**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Python. Программы с линейной структурой**

**Методические указания к лабораторной работе**



**Рязань 2016**

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Python. Программы с линейной структурой**

**Лабораторная работа №2**

**Методические указания к лабораторной работе**

**Рязань 2016**

УДК 004.432

Python. Программы с линейной структурой: методические указания к лабораторной работе. / Рязан. гос. радиотехн. универ.; Сост.: А.Н. Пылькин, Н.Н. Степанов, Н.А. Тярт. – Рязань, 2016 г.

Рассмотрены правила составления программ на языке Python, имеющих линейную структуру. Даны определения и синтаксис основных объектов программы: идентификаторы, комментарии, переменные, встроенные (стандартные) типы данных, простейшие операторы ввода и вывода данных. Значительное место отведено описанию арифметических выражений и приоритету выполнения операций, а также определен список встроенных функций языка Python.

В качестве практического задания предлагается составить простейшую программу на языке Python, реализующую вычисления по формулам.

Табл.: 4. Ил.: 2. Бибилиогр.: 4 назв.

Печатается по решению Научно-методического совета Рязанского государственного радиотехнического университета.

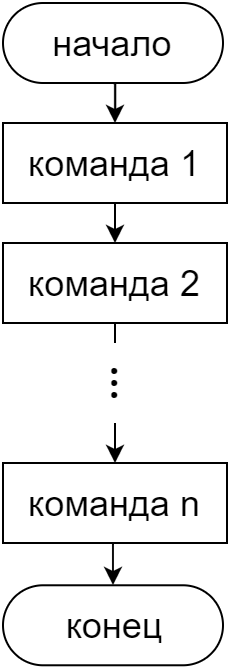
Рецензент: кафедра информатики, информационных технологий и защиты информации ФГБОУ ВО «Липецкий государственный педагогический университет им. П.П. Семенова-Тян-Шанского»  
(зав. каф., к.т.н., доц. Скуднев Д.М.).

# Цель работы

Получение базовых знаний об объектах программы на языке Python, а также получение практических навыков написания программ с линейной структурой.

# Линейный алгоритм

Линейный алгоритм включает последовательность команд (указаний, действий, операторов), выполняемых друг за другом в естественном порядке. На схеме линейный алгоритм представляется последовательностью блоков (см. рис. 1).



*Рис. 1.*

Программа, реализующая линейный алгоритм, представляет собой обычную последовательность команд. Команды выполняются последовательно, с первой и по последнюю (если в работе программы не возникает ошибок).

# Основные элементы языка Python

Идентификаторы в Python представляют собой имена, используемые для обозначения переменной, функции, класса, модуля или другого объекта. Идентификатор должен начинаться с латинской буквы (a…z) или со знака подчеркивания (\_), после которых может идти произвольное количество букв, знаков подчеркивания и чисел (от 0 до 9). Python поддерживает кириллицу (русские буквы) в идентификаторах, однако строго не рекомендуется использовать эту возможность. Не все языки (среды, программные средства) имеют поддержку кириллицы, что может вызвать большие трудности в дальнейшем.

В Python недопустимо использование знаков препинания или специальных символов, таких как @, $ или %, в качестве идентификаторов. Кроме того, Python чувствителен к регистру, то есть *cat* и *Cat* это два разных имени.

Как и в других языках программирования, в Python имеются зарезервированные (ключевые) слова. Полный список ключевых слов для Python приведен в табл. 1.

Таблица 1. Зарезервированные слова

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| False | None | True | and | as | assert |
| break | class | continue | def | del | elif |
| else | except | finally | for | from | global |
| if | import | in | is | lambda | nonlocal |
| not | or | pass | raise | return | try |
| while | with | yield |  |  |  |

Зарезервированные слова нельзя использовать в качестве имени переменной или любого другого идентификатора. Все ключевые слова Python состоят только из букв в нижнем регистре (кроме False, None и True) и имеют определенный смысл, значение которого поясняется в дальнейшем. В зависимости от версии Python, получить список ключевых слов можно в любой момент, введя в интерпретаторе команду:

help("keywords")

Основные правила написания программы.   
В отличие от других языков (например, Pascal и C/C++), в Python в конце строки не требуется ставить никакого завершающего символа. Конец строки (переход на новую строку) обозначает конец инструкции. Однако, символ «;» может использоваться для ввода нескольких инструкций на одной строке, например:

**import** random; x = random.randint(); **print** x

Но данный стиль написания кода резко ухудшает его читаемость, поэтому рекомендуется писать каждую инструкцию с новой строки.

