257 # 257
b = 257 # 257
print(a == b, a is b, 257)
```

```
True True 255
True True 256
True True 257
```

неявное указание

```
a = 255
b = 255
print(a == b, a is b, 255)

a = a + 1 # 256
b = b + 1 # 256
print(a == b, a is b, 256)

a = a + 1 # 257
b = b + 1 # 257
print(a == b, a is b, 257)
```

```
True True 255
True True 256
True False 257
```

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','R','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?

Преобразуем его во множество

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

Какова его длина?

Давайте напечатаем множество:

```
print(mno)
```

Итерирование множества

- months = {"Jan", "Feb", "March", "Apr", "May", "June", "July", "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

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

SET	LIST	DICT
Add	Append	Setdefault
Copy	Copy	Copy
Pop	Pop	Pop
Clear	Clear	Clear
Union	Extend	Update
Len	Len	Len
Set()	List()	Dict()
{1,2,3,}	[1,2,3, 1,2,3]	{1:11, 2:22, 3:33}

Добавление элементов

- months = set(["Jan", "March", "Apr", "May", "June", "July", "Aug", "Sep", "Oct", "Nov", "Dec"])
-
- months.add("Feb")
- print(months)

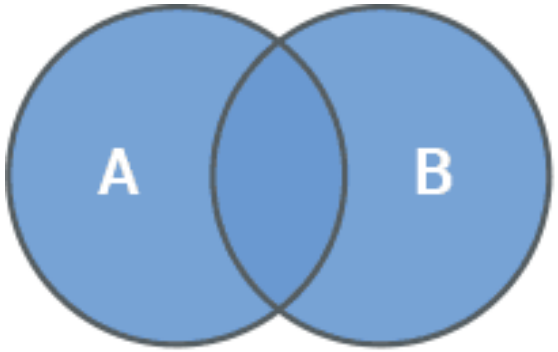
- Что будет напечатано? На каком месте будет Февраль?

Удаление элемента из множеств

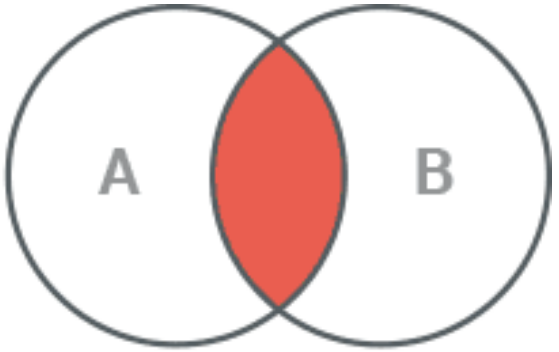
- >>>num_set = {1, 2, 3, 4, 5, 6}
- >>>num_set.discard(3)
- >>>print(num_set)
- {1, 2, 4, 5, 6}

- Метод num_set.remove(7) аналогичный но вызовет ошибку при отсутсвии элемента.

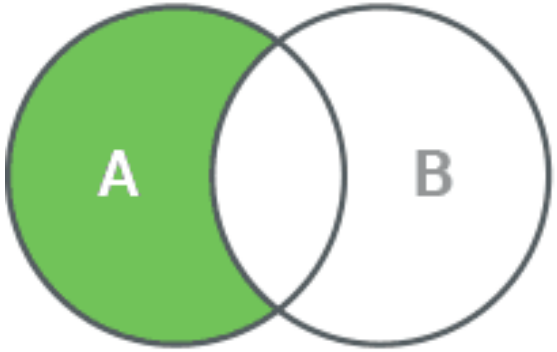
Из теории множеств



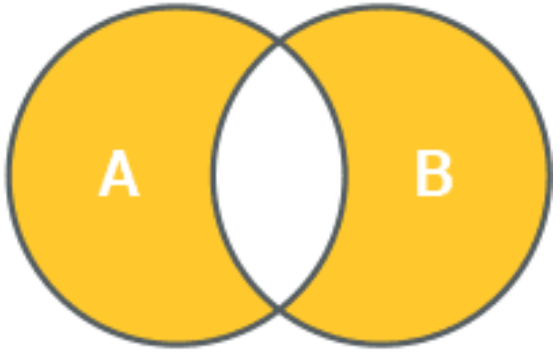
Union



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 следующую операцию:
- 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

Вызов функции

- Можно ли так вызвать функцию ?
- z = summ(10, 20)

