ЛАБОРАТОРНАЯ РАБОТА 1. Типы данных. Списки, словари, кортежи, множества.

Цели и задачи

Цель лабораторной работы: изучить основы языка программирования Питон и научиться работать с целочисленной арифметикой.

Теоретическое обоснование

Типы данных и функции вывода

Программы на языке Питон представляют собой обычные текстовые файлы, в которых записана последовательность команд. Код легко читается и интуитивно понятен.

Например, выводящая Hello, world! программа записывается всего в одну строку:

```
print('Hello, world!')
```

В этой программе вызывается функция печати print, которой в качестве параметра передается строка, содержащая в себе фразу Hello, world!. Если мы хотим задать какую-то строку, то должны обрамлять её одинарными (') или двойными(") кавычками, иначе она будет интерпретироваться как код на языке Питон.

Кроме строк рассмотрим целочисленный тип данных. Например, можно посчитать результат вычисления арифметического выражения 2 + 3 и вывести его с помощью такой однострочной программы на языке Питон:

```
print(2 + 3)
```

Такая программа выведет результат вычисления выражения, который будет равен 5. Если бы числа 2 и 3 были заключены в кавычки, то они интерпретировались бы как строки, а операция + проводила бы конкатенацию (склеивание) строк. Например, такой код:

```
print('2' + '3')
```

выведет 23 - строку, состоящую из склеенных символов '2' и '3'.

Функция print может принимать и несколько параметров, тогда они будут выводиться через пробел, причем параметры могут иметь различные типы. Если мы хотим получить вывод вида 2 + 3 = 5, то можем воспользоваться следующей программой

print('2 + 3 =', 2 + 3)

Обратите внимание, что в строке '2 + 3 =' нет пробела после знака =. Пробел появляется автоматически между параметрами функции print. Что же делать, если хочется вывести строку вида 2+3=5 (без пробелов)? Для этого понадобится именованный параметр sep (separator, разделитель) для функции print. Та строка, которая передается в качестве параметра sep будет подставляться вместо пробела в качестве разделителя. В этой задаче мы будем использовать пустую строку в качестве разделителя. Пустая строка задается двумя подряд идущими кавычками.

print('2+3=', 2 + 3, sep='')

В качестве параметра sep можно использовать любую строку, в том числе состоящую из нескольких символов. Если нам нужно сделать несколько разных разделителей для разных частей строк, то не остается другого выбора, кроме как использовать несколько подряд идущих функций print. Например, если мы хотим вывести строку вида 1 + 2 + 3 + 4 = 10, то можем попробовать воспользоваться следующим кодом:

print(1, 2, 3, 4, sep=' + ')

print(' = ', 1 + 2 + 3 + 4, sep='')

Однако, вывод такого кода нас огорчит. Он будет выглядеть как:

1 + 2 + 3 + 4

= 10

Это связано с тем, что после каждой функции print по умолчанию осуществляется перевод строки. Для изменения того, что будет печататься после вывода всего, что есть в функции print можно использовать

именованный параметр end. Например, в нашем случае после первого print мы не хотели бы печатать ничего. Правильный код выглядит следующим образом:

print(1, 2, 3, 4, sep=' + ', end='')

print(' = ', 1 + 2 + 3 + 4, sep='')

В качестве end также можно использовать абсолютно любую строку.

Списки

Списки в Python – упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы элементов могут отличаться).

Чтобы использовать списки, их нужно создать. Создать список можно двумя способами(их вообще больше, но для простоты рассмотрим два самых часто используемых).

Способ 1. Использовать квадратные скобки.

```
# Пустой список
>>> empty_list = []
# Список, содержащий буквы
>>> letters_list = ['a', 'b', 'c']
# Список с элементами разных типов
     >>> rnd_list = ['a', 'b', True, [1, 2, 3]]
```

Способ 2: Использовать метод list:

```
# Создаем список на основе строки 'abc'
>>> list('abc')
     ['a', 'b', 'c']
```

Т. о. список может содержать любое количество любых объектов (в том числе и вложенные списки), или не содержать ничего.

Кортежи

Кортеж(tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список. Кортеж, по сути - неизменяемый список.

Может возникнуть вопрос, а зачем нужны кортежи, если есть списки? Причин несколько:

1. Защита от дурака. То есть кортеж защищен от изменений, как намеренных(что плохо), так и случайных (что хорошо).

2. Кортежи в памяти занимают меньший объем по сравнению со списками.

