АКАДЕМИЯ АЙТИ

День 15

academyit.ru

* Строки. Методы и функции. Использование срезов.
* Кортеж. Основные операции с кортежем. Распаковка кортежа.
* Список. Основные операции со списком.
* Словарь. Основные операции со словарем.
* Множества. Основные операции с множеством

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur

Поверка домашнего задания

Задача 5

У вас есть девять цифр: 1, 2, …, 9. Именно в таком порядке.
Вы можете вставлять между ними знаки «+», «-» или ничего.
У вас будут получаться выражения вида 123+45-6+7+89.
Найдите все из них, которые равны 100.

Словари

Словари

- Словарь (dict) состоит из набора ключей и соответствующих им значений
- Значения могут быть любыми объектами
- Ключи могут быть любыми неизменяемыми объектами, в частности числами, строками, кортежами.
- Список или множество не могут быть ключом.
- Одному ключу соответствует ровно одно значение. Но одно и то же значение, в принципе, можно сопоставить разным ключам.

Создание словаря

```
authors = {'Python': 'Гвидо ван Россум',
           'C#': 'Андерс Хейлсберг',
           'Java': 'Джеймс Гослинг',
           'C++': 'Бьёрн Страуструп'}

print(authors)
print(authors['Python'])

d = dict{[("кот", 10),("и",100), ("снег", 2)])
d1 = {} # пустой словарь
d2 = dict()
```

Генераторы словарей

```
squares = {i: i**2 for i in range(5)}
print(squares )

counts =
{c: word.count(c) for c in set(word)}
print(counts)
```

Доступ к элементам словаря

Обращение – по ключу

```
d["кот"] = d["кот"] + 1
```

Если ключ не найден – ошибка. Для проверки, есть ли ключ, можно использовать in.

```
s = input()
if s in d:
  print(d[s])
else:
  print("Нет ключа")
```

Другой способ – метод get. Второй аргумент – значение, которое вернет метод, если нет такого ключа.

```
d[s] = d.get(s, 0)
```

Действия с элементами

Добавление пары ключ – значение

```
d["собака"] = 10
```

Изменение значения

```
d["собака"] = 20
```

Удаление ключа

```
del d["кот"]
```

Извлечение последнего добавленного элемента

```
d.popitem()
```

Очистка словаря

```
d.clear()
```

Перебор элементов

Перебор по ключам

```
for k in d:
    print(k, d[k])
```

Преобразование в список

```
for v in d.values(): # значения
    print(v)
for v in d.keys(): # ключи
    print(v)
for k, v in d.items(): # ключ-значение
    print('k =', k, ', v =', v)
```

Несколько значений для ключа

Можно использовать словарь списков или словарь словарей

```
my_dict =
{'swear':['клясться', 'ругаться'],
'dream' : ['спать', 'мечтать']}
for s in my_dict['swear']:
  print(s)
```

Пример

```
Capitals = {'Russia': 'Moscow',
'Ukraine': 'Kiev', 'USA': 'Washington'}
Countries = ['Russia', 'France', 'USA',
'Russia']
for country in Countries:
  if country in Capitals:
    print(country + ': ' + Capitals[country])
  else:
    print('В базе нет страны ' + country)
```

Когда нужны словари?

1. Подсчет числа каких-то объектов. Ключи - объекты, а значения — их количество.
2. Хранение каких-либо данных, связанных с объектом. Ключи — объекты, значения — связанные с ними данные. Пример – номер телефона.
3. Установка соответствия между объектами (например, “родитель—потомок”). Ключ — объект, значение — соответствующий ему объект.
4. Если нужен обычный массив, но максимальное значение индекса элемента очень велико, и при этом будут использоваться не все возможные индексы (так называемый “разреженный массив”).

Функции: дополнительные возможности

Функции.

