МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Управляющие конструкции языка Python

Студент гр. 8381 _____ Почаев Н.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

Цель работы

Изучить основные управляющие конструкции языка Python, используемые типы даннных, циклы и операторы. Научиться применять такие типы данных, как списки и строки. Изучить работу с функциями в языке.

Задание

На вход программе подается список длины 10, элементы которого разделены символом перевода строки, и число, которое означает номер операции. Реализуйте функцию-меню, которая должна в зависимости от номера операции, выводить следующее:
0: Только числа исходного списка.
1: Сумму всех четных чисел исходного списка.
2: Строку, полученную путем конкатенации всех элементов списка, длина которых меньше или равна трем.
3: Произведение всех нечетных чисел исходного массива.
4: Каждый пятый символ каждой строки (нумерация элементов строки начинается с нуля). При этом полученные строки становятся элементами нового списка. Строкой считается элемент списка, который нельзя привести к целому числу.
5: Индекс каждого элемента и сам элемент.
Любой другой символ: Исходный список.

Выполнение работы

Написание кода производилось на базе системы Linux Ubuntu 18.04 через интегрированную среду разработки PyCharm. Исходный код можно посмотреть в приложении А.

1. В начале в основном теле программы происходит считывание элементов списка, а затем передаётся номер функции, необходимой для выполнения.

2. После ввода происходит проверка функции на корректность введённых значений и перенаправление на выполнение необходимой функции.

3. Разветвление на необходимые функции, в зависимости от переданного значения, происходит при помощи оператора if-elif-else.

4. Каждая функция выполняется в зависимости от требований задания. При этом в работе использует вспомогательная функци is_number, применяемая для проверки содержания в строке только чисел.

- Функция is_number(x) пытается преобразовать строку к типу float и в случае ошибки передаёт логическое False, для данной операции использует оператор except.
- Функция only_numb(ar0) использует функцию is_number(x) для проверки элемента списка на число и выводит его в случае передачи функцией логического True.
- Функция even_sum производит дополнительную проверку на чётность числа и находит сумму чисел.
- Функция concat(ar2) производит проверку элементов списка (строки) на длину (<=3) и «склеивает» в единую строку, возвращаемую функцией.
- Функция odd_numb(ar3) аналогично проверяет передаваемую строку из списка на число и считает произведение всех нечётных.
- Функция fvie_str(ar4) проверяет передаваемую строку из списка на то, что оно не является числом, и её длина больше 4. После этого в цикле for при помоощи функции range() «склеивает» в единую строку, возвращаемую функцией, разделяя элемнта разных строк пробельным символом.
- Функция arr_ind(ar5) выводит элементы списка вместе с их индексами с помощью цикла со счётчиком for и переменной индекса (ind).

5. Если программе было передано неверное значение функции, то выводится исходный введённый список.

Выводы

В ходе лабораторной работы были изучены основы программирования на языке Python и написана программа, использующая описанные выше типы данных, циклы и операторы. В результате были приобретены знания по правильному объявлению константных переменных, передаче аргументов функции, работе со списками и строками. Также были изучены специальные функции, например, range() и lits.append().

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: main.py

```
SIZE = 10

# проверка на число

def is_number(x):
    try:  # обработка исключения
        float(x)
        return True
    except ValueError:
        return False

def only_numb(ar0):
    s_on_numb = []
    for m in ar0:
        if is_number(m):
            s_on_numb.append(m)
    return s_on_numb

def even_sum(ar1):
    r_sum = 0
    for j in ar1:
        if is_number(j) and int(j) % 2 == 0:
            r_sum += int(j)
    return r_sum

def concat(ar2):
    s_con = ''
    for k in ar2:
        if len(k) <= 3:
            s_con += k
    return s_con

def odd_numb(ar3):
    odd_pr = 1
    for l in ar3:
        if is_number(l) and int(l) % 2 != 0:
            odd_pr *= int(l)
    return odd_pr

def five_str(ar4):
    fiv = ''
    for t in ar4:
        if len(t) > 4 and not is_number(t):
            for h in range(5, len(t), 5):
                fiv += t[h]
            fiv += ' '
    return fiv
```

```
def arr_ind(ar5):
    ind = 0
    for h in ar5:
        print(ind, h)
        ind += 1
    return

def just_list(ar6):
    for v in ar6:
        print(v, end=' ')

initial = []
for i in range(SIZE):
    new_el = input()
    initial.append(new_el)

command = input()
if is_number(command) and int(command) in range(6):
    command = int(command)
    if command == 0:
        temp = only_numb(initial)
        for z in temp:
            print(z, end=' ')
    elif command == 1:
        print(even_sum(initial))
    elif command == 2:
        print(concat(initial))
    elif command == 3:
        print(odd_numb(initial))
    elif command == 4:
        print(five_str(initial))
    elif command == 5:
        arr_ind(initial)
else:
    just_list(initial)
```