3. Возможность использовать кортежи в качестве ключей словаря(словари рассмотрим далее). Список в качестве ключа использовать не получится.

Как работать с кортежами? Примерно так же, как и со списками. Создать кортеж можно 2 способами:

Способ 1. Использовать метод tuple():

```
# Создаем кортеж из набора цифр
>>> tpl = tuple((1, 2, 3, 4))
>>> print(tpl)
(1, 2, 3, 4)
# Создаем кортеж из строки
>>> tpl = tuple('Строка')
>>> print(tpl)
     ('С', 'т', 'р', 'о', 'к', 'а')
```

Способ 2. Воспользоваться круглыми скобками ():

```
>>> tpl = (1, 2, 3, 4)
>>> print(tpl)
     (1, 2, 3, 4)
```

Как работать с кортежами?

1. Можно применять операции над списками, не изменяющие список (сложение, умножение на число, методы index() и count() и некоторые другие операции).

2. Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса. Но, как уже было сказано – изменять элементы кортежа нельзя!

3. Удалить отдельные элементы из кортежа невозможно. Но можно удалить кортеж целиком.

4. На базе кортежа можно создать список, верно и обратное утверждение.

Словари

Словари(dict) в Python – неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами.

Создать словарь можно двумя способами(их вообще 4, но для простоты рассмотрим два самых часто используемых).

Способ 1. Использовать фигурные скобки {}.

```
# Пустой словарь
>>> empty_dict = {}
# Словарь с двумя парами ключ-значение
     >>> value_dict = {'key1': 'value1', 'key2': 4}
```

Способ 2: Использовать метод dict():

```
# Обратите внимание, что
# 1) для ключей словаря не указываются скобки
# 2) вместо двоеточия используется знак равенства
>>> dct = dict(key1='value1', key2=4)
     {'key1': 'value1', 'key2': 4}
```

Множества

Множество(set) – это «контейнер», содержащий не повторяющиеся элементы в случайном порядке.

Создать множество можно 2 способами:

Способ 1. Использовать метод set():

```
# Создаем множество из набора цифр
>>> st = set({1, 2, 3, 4})
>>> print(st)
{1, 2, 3, 4}
# Создаем множество из строки
>>> st = set('Строка')
>>> print(st)
     {'к', 'С', 'т', 'а', 'р', 'о'}
```

Способ 2. Воспользоваться фигурными скобками {}:

```
>>> st = {1,2,3,4}
>>> print(st)
{1, 2, 3, 4}
# А вот пустое множество так создать нельзя
# Потому что при такой записи получим словарь
>>> st = {}
>>> type(st)
     <class 'dict'>
```

Переменные

В некоторых задачах вычисления удобно проводить используя вспомогательные переменные. Например, в школьных формулах по физике было удобно вычислять не гигантское выражение целиком, а запоминая результаты вычисления во вспомогательные переменные. Для примера решим задачу вычисления пройденного расстояния по известному времени и скорости:

```
speed = 108
time = 12
dist = speed * time
print(dist)
```

В этой программе мы создаем три переменные: speed для скорости, time для времени и dist для вычисленного расстояния. При использовании переменых в арифметическом выражении просто используется значение, которое лежит в переменной.

Для присваивания значения переменной используется знак =. Имя переменной должно быть записано слева от знака присваивания, а арифметическое выражение (в котором могут быть использованы числа и другие уже заданные переменные) - справа. Имя переменной должно начинаться с маленькой латинской буквы, должно быть осмысленным (английские слова или общеупотребимые сокращения) и не должно превышать по длине 10-15 символов. Если логичное имя переменной состоит из нескольких слов, то нужно записывать его с помощью camelTyping (каждое новое слово кроме первого должно быть записано с большой буквы).

Арифметические выражения

Мы уже использовали арифметические выражения в наших программах, в частности операции + и *. Также существует ряд других арифметических операций, которые приведены в таблице:

Знак	Операция	Операнд1	Операнд2	Результат
+	Сложение	11	6	17
-	Вычитание	11	6	5
*	Умножение	11	6	66
//	Целочисленное деление	11	6	1
%	Остаток от деления	11	6	5
**	Возведение в степень	2	3	8

Все операции инфиксные (записываются между операндами), т.е., например, для возведения 2 в степень 3 нужно писать 2**3.

Особо остановимся на операциях вычисления целой части и остатка от деления от числа.

Пусть заданы два числа A и B, причем B > 0. Обозначим за C целую часть от деления A на B, C = A // B, а за D - остаток от деления A на B, D = A % B.

