Гарвардский курс CS50

Методичка для преподавателей

Длительность занятия 3 часа

Модуль 6. Занятие 2

Цель занятия:

- Методы работы со списками
- Словари в Python

Ход занятия:

1. **Проверка домашней работы**
   В качестве домашнего задания ученики получили практические задачи на тему строк. Проверка проходит в стандартном режиме.
2. **Методы работы со списками**
   Перед началом углубления темы списков преподавателю рекомендуется напомнить тот материал, который был пройден на прошлом занятии. Далее можно переходить к первым методам.
   При разборе кода со скриншота № 1 стоит обратить внимание учеников на то, что в Python можно создавать пустые списки. И добавлять элементы в такие списки нужно именно с помощью метода append.
   В результате выполнения этого кода, пользователь сможет и задать длину списка, и каждый элемент этого списка с помощью клавиатурного ввода.

```
1 a = []
2 for i in range(int(input())):
3     a.append(int(input()))
4 print(a)
```

Скриншот №1

Продолжается изучение новых функций с функции len(), которая умеет возвращать в Python и длину списка, и длину строки (массив символов же). Пример, который можно продемонстрировать ученикам доступен на скриншоте № 2.

```
a = []
for i in range(int(input())):
    a.append(int(input()))

for i in range(len(a)):
    print(a[i])
```

Скриншот №2

Обратите внимание учеников на удобство этого метода. Это позволяет нам совершать действия над списками абсолютно произвольной длины. Далее группа переходит к разбору других способов инициализации списков.
Метод split() – метод, применяющийся к строке и возвращающий список строк, которые получаются, если исходную строку разрезать на части по пробелам, или другим символам.
Для демонстрации можно использовать сразу два примера. Сначала рекомендуется разобрать пример со скриншота № 3, а затем показать тонкости такого метода инициализации на примере скриншоте № 4. Если сформулировать эту тонкость простым языком – разделив строку на элементы списка, мы получаем также строковые элементы. В случае, если мы хотим работать со списком чисел, то придётся их преобразовать, что показано на скриншоте № 4.

```
# на вход подаётся строка
# 1 2 3
s = input()  # s == '1 2 3'
a = s.split()  # a == ['1', '2', '3']
```

Скриншот №3

```
a = input().split()
for i in range(len(a)):
    a[i] = int(a[i])
```

Скриншот №4

Также нельзя пройти мимо метода join, который является обратным действием метода split (соединяет список строковых элементов воедино). Пример, который отлично продемонстрирует ученикам работу этого метода доступен на скриншоте № 5.

```
a = ['red', 'green', 'blue']
print(' '.join(a))
# вернёт red green blue
print(''.join(a))
# вернёт redgreenblue
print('***'.join(a))
# вернёт red***green***blue
```

Скриншот №5

Чтобы показать ученикам более реальный способ применения этих методов, рекомендуется продемонстрировать реальный способ, к примеру, составления URL строки. Код, который это демонстрирует возможно найти на скриншоте № 6.

```
youtube = 'https://www.youtube.com/results?search_query='
search_query = input().split(' ')
result_query = youtube + '+'.join(search_query)
print(result_query)
```

Скриншот №6

Далее стоит упомянуть о методах, которые заслуживают отдельного внимания, но которые будут и так понятны ученикам по своему применению. Во время практической тренировки всё равно будет использоваться большая часть от них.
Пример таких методов доступен на скриншоте № 7.

x in A	Проверить, содержится ли элемент в списке. Возвращает True или False
x not in A	То же самое, что not(x in A)
min(A)	Наименьший элемент списка
max(A)	Наибольший элемент списка
A.index(x)	Индекс первого вхождения элемента x в список, при его отсутствии генерирует исключение ValueError
A.count(x)	Количество вхождений элемента x в список

Скриншот №7

После такого количества новой информации о, казалось бы, привычной структуры данных рекомендуется потратить значительное время занятия на решения задач. Решать нужно по ситуации, но рекомендуется решить минимум три задачи из списка ниже.

Список задач:

1) Выведите все элементы списка с четными индексами (то есть A[0], A[2], A[4], ...).

2) Дан список чисел. Выведите все элементы списка, которые больше предыдущего элемента.
3) Дан список чисел. Выведите значение наибольшего элемента в списке, а затем индекс этого элемента в списке. Если наибольших элементов несколько, выведите индекс первого из них.
4) В списке все элементы различны. Поменяйте местами минимальный и максимальный элемент этого списка.
5) Дан список чисел. Посчитайте, сколько в нем пар элементов, равных друг другу. Считается, что любые два элемента, равные друг другу образуют одну пару, которую необходимо посчитать.

3. Словари в Python

Продолжая разговор о структурах данных в Python стоит уделить отдельное внимание тем структурам, которые ученики не проходили в блоке С. Пока воспоминания от списков и методов работы списков свежи, рекомендуется подробно разобрать, что такое словарь в Python и показать случаи, в которых они могут быть удобны.
При объяснении этой структуры данных, преподавателю рекомендуется проводить аналогию со списком, при этом акцентируя, что вместо индексов в случае со словарями мы используем ключи значения. В качестве примера, демонстрирующими работу словарей рекомендуется подробно разобрать код со скриншота № 8.

```
1  # Создадим пустой словать Capitals
2  Capitals = dict()
3
4  # Заполним его несколькими значениями
5  Capitals['Russia'] = 'Moscow'
6  Capitals['Ukraine'] = 'Kiev'
7  Capitals['USA'] = 'Washington'
8
9  Countries = ['Russia', 'France', 'USA', 'Russia']
10
11 for country in Countries:
12     # Для каждой страны из списка проверим, есть ли она в словаре Capitals
13     if country in Capitals:
14         print('Столица страны ' + country + ': ' + Capitals[country])
15     else:
16         print('В базе нет страны с названием ' + country)
```

Скриншот №8

А в качестве примера работы со словарём можно использовать скриншот № 9. На этом примере стоит обратить отдельное внимание учеников на то, каким образом проинициализирован словарь, а также как удалять отдельные элементы словаря.

```python
A = {'ab' : 'ba', 'aa' : 'aa', 'bb' : 'bb', 'ba' : 'ab'}

key = 'ac'
if key in A:
    del A[key]

try:
    del A[key]
except KeyError:
    print('There is no element with key "' + key + '" in dict')
print(A)
```

Скриншот №9

Оставшееся время от занятия рекомендуется посвятить ответам на вопросы, а также, если преподаватель считает, что группе требуется дополнительный разбор пройденных структур хранения данных, решить несколько задач из рекомендуемого домашнего задания.

Рекомендуемое домашнее задание:

В качестве практического задания рекомендуется дать задачи на отработку пройденных структур данных.

Список задач:

1) В единственной строке записан текст. Для каждого слова из данного текста подсчитайте, сколько раз оно встречалось в этом тексте ранее. Словом считается последовательность непробельных символов идущих подряд, слова разделены одним или большим числом пробелов или символами конца строки.
2) Выведите все четные элементы списка. При этом используйте цикл for, перебирающий элементы списка, а не их индексы!
3) Дан список, упорядоченный по неубыванию элементов в нем. Определите, сколько в нем различных элементов.
4) Переставьте соседние элементы списка (A[0] с A[1], A[2] с A[3] и т. д.). Если элементов нечетное число, то последний элемент остается на своем месте.

Листинги всех решенных во время занятия задач доступны в соответствующем репозитории GitHub.