ЛАБОРАТОРНАЯ РАБОТА

Тема: ***Использование списков в Питоне***

Цель работы: Познакомиться с простейшими свойствами и приемами использования списков.

Содержание

Списки как составной тип данных

Все языки программирования обеспечивают возможность использования *составных типов данных*. Обычно это *массивы*, составляемые из данных *одного типа*, либо *структуры* (записи), которые строятся из данных *разных типов*.

В Питоне используется несколько составных типов данных, среди которых наиболее универсальный — списки (*англ.* `list`).

Список в Питоне — это *индексированная последовательность* данных, которая в рамках программы рассматривается как единый объект данных. Данные, составляющие список, называются *элементами* списка. Количество элементов в списке называется *длиной списка*.

Важно отметить, что элементы списка могут принадлежать как к одному, так и к разным типам. В том числе, элементами списков могут быть даже *другие списки*.

Синтаксическая конструкция списков в программах:

- группа элементов данных,
- перечисленных через *запятые*,
- помещенная в *квадратные скобки*.

Например, переменная `my_list` будет ссылаться на список из трех элементов
`>>> my_list = [ "a", 1.5, "bcd" ].`

Пустым называется список, в котором нет элементов. Он записывается в виде пары пустых квадратных скобок.

Как и строки, списки — это тип последовательностей Питона. Принципиальное различие между ними в том, что строки состоят из символов, а списки — из данных любых типов.

Так как элементы списков индексированы, доступ к ним можно осуществлять с помощью операции индексации. Можно также строить срезы по тем же правилам, что и для строк.

```
>>> my_list[ : : -1]
['bcd', 1.5, 'a']
```

ЗАДАНИЕ 1 (*Простейшие свойства списков*)

1. ***Свойства непустых списков.***

Переменной `my_list` присвоен список

```
>>>  my_list = ["победа","ничья", "поражение"]
```

С помощью функции `type()` убедиться, что переменная `my_list` действительно ссылается на список.

С помощью функции `len()` вычислить длину списка.

Обратиться к членам списка так, чтобы получить значение `"ничья"`.

Убедиться, что свое значение внутреннего идентификатора `id` есть у каждого элемента списка и всего списка в целом.

2. ***Свойства пустого списка.***

а) Убедиться, что конструкция **`[]`** описывает
– элемент данных типа "список",
– последовательность, не имеющую элементов.

б) Построить условное выражение, которое дает результаты в виде строк "`пустой`" (условие выполнено) или "`не пустой`". В роли логического выражения (для записи условия) использовать пустой список. Объяснить результат вычисления выражения.

в) С помощью функций `type()` и `len()` проверить, допустима ли конструкция

`[ [], [] ]`

Какой тип данных она описывает? Является ли конструкция пустым списком? Сколько в ней элементов и что является её элементами? Одинаковые или разные значения внутреннего идентификатора `id` имеют элементы этой последовательности?

Списки как изменяемые типы данных

В Питоне, в отличие от строк или чисел, списки являются *изменяемыми* данными. Это значит, что они обладают *методами*, которые позволяют менять хранимое объектом данных значение. У строк и чисел таких методов нет.

а) "Сложение" списков.

Знак плюс (+) между двумя списками позволяет добавить элементы второго списка *в конец* первого.

```
>>> [1, 2]+[3, 4, 5]
[1, 2, 3, 4, 5]
```

Замечание.

Аналогичную операцию слияния списков можно выполнить с помощью *метода* списковых объектов `lst1.extend(lst2)`. Однако метод применим только, если `lst1` является *переменной* спискового типа.

```
>>> lst1=[1,2]
>>> lst1.extend([3,4,5])
>>> lst1
[1, 2, 3, 4, 5]
```

б) Умножение списка на число.

Операция умножения (`*`) между списком и числом позволяет создать *новый* список, в котором элементы исходного списка будет повторены заданное число раз.

```
>>> [1,2]*3   # или 3*[1,2]
[1, 2, 1, 2, 1, 2]
```

в) Добавление элемента в конец списка.

Метод списковых объектов `append(elem)` позволяет добавить один элемент `elem` в конец списка, для которого вызывается метод.

```
>>> x = [1,2]
>>> x.append(3)
>>> x
[1, 2, 3]
```

г) Вставка элемента перед заданным.

Метод `insert(position, elem)` позволяет вставить в список один *элемент* `elem` перед элементом с индексом `position` (т.е. в любое место списка). Например,

```
>>> x=['b','d']
>>> x.insert(1,'c')      # перед 'd'
>>> x.insert(0,'a')      # перед 'b'
>>> print(x)
['a', 'b', 'c', 'd']
```

Целочисленный аргумент `position` является индексом элемента, перед которым будет произведена вставка.

ЗАДАНИЕ 2 (*Простейшие операции со списками*)

1. ***Добавление элементов в список.***

а) Создать список из строк "хорошо" и "удовлетворительно" и сохранить его в переменной mark1.

Используя print вывести mark1 и значение соответствующего ей id.

б) С помощью метода append() добавить в конец списка mark1 значение "неудовлетворительно".

С помощью метода insert() добавить в начало списка mark1 значение "отлично".

Используя print, вывести значения mark1 и соответствующего ей id.