Тогда должны выполняться следующие утверждения:

$A = B \times C + D$

$0 \leq D < B$

Эти утверждения необходимы для понимания процесса взятия остатка от деления отрицательного числа на положительное. Нетрудно убедиться, что если -5 разделить на 2, то целая часть должна быть равна -3, а остаток равен 1. В некоторых других языках программирования остатки в такой ситуации могут быть отрицательными, что неправильно по математическим определениям.

В случае, если $B < 0$ выполняются следующие утверждения:

$A = B \times C + D$

$B < D \leq 0$

Например, при делении 11 на -5 мы получим целую часть равную -3, а остаток будет равен -4.

Если же разделить -11 на -5, то целая часть будет равна 2, а остаток будет равен -1.

Обратите внимание, что целые числа в Питоне не имеют ограничений на длину (кроме объема доступной памяти).

Операции над строками

Строки также можно сохранять в переменные и использовать в некотором ограниченном количестве выражений. В частности, можно склеивать две строки с помощью операции +:

```
goodByePhrase = 'Hasta la vista'
person = 'baby'
print(goodByePhrase + ', ' + person + '!')
```

Складывать число со строкой (и наоборот) нельзя. Но можно воспользоваться функцией str, которая по числу генерирует строку. Str - это сокращение от слова string, которое можно перевести на русский как "строка, которая представляет собой последовательность символов". Например, задачу про вывод 2 + 3 = 5 можно решить и таким способом:

```
answer = '2 + 3 = ' + str(2 + 3)
print(answer)
```

Чтение данных

Можно умножить строку на целое неотрицательное число, в результате получится исходная строка, повторенная заданное число раз:

```
word = 'Bye'
phrase = word * 3 + '!'
print(phrase)
```

Программы, которые умеют только писать, но не умеют читать, редко представляют интерес для пользователей. Узнавать что-то из внешнего мира

наши программы будут с помощью функции input(). Эта функция считывает строку из консоли, чтобы закончить ввод строки нужно нажать Enter. Под строкой в данном случае понимается английское слово line, что означает "строка, оканчивающаяся переводом строки". Например, если в такую программу:

```
name = input()
print('I love', name)
```

ввести слово Python, то она напечатает I love Python.

Во многих задачах нам требуется работать со введенными числами, а читать мы умеем только строки. Чтобы преобразовать строку, состоящую из цифр (и, возможно, знака "-" перед ними) в целое число можно воспользоваться функцией int (сокращение от английского integer, "целое число"). Например, решение задачи о сложении двух чисел будет выглядеть так:

```
a = int(input())
b = int(input())
print(a + b)
```

Функция int может быть применена не только к результату, возвращаемому функцией input, но и к произвольной строке.

В строках могут быть не только буквы, цифры и прочие знаки препинания, но и, например, символы табуляции и перевода строки. Чтобы использовать эти символы в константной строке в коде программы необходимо записывать их как \t и \n соответственно. Использование бэкслеша перед символом называется экранирование. Также существуют и другие символы, которые требуют бэкслеша перед собой. Например, это кавычки \' и \" (использование бэкслеша просто необходимо, если в строке используются

оба типа кавычек), а также, собственно, символ бэкслеша, который надо записывать как \\.

В случае считывания с помощью input символы в консоли экранировать не нужно.

Условная инструкция

В линейной структуре программы все инструкции выполняются последовательно одна за одной.

Допустим, мы хотим по данному числу x определить его абсолютную величину (модуль). Программа должна напечатать значение переменной x, если x>0, или же величину −x в противном случае. Линейная структура программы нарушается: в зависимости от справедливости условия x>0 должна быть выведена одна или другая величина. Соответствующий фрагмент программы на Python имеет вид:

```
x = int(input()) i
f x > 0:
        print(x)
else:
        print(-x)
```

В этой программе используется условная инструкция if (если). После слова if указывается проверяемое условие x > 0, завершающееся двоеточием. После этого идет блок (последовательность) инструкций, который будет выполнен, если условие истинно. В нашем примере это вывод на экран величины x. Затем идет слово else (иначе), также завершающееся двоеточием, и блок инструкций, который будет выполнен, если проверяемое условие неверно, то есть будет выведено значение -x.

Примеры решения задач

Рассмотрим несколько задач, решаемых с помощью арифметических операций, которые показывают некоторые идеи.