Также в Python нет необходимости в операторных скобках. Объединение ряда инструкций в блок осуществляется отступами (табуляцией). Выглядит это следующим образом:

Основная инструкция:   
 Вложенный блок инструкций

Величина отступа может быть любой, главное – соблюдать одну величину отступа в пределах одного вложенного блока инструкций. Основная инструкция должна заканчиваться двоеточием.

При необходимости перенести часть команды на новую строку (в случае очень больших команд) в конце каждой предыдущей строки в качестве символа переноса необходимо ставить символ «\». Например:

x = x1 + \  
 x2 + \  
 x3

Выражения, находящиеся внутри скобок (круглых ( ), квадратных [ ] или фигурных { }), не нуждаются в символе переноса строки:

x = (x1 +

x2 +

x3)

Кавычки. В **Python** можно использовать одинарные   
( ' ), двойные ( " ) и тройные ( ''' или """ ) кавычки для обозначения строк и символов. При этом начинаться и заканчиваться строка должна одинаковыми кавычками. Строка, занимающая несколько строк кода, должна быть обрамлена тройными кавычками. Например:

name = ‘Some text’

surname = “Some text”  
help = ””” Some  
 text ”””

Комментарии. Однострочный комментарий в Python обозначается символом # в начале строки. Блок комментариев представляется в виде множества однострочных комментариев – в начале каждой строки ставится символ #.

*# Комментарий*

**print**(“Hello, Python”) *# Комментарий*

В Python присутствует возможность делать многострочные комментарии в виде строк в тройных кавычках, однако такой стиль комментирования крайне не рекомендуется.

Переменные. В Python отсутствует строгая типизация, т.е. не требуется заранее объявлять переменную с указанием ее типа. Python устроен по принципу динамической типизации – переменная объявляется автоматически во время ее ввода и присвоения ей значения. Тип переменной выбирается в зависимости от присваиваемого ей значения. Оператор присваивания в языке представлен символом =. Например:

x = 1 *# объявлена переменная x типа int  
 # с начальным значением 1*

age = ’18 лет’ *# объявлена переменная age  
 # типа string со значением ’18 лет’*  
**print**(x) *# вывод значений переменных на экран*  
**print**(age)

Возможно множественное присваивание значения нескольким переменным:

a = b = c = 1

При этом операция будет осуществлена в порядке слева направо: сначала переменная c примет значение 1, затем b примет значение c, и в конце a примет значение b. В итоге значения всех трех переменных будут равны 1, но следует учитывать очередность выполнения операторов.

# **Встроенные типы данных**

Встроенные (или стандартные) типы данных используются для хранения определенных значений и обладают некоторыми правилами обработки и преобразования данных. К стандартным относятся следующие типы:

1. Числа (numbers).
2. Строки (string).
3. Списки (list).
4. Кортежи (tuple).
5. Словари (dictionary).

Числовые типы данных предназначены для хранения числовых значений, и изменение значения приводит к созданию нового объекта (и удалению старого). Числовой объект создается в результате выполнения присваивания, например:

x1 = 1.5  
x2 = 4

Для удаления числового объекта используется ключевое слово del, например:

**del** x1, x2

В языке Python используются различные числовые типы данных:

* *int* (целые числа);
* *float* (числа с плавающей точкой);
* *complex* (комплексные числа).

*Примечание.* В Python 2 помимо трех вышеупомянутых типов был также четвертый тип *long* (длинные целые числа); при этом тип *int* имел ограничение по максимальному значению, а тип *long* был неограниченным. В Python 3 тип *long* упразднен, а его место занял тип *int*.

Пример создания объектов целого типа:

x = 4  
y = 5

z = x \* y

В результате выполнения инструкции z = x \* y переменной z (целого типа) будет присвоено целочисленное значение 20. Аналогично выполняются действия с данными типа *float*. Если арифметическая операция выполняется с операндами типов *int* и *float*, то осуществляется преобразование целого операнда в число с плавающей точкой.

Комплексное число записывается в форме суммы действительной и комплексной частей, при этом мнимая часть записывается с суффиксом j (или J): real + imag j, например: (9+3j), 1j, 2J, (-1+0j).

Действительная и мнимая части представляются числами с плавающей точкой. Чтобы обратиться к действительной и мнимой части комплексного числа z, используются составные имена z.real и z.imag.