Довольно красивой особенностью Python-а является возможность определения функции, которая принимает разные количества аргументов. Определим функцию `printer`, которая принимает любое количество аргументов — все аргументы записываются в `tuple args`. Затем функция печатает по порядку все аргументы:

```
def printer(*args):
    print(type(args))
    for argument in args:
        print(argument)

printer(1, 2, 3, 4, 5)
```

```
<class 'tuple>
1
2
3
4
5
```

Функции.

Точно так же это работает в случае со словарями, в данном случае мы можем определить функцию `printer`, которая принимает разное количество именованных аргументов. При этом переменная `kwargs` будет иметь тип `dict`.

```
def printer(**kwargs):
    print(type(kwargs))
    for key, value in kwargs.items():
        print('{}: {}'.format(key, value))

printer(a=10, b=11)
```

```
<class 'dict'>
a: 10
b: 11
```

```
isinstance(object, classinfo)
```

Параметры:

- `object` - объект, требующий проверки,
- `classinfo` - класс, кортеж с классами или рекурсивный кортеж кортежей или с версии Python 3.10 может быть объединением нескольких типов (например `int | str`).

Возвращаемое значение:

- `bool`.

Описание:

Функция `isinstance()` вернет `True`, если проверяемый объект `object` является **экземпляром** указанного класса (классов) или его подкласса (прямого, косвенного или виртуального).

Если объект `object` не является экземпляром данного типа, то функция всегда возвращает `False`.

Функцией `isinstance()` можно проверить класс, кортеж с классами, либо рекурсивный кортеж кортежей. Другие типы последовательностей аргументом `classinfo` не поддерживаются.

Если аргумент `classinfo` не является классом, либо кортежем с классами, а с версии Python 3.10 записью, объединяющей нескольких типов (например `int | str`), то возбуждается исключение `TypeError`.

Функции.

Точно так же мы можем разыменовывать (разворачивать) словари, используя **:

```
def printer(**kwargs):
    print(type(kwargs))
    for key, value in kwargs.items():
        if isinstance(value, dict):
            print(str(key)+': ',end=' ')
            printer(**value)
        else:
            print('{}: {}'.format(key, value))

something = {'a':1,'inner':{'a_inner':2}}
printer(**something)
```

```
<class 'dict'>
a: 1
inner:  <class 'dict'>
a_inner: 2
```

Это используется практически везде и позволяет вам определять очень гибкие функции, которые принимают различное количество аргументов — именованных и позиционных

Функции.

Функции в Python — это такие же объекты, как и, например, строки, списки или классы. Их можно передавать в другие функции, возвращать из функций, создавать на лету — то есть это объекты первого класса.

```
def caller(func, params):
    return func(*params)


def printer(greeting, name):
    print(f'{greeting}, {name}!')


caller(printer, ['Hi', 'Ivan'])
```

Hi, Ivan!

Функции.

Итак, функции можно передавать в функции. Также их можно создавать внутри других функций.

```
def get_multiplier():
    def inner(a, b):
        return a * b

    return inner


multiplier = get_multiplier()
multiplier(10, 11)
```

110

Т.к. мы вернули другую функцию, в переменной `multiplier` теперь хранится функция `inner`:

```
print(multiplier.__name__)
```

inner

Функции.

Давайте попробуем определить функцию `inner`, которая будет принимать один аргумент и умножать его всегда на то самое число, которое мы передали в `get_multiplier`. Например, мы передаем `get_multiplier` двойку и получаем функцию, которая всегда умножает переданный ей аргумент на двойку. Эта концепция называется "замыканием".

```
def get_multiplier(number):
    def inner(a):
        return a * number

    return inner


multiplier_by_2 = get_multiplier(2)
multiplier_by_2(10)
```

```
20
```

Этот приём очень важен и в дальнейшем будет использоваться в декораторах.

Рекурсия

Рекурсивные функции

Рекурсия — это способ определения значения через одно или несколько предыдущих значений

Числа Фибоначчи:

- $F_1 = F_2 = 1$
- $F_n = F_{n-1} + F_{n-2}$ при $n > 2$

1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Рекурсивная функция — это функция, которая вызывает сама себя напрямую или через другие функции.

Пример. Факториал.