Задача 1. Пусть есть два товара, первый из них стоит A рублей B копеек, а второй - C рублей D копеек. Сколько рублей и копеек стоят эти товары вместе.

В задачах, где есть несколько размерностей величин (например, рубли и копейки, километры и метры, часы и минуты) следует переводить все в наименьшую единицу измерения, осуществлять необходимые действия, а затем переводить обратно к нужным единицам.

В нашей задаче наименьшей единицей являются копейки, поэтому все цены следует перевести в них, затем сложить их, а затем перевести результат обратно в рубли и копейки. Код решения будет выглядеть так:

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
cost1 = a * 100 + b
cost2 = c * 100 + d
totalCost = cost1 + cost2
print(totalCost // 100, totalCost % 100)
```

Для определения количества рублей нужно разделить цену в копейках на 100 нацело, а для определения оставшегося числа копеек - посчитать остаток от деления на 100.

Задача 2. Вася играет в Super Mario Bros. очень хорошо и получил N дополнительных жизней. Известно, что переменная, в которой хранится количество жизней может принимать значения от 0 до 255. В случае, если было 255 жизней и игрок получил дополнительную жизнь, счетчик обнуляется. Сколько жизней на счетчике?

В этой задаче достаточно посчитать остаток от деления введенного числа на 256. Такие действия часто требуются, например, при работе со временем (при переходе через сутки счетчик времени обнуляется). Решение задачи выглядит так:

```
n = int(input())
print(n % 256)
```

Задача 3. Вводится число N, необходимо отрезать от него K последних цифр. Например, при N = 123456 и K = 3 ответ должен быть 123.

Для решения этой задачи нужно понять, что происходит при целочисленном делении на 10 (основание системы счисления). Если мы разделим число на 10, то будет отброшена последняя цифра, независимо от того, какой она была. Если разделим число на 100 - будет отброшено две последние цифры. Исходя из этого получается решение задачи: необходимо просто разделить число N на 10 в степени K:

```
n = int(input())
k = int(input())
print(n // 10**k)
```

Как переменные устроены внутри

В языке Питон все переменные являются ссылками на объекты. Каждый объект имеет тип (нам известны int и str) и содержимое, в нашем случае конкретное число или последовательность символов.

Переменные (ссылки) в языке Питон удобно представлять себе как ярлычки на веревочке, которые привязаны к какому-то объекту. Вообще говоря, к одному объекту может быть привязано сколь угодно много ярлыков. Различные переменные с одинаковым значением фактически являются ярлычками, привязанными к одному и тому же объекту.

Типы int и str в Питоне являются неизменяемыми. Любое присваивание в Питоне не может изменить неизменяемый тип, а может только изменить место, на которое указывает ссылка (и, при необходимости, сконструировать новый объект).

Например, команда x = 2, приведет сначала к созданию объекта типа "целое число" со значением 2 в памяти, а затем к созданию переменной x, которая будет являться ссылкой на этот объект.

Если после этого написать y = 2, то новый объект со значением 2 создаваться не будет, а создастся только новая ссылка с именем y, показывающая на тот же самый объект, что и ссылка x.

Если теперь написать строку x = 3, то с объектом со значением 2 ничего не случится, ведь он не неизменяемый. Создастся новый объект со значением 3, ссылка x отвяжется от объекта со значением 2 и привяжется к новому объекту 3. При этом к объекту 2 останется привязана ссылка y.

Если изменить и значение переменной y, то у объекта 2 не останется ссылок на него. Поэтому он может быть безболезненно уничтожен при сборке мусора, ведь получить к нему доступ уже невозможно - на него не ссылается ни одна переменная.

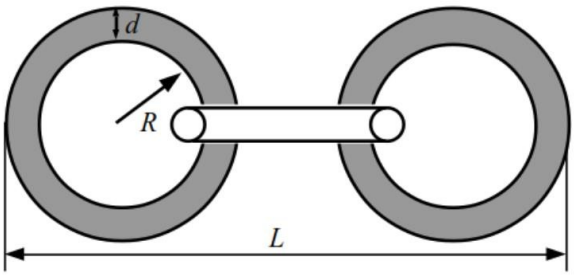
Константные значения в программе (например, явно заданные числа в исходном коде программы) также являются ссылками на объекты, содержимое которых совпадает со значением этих констант. Однако эти ссылки не могут быть изменены и не могут участвовать в присваивании с левой стороны от знака =.

Методика и порядок выполнения работы

Решить задачи. Необходимо решить 5 задач из таблицы. Номера задач взять у преподавателя.

№	Условие задачи	Пример	
		Ввод	Вывод
1	1 **Делёж яблок** n школьников делят между собой k яблок поровну, неделящийся остаток остаётся в корзинке. Сколько яблок	3 14	4 2

	достанется каждому школьнику? Сколько яблок останется в корзинке? Программа получает на вход в первой строке натуральное число n, а во второй — целое неотрицательное число k, и должна вывести два целых числа: количество яблок у каждого школьника и количество яблок, оставшихся в корзинке.		
2	**Настольный теннис** Феофилакт хочет купить ракетки и шарики для игры в настольный теннис. Один комплект ракеток стоит A рублей, один шарик стоит B рублей. У Феофилакта есть C рублей, C≥A, он покупает один комплект ракеток и шарики на оставшиеся деньги. Сколько шариков купит Феофилакт? Программа получает на вход три целых числа: A, B, C. Программа должна вывести ответ на задачу.	20 10 55	3
3	**Предпоследняя цифра** Дано натуральное число. Найдите число десятков в его десятичной записи (то есть предпоследнюю цифру его десятичной записи). Если заданное число является однозначным, то необходимо вывести 0.	179	7
4	**Сумма цифр** Дано четырёхзначное число. Найдите сумму его цифр.	2020	4
5	**Электронные часы** С начала некоторых суток прошло n минут. Определите, сколько часов и минут будут показывать электронные часы в этот момент. На вход программе подается целое неотрицательное число n. Программа должна вывести два числа: количество часов (от 0 до 23) и количество минут (от 0 до 59).	150	2 30

6	**Цепь** Из проволоки толщиной d миллиметров сделали кольца. Внутренний радиус каждого кольца составляет R миллиметров. Всего сделали n колец и их соединили в цепь. Определите длину получившейся цепи L. На рисунке изображен пример для n=3. 	2 10 3	64
7	**Автопробег** За день машина проезжает n километров. Сколько дней нужно, чтобы проехать маршрут длиной m километров? Программа получает на вход в первой строке натуральное число n и во второй строке целое неотрицательное число m.	700 750 700 2100	2 3
8	**Парты** В некоторой школе решили набрать три новых математических класса и оборудовать кабинеты для них новыми партами. За каждой партой могут сидеть двое учащихся. Известно количество учащихся в каждом из трёх классов. Определите, какое наименьшее число парт, которое нужно приобрести для них. Обратите внимание, что школьники из разных классов не могут сидеть за одной партой. Программа получает на вход три неотрицательных целых числа: количество учащихся в каждом из трёх классов, каждое в отдельной строке. Программа должна вывести ответ на задачу.	17 22 23	32
9	**Страницы книги** На каждой странице книги напечатано ровно k строк: на первой странице находятся строки с 1 по k, на второй —		2 50

	с k+1 по 2k и т.д. Определите, на какой странице находится строка номер nn и какой по счёту будет эта строка на странице. Даны натуральные числа k и n, каждое в отдельной строке. Программа должна считать их и вывести два числа: номер страницы и номер строки на странице.	50 100	
10	**Шахматная доска** Шахматная доска состоит из n×m клеток, покрашенных в чёрный и белый цвет в шахматном порядке. При этом клетка в левом нижнем углу доски покрашена в чёрный цвет. Определите, сколько всего на доске чёрных клеток. Программа получает на вход натуральные числа n и m. Программа должна вывести ответ на задачу.	3 4	6
11	**Следующее чётное** Дано целое число n. Выведите следующее за ним чётное число.	7 8	8 10
12	**Симметричное число** Дано целое неотрицательное число, меньшее 10000. Если число имеет меньше 4 знаков в десятичной записи, то нужно считать, что его десятичная запись дополняется слева незначащими нулями (например, число 120 дополняется до 0120). Определите, является ли его десятичная запись симметричной. Если число симметричное, то выведите 1, иначе выведите любое другое целое число.	2002 2008 440	1 37 1
13	**Максимум** Напишите программу, которая считывает два натуральных числа a и b и выводит наибольшее значение из них. При решении задачи можно пользоваться только целочисленными арифметическими операциями ++, −−, **, ////, %%, ==. **Нельзя пользоваться такими конструкциями,** как ветвления, циклы, функции.	8 5 5 8	 8 8

		5 5	 5

Контрольные вопросы

1. Какие арифметические операции есть в python?
2. Что такое переменная?
3. Как получить данные введенные пользователем?
4. Что такое множество?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-5].