Что можно сказать про значение mark1? Переменная ссылается на прежний объект с обновленным значением или созданный при добавлениях элементов новый объект?

в) С помощью среза выделить из списка mark1 список из первых трех элементов и вывести его на экран.

2. Повторение элементов списка.

а) Создать список из строк "чётный" и "нечётный" и сохранить его в переменной mark2.

Используя print вывести значения mark2 и соответствующего ей id.

б) С помощью операции * добиться, чтобы значения элементов mark2 были повторены в списке еще один раз и сохранить результат в переменной mark3.

Используя print вывести значения mark2 и соответствующего ей id, а затем значения mark3 и соответствующего новой переменной id.

Использование списков для представления данных

Рассматривается следующая задача.

Имеются результаты ЕГЭ абитуриента, поступающего на специальности информационного профиля: Математика — 78, Информатика — 75, Русский язык — 62.

Необходимо выполнить простейшую программную обработку данных — вывести результаты ЕГЭ по дисциплинам и напечатать сумму баллов.

Выбор структуры данных

Если данные задачи достаточно сложно устроены, то выбор неудачного их представления в программе может существенно усложнить обработку информации. Поэтому для решения задачи необходима стадия проектирования, в которую входит выбор того формата представления данных, который наиболее соответствует требованиям задачи.

При принятии проектных решений можно исходить из разных критериев: быстрота доступа к данным, объем необходимой памяти для хранения информации или удобство обработки.

Учитывая небольшой объем информации в задаче, основным критерием будет третий — удобство обработки.

Для рассматриваемой задачи можно использовать несколько форматов задания исходных данных.

- Представление всей информации одной строкой:

```
ege1 = "Математика – 78, Информатика – 75, Русский язык – 62".
```

- Представление информации списком строк, в котором каждая строка содержит данные об одной дисциплине:

```
ege2 = ["Математика – 78", "Информатика – 75", "Русский язык – 62"]
```

- Представление информации единым списком из чередующихся строк и чисел:

```
ege3 = [ "Математика", 78, "Информатика", 75, "Русский язык", 62 ]
```

- Представить информацию списком списков, где каждый вложенный список содержит данные об одной дисциплине:

```
ege4 = [ ["Математика",78],  ["Информатика", 75], ["Русский язык", 62]]
```

ЗАДАНИЕ 3 (*Обработка информации из списков*)

1. Наиболее простым для обработки является формат, представленный переменной ege4.

а) Создать новый файл программы.

б) В нем исходный список сохранить в переменной `ege4`.

Он состоит из трех элементов `["Математика",78]`, `["Информатика", 75]` и `["Русский язык",62]`, с индексами `0`, `1` и `2`. Операция индексации будет возвращать один из этих трех элементов.

в) Каждый из таких элементов, в свою очередь, сам является списком из двух элементов с индексами `0` и `1` (учебная дисциплина и баллы). Поэтому, для доступа к каждому элементу данных нужно использовать два индекса (каждый в своих квадратных скобках).

С помощью операции индексации составить выражение, которое будет выбирать из списка название дисциплины "`Информатика`".

г) Написать функцию `total(ege)`, которой передается список с результатами `ege4`. В функции из списка поочередно извлекаются и запоминаются в отдельных переменных баллы, набранные по каждой из трех дисциплин. Затем эти баллы суммируются и полученная сумма становится возвращаемым значением функции.

д) На глобальном уровне программы выполняется вызов функции `total(ege)`, Её возвращаемое значение запоминается во вспомогательной переменной.

е) С помощью инструкции форматного вывода и функции `print` на экран выводится текст и результат: "`Общая сумма баллов ...`".

2. В следующем варианте программы используется формат, представленный переменной `ege3`.

а) Новый список с исходными данными назовем `ege3`.

б) Сделать копию функции `total2` и откорректировать её тело так, чтобы она извлекала и суммировала баллы из нового формата списка.

На глобальном уровне программы предусмотреть вызов функции `total2(ege3)`. Возвращаемое значение запоминается во вспомогательной переменной и с помощью инструкции форматного вывода и `print` вновь на экран выводится "`Общая сумма баллов ...`".

г) Реализовать другой способ решения задачи. Написать функцию `convert_list(ege)`, которая преобразует список `ege3` в список `ege_new`, имеющий формат списка `ege4`.

Для этого нужно трижды из исходного списка с помощью срезов выделить подсписки, состоящие из названия дисциплины и баллов по ней.

Затем объединить их в список *типа* `ege4`, который возвращает функция. А далее для вычисления суммы баллов применить функцию `total` из задания 3.1.

Вопросы для самоконтроля

1. Чем различаются два основных вида составных типов данных в языках программирования?
2. Что понимается под списками в Питоне? Какие типы данных могут использоваться при построении списков?
3. Что называется длиной списка?
4. Как записываются списки?
5. Что такое пустой список и как он записывается? Могут ли пустые списки быть элементами других списков?
6. Можно ли получать срезы списков?
7. Что означает высказывание "списки в Питоне являются изменяемыми"?
8. Что получается при "сложении" списков?
9. Что получается при "умножении" списка на целое положительное число?
10. Как добавить элемент в конец списка?
11. Можно ли добавить новый элемент в начало списка? Если можно, то как это сделать?
12. Можно ли добавить новый элемент в середину списка? Если можно, то как это сделать?