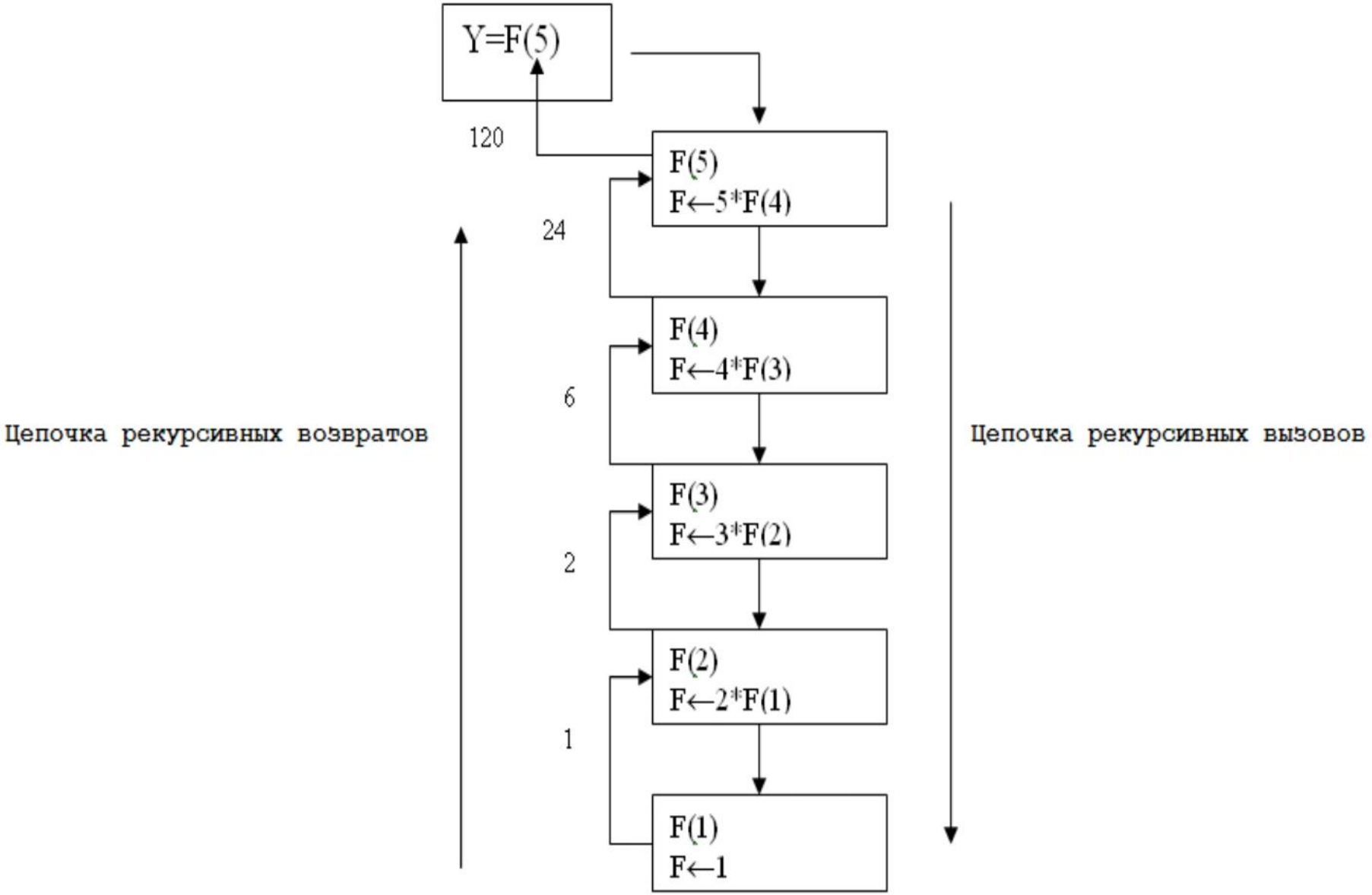
$n! = 1 * 2 * 3 * \ldots * n$

$F(1)=1$

$F(n)=n*F(n-1)$

```
def fact(n):
    if n == 1:
        return 1
    else:
        return n * fact(n - 1)
```

Вычисление факториала



Разработка рекурсивных функций

1) Какой случай (для какого набора параметров) будет не рекурсивным и что функция возвращает в этом случае?
2) Как свести задачу для какого-то набора параметров к задаче для другого набора параметров с меньшими значениями?