```
def summ(x, y):
    return x + y
```

- **NameError:** name 'summ' is not defined

- z = sum(10, 20) # а так можно

Что вернет функция без return:

- #Определение функции:
- **def** *summ*(x, y):
- result = x + y
-
- #вызов функции
- answer = *summ***(**100, -50**)**
- **print(**answer**)**

- answer is None

Что вернет функция в отсутствии аргументов ?

- #Определение функции:
- **def** *summ*(x, y):
- result = x + y
- return result;
-
- #вызов функции
- answer = *summ***(**100**)**
- **TypeError:** summ() missing 1 required positional argument: 'y'

Задание

Напишите программу, которая содержит функцию вычисления подоходного налога (13%) от суммы дохода, передаваемой как параметр.

Программа в цикле вводит доход, вызывает функцию и печатает подоходный налог.

Параметры по умолчанию

- Для некоторых параметров в функции можно указать значение по умолчанию,
- таким образом если для этого параметра не будет передано значение
- при вызове функции, то ему будет присвоено значение по умолчанию.
- def premium(salary, **percent=10**):
  p = salary * percent / 100
  return p

  result = premium(60000) # указана только salary, percent = 10 по умолчанию
  print(result)
- 6000.0
- print(premium(60000,20)) # указаны и salary, и percent
- 12000.0

Пустая функция

- def empty(var1, var2):
  pass

- result = empty(8, 10)
  print(result)
- **None**

Именованные параметры

- Иногда происходит такая ситуация, что функция требует большое количество аргументов.
- Пример:
- **def** *my_func*(arg1, arg2, arg3, arg4, arg5, arg6):
  pass # оператор, если кода нет
- result = *my_func*(1, 2, 3, 4, 5, 6)
- В такой ситуации код не всегда удобно читать и тяжело понять, какие переменные к каким параметрам относятся.
- result = my_func(**arg2=2**, **arg1=1**, arg4=4, arg5=5, arg3=3, arg6=6)

Функция с неограниченным количеством позиционных аргументов

- ***args** - произвольное число позиционных аргументов

- **def** manyargs(var1, ***args**):
  print(type(args))
  print(args)

  result = manyargs(**4**, 9, 1, 3, 3, 1)
- <class '**tuple**'>
- (9, 1, 3, 3, 1)

Функции с неограниченным количеством именнованных аргументов

- ****kwargs** - произвольное число именованных аргументов.
- def manykwargs(****kwargs**):
  print(type(kwargs))
  print(kwargs)
- manykwargs(name=’Piter’, age=20)
- <class **'dict'**>
- {'name': 'Piter', 'age': 20}

- Можно ли мне по другому назвать параметр ****kwargs ?**
- Можно, только вас никто не поймет!

Все вместе

- ***args -** произвольное число позиционных аргументов
- ****kwargs** - произвольное число именованных аргументов

- Параметр с одним префиксом * может захватывать любое количество позиционных аргументов в кортеж.
- Параметр с двойным префиксом ** может захватывать любое количество ключевых аргументов в словарь.

- def many_all(var1, ***args**, ****kwargs**):
  print(var1)
- print(**args**)
  print(**kwargs**)
- many_all(10, 34, 77, name='Piter', age=20)
- 10
- (34, 77)
- {'name': 'Piter', 'age': 20}

Задание

Доработайте функцию, чтобы одним из аргументов ее была ставка налога (по умолчанию 13%).

Вызовите ее только с одним параметром – суммой дохода.

Вызовите ее с двумя параметрами – суммой дохода и ставкой ПН (13 или 15 или 20).

Задание

Напишите функцию, которая принимает в качестве аргумента список, состоящий из слов.

Например, def uni_let(lst):

Результатом функции получается кортеж, состоящий из двух элементов:

1. Строка из всех уникальных букв, отсортированная в алфавитном порядке
2. Количество таких уникальных букв.

Т.е. в конце функции должен стоять оператор return string, number

Проверьте ее работу на списке из нуля слов, из одного слова, из двух и т.д.