Строки. Кроме чисел, Python может работать со строками (*string*), которые могут быть описаны различными путями. Строка состоит из символов таблицы ASCII, последовательность которых заключена в кавычки (стр. 5), например:

‘Python’

“Питон”  
‘Язык программирования “Питон” ’

‘x1 = ’

Длинные строки (как и любые инструкции в общем случае) могут быть разбиты на несколько экранных строк при использовании символа переноса ‘\’, например:

myString = “Это очень \

и очень \

длинная строка”

Преобразование типов данных. В некоторых случаях необходимо выполнять преобразование одного типа в другой тип. Для этого существуют встроенные функции приведения типов, некоторые из которых представлены в табл. 2.

Таблица 2. Функции преобразования

|  |  |  |
| --- | --- | --- |
| **Функция** | **Описание** | **Примеры использования** |
| int(x [,base]) | Преобразует х в целое число. | int(12.8) -> 12 |
| float(x) | Преобразует х в число с плавающей точкой. | float(10) -> 10.0 |
| complex(real  [,imag]) | Создает комплексное число. | complex(20) -> (20+0j) |
| str(x) | Преобразует х в строку. | str(10) -> '10' |

# Ввод и вывод данных

Для вывода полученных результатов на консоль в Python есть встроенная функция print(). Синтаксис функции и описание ее параметров:

**print**(value1, ..., [valueN], [sep=' '], [end='**\n**'], [file=sys.stdout], [flush=False])

1. value1, …, valueN – список выводимых значений любого типа.
2. sep – разделитель между значениями (value1, …, valueN). По умолчанию - одиночный пробел.
3. end – конец вывода. По умолчанию – символ переноса строки.
4. file – выходной поток. По умолчанию – консоль (экран монитора).
5. flush – флаг, определяющий режим вывода в файл при наличии цикла. Действует только в том случае, если выбран файл в аргументе file. При значении True вывод в файл происходит незамедлительно при вызове функции print(). При значении False (по умолчанию) вывод в файл будет произведен только после выхода из цикла.

В простейшем случае обязательным параметром является только список объектов для вывода:

**print**(value1, ..., [valueN])

Примером применения функции print() является следующий фрагмент программы:

**print**(3 \* 7, (17 - 2) \* 8)

*# Будет выведен в консоль результат*

*# двух выражений через пробел:*

*# 21 120*

x = 5

**print**(2 \* x)

*# 10*text = ‘Hello, World!’

**print**(text)

*# Hello, World!*

Ввод данных осуществляется с помощью функции input(). Ее синтаксис прост:

**input**([prompt])

Необязательный параметр promt – текст приглашения на ввод. Например:

name = input(“Введите ваше имя:”)

*# Результат – строка «Введите ваше имя:»*

*# с курсором ввода после двоеточия*

Функция input() возвращает введенное значение, которое необходимо присвоить какой-либо переменной. При этом следует помнить, что функция возвращает строковое значение (тип *string*). При необходимости ввода, например, числовых значений нужно использовать приведение типов:

x = input() *# ввели 10; x = ‘10’ – тип string*  
x = float(input()) *# ввели 10; x = 10.0 – тип float*

# Арифметические операции

Для целых чисел (тип int) в языке Python определены следующие операции:

+ - сложение;

– - вычитание;

\* - умножение;

// - целочисленное деление (аналог div в Pascal);

% - остаток от целочисленного деления (аналог mod в Pascal).

\*\* - возведение в степень.

В результате применения операции деления / получается вещественное число (типа *float*). Также операция возведения в степень \*\* возвращает вещественное число, если степень отрицательна. Пример операций:

**print**(17 / 3) *# выведет 5.66666666667*  
**print**(17 // 3) *# выведет 5*  
**print**(17 % 3) *# выведет 2*

В случае приведения типа float к типу int округление происходит в сторону 0, например:

int(1.7) == 1

int(-1.7) == -1.

# Приоритет выполнения операций

Порядок выполнения отдельных операций в одной команде определяется приоритетом. Сначала выполняются операции с более высоким приоритетом, например, в выражении «2+2\*3» будет выполнено сначала умножение (как и по правилам математики), затем сложение (результат = 8). При необходимости порядок выполнения операций можно изменить с помощью скобок, например,

В языке Python используются различные операции в командах. В табл. 3 приведен полный список этих операций в порядке убывания приоритета (от наивысшего до наинизшего). В настоящей лабораторной работе используются простейшие операции, другие (более сложные) будут использоваться в других работах – их действие (результат выполнения) определяется в дальнейшем.