Задачи, которые необходимо решить с помощью рекурсии:

1. Вычислить сумму элементов набора чисел
2. Вычислить количество отрицательных чисел в наборе
3. Вычисление суммы чисел с поддержкой вложенных списков
4. Возврат ряда Фибоначчи
5. Реверсирование числа
6. Возведение числа x в степень y
7. Определение максимального элемента списка
8. Перевод из десятичной системы исчисления в двоичную

Домашнее задание

Задача №3760. Словарь синонимов

Вам дан словарь, состоящий из пар слов. Каждое слово является синонимом к парному ему слову. Все слова в словаре различны. Для одного данного слова определите его синоним.

Входные данные

Программа получает на вход количество пар синонимов N. Далее следует N строк, каждая строка содержит ровно два слова-синонима. После этого следует одно слово.

Выходные данные

Программа должна вывести синоним к данному слову.

Примечание

Эту задачу можно решить и без словарей (сохранив все входные данные в списке), но решение со словарем будет более простым.

Примеры

входные данные
3 Hello Hi Bye Goodbye List Array Goodbye
выходные данные
Bye

Задача №3764. Частотный анализ

Дан текст. Выведите все слова, встречающиеся в тексте, по одному на каждую строку. Слова должны быть отсортированы по убыванию их количества появления в тексте, а при одинаковой частоте появления — в лексикографическом порядке.

Указание. После того, как вы создадите словарь всех слов, вам захочется отсортировать его по частоте встречаемости слова. Желаемого можно добиться, если создать список, элементами которого будут кортежи из двух элементов: частота встречаемости слова и само слово. Например, `[(2, 'hi'), (1, 'what'), (3, 'is')]`. Тогда стандартная сортировка будет сортировать список кортежей, при этом кортежи сравниваются по первому элементу, а если они равны — то по второму. Это **почти** то, что требуется в задаче.

Входные данные

Вводится текст.

Выходные данные

Выведите ответ на задачу.

Примеры

входные данные
hi hi what is your name my name is bond james bond my name is damme van damme claude van damme jean claude van damme
выходные данные
damme is name van bond claude hi my james jean what your

Задание 3

Напишите функцию **def f7(list1)**, которая будет выполняться рекурсивно и определять максимальное число из списка.

Ввод: list1 = [400, 1300, 700, 561, 123]
Вывод: 1300

Задание 4

Напишите функцию **def f3(list1)**, которая будет выполняться рекурсивно и вычислять сумму чисел из списка (с поддержкой вложенных списков).

Ввод: list1 = [-3, 8, 4, -7, [-1, 8, [2, [-4, 3]]]]
Вывод: 10

АПЛАНА
ГРУППА КОМПАНИЙ

Входит в ГК Аплана

АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning и очное обучение

Направления обучения:

- Информационные технологии
- Информационная безопасность
- ИТ-менеджмент и управление проектами
- Разработка и тестирование ПО
- Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск, Хабаровск, Красноярск, Тюмень, Нижний Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис в Москве

Разработка программного обеспечения и информационных систем

Ежегодные награды Microsoft, Huawei, Cisco и другие

Программы по импортозамещению

Ресурсы более 400 высококлассных экспертов и преподавателей

Сеть региональных учебных центров по всей России

Крупные заказчики

СБЕР

РЖД

Билайн

TATNEFT

НОРНИКЕЛЬ

100+
сотрудников

АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:
Москва, Варшавское шоссе 47, корп. 4, 7 этаж
Тел: +7 (495) 150-96-00
academy@it.ru
academyit.ru