Таблица 3. Операции в Python в порядке убывания приоритета выполнения

|  |  |
| --- | --- |
| **Операция** | **Описание действия** |
| \*\* | Возведение в степень. |
| ~ + – | Комплементарные операции. |
| \* / % // | Умножение, деление, деление по модулю, целочисленное деление. |
| + – | Сложение и вычитание. |
| >> << | Побитовый сдвиг вправо и влево. |
| & | Бинарное "И". |
| ^ | | Бинарное  "ИСКЛЮЧАЮЩЕЕ ИЛИ" и бинарное "ИЛИ". |
| < <= > >= | Операции сравнения (меньше, меньше или равно, больше, больше или равно). |
| <> == != | Операции сравнения на равенство. |
| = %= /= //= -= += \*= \*\*= | Присваивание. |
| is is not | Тождество и не тождество. |
| in not in | Принадлежит множеству и не принадлежит множеству. |
| not or and | Логическое отрицание, сложение и умножение. |

# Функции модуля math

Для выполнения сложных вычислительных операций крайне полезными могут оказаться встроенные функции языка, реализованные в библиотеке math. Для того, чтобы использовать эти функции, необходимо подключить данный модуль. Сделать это можно двумя способами.

Первый вариант – команда import math. При этом функции данного модуля будут вызываться в виде math.<имя\_функции>. Например:

**import** math

x = math.ceil(4.2)

y = math.log(4.8)

Второй вариант – команда from math import <имя\_функции>. При этом пропадет необходимость указывать имя модуля перед вызовом функции. Вместо имени функции может быть указан символ \* для выбора сразу всех функций из модуля:

**from** math **import** \*

x = ceil(4.2)  
y = log(4.8)

В табл. 4 представлен полный список функций, входящих в модуль math. Имена функций в таблице указаны без приставки math.<>

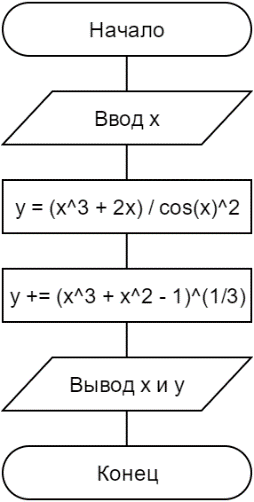
Таблица 4. Функции модуля math

|  |  |
| --- | --- |
| **Функция** | **Описание действия** |
| ceil(X) | Округление значения X до целого вверх. |
| copysign(X, Y) | Возвращает число, имеющее модуль такой же, как и у числа X, а знак – как у числа Y. |
| fabs(X) | Модуль X, т.е. |X| . |
| factorial(X) | Факториал числа X, т.е. X! . |
| floor(X) | Округление значения X до целого вниз. |
| fmod(X, Y) | Остаток от деления X на Y. |
|  |  |
|  |  |
| Продолжение таблицы 4 | |
| frexp(X) | Возвращает мантиссу и экспоненту числа. X -> A \* 2B. Применение: A,B = math.frexp(X). |
| ldexp(X, I) | Возвращает значение X \* 2I. Функция, обратная функции math.frexp(). |
| fsum(x1,x2,…,xN) | Сумма всех членов последовательности (x1,x2,…,xN). Эквивалент встроенной функции sum(), но math.fsum() более точна для чисел с плавающей точкой. |
| isfinite(X) | Возвращает True, если X является числом, иначе – False. |
| isinf(X) | Возвращает True, если X является бесконечностью, иначе – False. |
| isnan(X) | Возвращает True, если X является NaN *(Not a Number – не число)*, иначе – False. |
| modf(X) | Возвращает дробную и целую часть числа X. Оба числа имеют тот же знак, что и X. Например,  math.modf(20.1) ->(0.1, 20) |
| trunc(X) | Усекает значение X до целого. |
| exp(X) | eX |
| expm1(X) | eX – 1. При X → 0 точнее, чем math.exp(X)–1. |
| log(X, [base]) | Логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм. |
| log1p(X) | Натуральный логарифм (1 + X). При X → 0 точнее, чем math.log(1+X). |
| log10(X) | Логарифм X по основанию 10. |
| log2(X) | Логарифм X по основанию 2. |
| pow(X, Y) | XY |
| sqrt(X) | Квадратный корень из X. Альтернатива: X\*\*(1/2).  Но X\*\*(1/2) вернет комплексное число для отрицательного X, а sqrt(X) выдаст ошибку. |
| acos(X) | Арккосинус X. В радианах. |
| asin(X) | Арксинус X. В радианах. |
|  |  |
| Окончание таблицы 4 | |
| atan(X) | Арктангенс X. В радианах. |
| atan2(Y, X) | Арктангенс Y/X. В радианах. С учетом четверти, в которой находится точка (X, Y). |
| cos(X) | Косинус X (X указывается в радианах). |
| sin(X) | Синус X (X указывается в радианах). |
| tan(X) | Тангенс X (X указывается в радианах). |
| hypot(X, Y) | Вычисляет гипотенузу треугольника с катетами X и Y (math.sqrt(x \* x + y \* y)). |
| degrees(X) | Конвертирует радианы в градусы. |
| radians(X) | Конвертирует градусы в радианы. |
| cosh(X) | Вычисляет гиперболический косинус. |
| sinh(X) | Вычисляет гиперболический синус. |
| tanh(X) | Вычисляет гиперболический тангенс. |
| acosh(X) | Вычисляет обратный гиперболический косинус. |
| asinh(X) | Вычисляет обратный гиперболический синус. |
| atanh(X) | Вычисляет обратный гиперболический тангенс. |
| erf(X) | Функция ошибок. |
| erfc(X) | Дополнительная функция ошибок (1– math.erf(X)). |
| gamma(X) | Гамма-функция X. |
| lgamma(X) | Натуральный логарифм гамма-функции X. |
| pi | π = 3,1415926... |
| e | e = 2,718281... |

# Пример выполнения задания

*Задание.* Вычислить значение уравнения:

*Схема алгоритма программы (рис. 2):*



*Рис. 2.*

*Текст программы:*

**from** math **import** \*  
  
x = float(input(**"Введите x: "**))  
print(**"x ="**, x)  
y = (x\*\*3 + 2\*x) / (cos(x)\*\*2)

y += (x\*\*3 + x\*\*2 - 1)\*\*(1/3)  
print(**"y ="**, y)

*Результаты выполнения:*

Введите x: 2  
x = 2.0  
y = 71.51677053907231

# Контрольные вопросы

1. Что такое линейный алгоритм и как он представляется на схеме?
2. Каковы правила написания идентификаторов в Python?
3. Как в Python происходит разделение команд (на одной строке, на нескольких строках), объединение команд в блок, перенос части команды на новую строку?
4. Какие типы числовых данных есть в Python?
5. Как осуществляется приведение типов и в каких случаях оно необходимо?
6. Как осуществляется ввод и вывод данных?
7. Какие арифметические операции над числами вы знаете?
8. Как использовать функции из модуля math?

# Задание

Для задания в соответствии со своим вариантом составить алгоритм и написать программу, реализующую ввод исходных данных, вычисление значения уравнения и вывод результата. Оформить отчет, содержащий текст варианта задания, схему алгоритма, код программы и результаты выполнения.

Во всех вариантах задания значения входных параметров (x, a) задаются произвольно при вводе исходных данных.

*Варианты задания:*

# Литература

1. Python на примерах. Практический курс по программированию. / А.Н. Васильев. – СПб.: Наука и Техника, 2016. – 432 с.
2. <http://pythonicway.com>
3. <https://pythonworld.ru>
4. <http://www.python-course.eu/python3_course.php>

Содержание

[Цель работы 3](#_Toc466665741)

[Линейный алгоритм 3](#_Toc466665742)

[Основные элементы языка Python 3](#_Toc466665743)

[Идентификаторы в Python 3](#_Toc466665744)

[Основные правила написания программы. 4](#_Toc466665745)

[Кавычки. 5](#_Toc466665746)

[Комментарии. 5](#_Toc466665747)

[Переменные. 6](#_Toc466665748)

[Встроенные типы данных 6](#_Toc466665749)

[Числовые типы данных 7](#_Toc466665750)

[Комплексное число 7](#_Toc466665751)

[Строки. 8](#_Toc466665752)

[Преобразование типов данных. 8](#_Toc466665753)

[Ввод и вывод данных 8](#_Toc466665754)

[Арифметические операции 10](#_Toc466665755)

[Приоритет выполнения операций 10](#_Toc466665756)

[Функции модуля math 12](#_Toc466665757)

[Пример выполнения задания 15](#_Toc466665758)

[Контрольные вопросы 16](#_Toc466665759)

[Задание 16](#_Toc466665760)

[Литература 18](#_Toc466665761)