Введение

Эта книга является учебным пособием для начинающих. Благодаря этой книге вы познакомитесь с языком программирования Python, а именно его третьей версией. Мы пройдём путь от простейших линейных алгоритмов до сложнейших конструкций с бесконечными циклами и фрактальным вызовом функций. Вы можете обучаться с помощью этой книги дома или на дополнительных занятиях в школе, при наличии компьютера. Для программирования вам понадобиться прежде всего установить сам Python с официального сайта, код можно писать хоть в Блокноте, но удобнее всего это делать в специализированном редакторе, таком как например PyCharm. Все примеры нужно запускать самому для полного понимания. Все пример можно скачать и запустить:

Условные обозначения:

S – всегда переменная типа string

I.Основы работы яп Python. Переменные. Их типы. И операции над ними.

**Параграф 1. Основы.**

Все яп переводятся в бинарный код для обработки процессором, этим занимается компилятор. Но “компилятор” в Python называется интерпретатор, он переводит python код в байт код. Это позволяет упростить язык до нельзя. Но не думайте, что у Python нету минусов, он работает заметно медленнее других яп.

В Python существуют различные библиотеки, расширяющие возможности. Чтобы ими пользоваться нужно прописать команду import.

Пример 1

import math  
import math as mt  
from math import \* #импортирет все из модуля math   
from math import sqrt  
  
  
# однострочный комментарий  
"""  
многострочный комментарий  
"""  
math.sqrt(144)  
mt.sqrt(144)  
sqrt(144)  
sin(45)

Чтобы вывести информацию в консоль в Python используется команда print(). Если вы хотите вывести число, просто напишите его в скобках, если-же вы хотите вывести строку, то она должна быть заключена в любые кавычки, чтобы вывести переменную, нужно просто ввести её название, если нужно вывести несколько значений подряд, они записываются через запятую.

Пример 2

print("Hello world!", 10)

Текст можно выводить форматировано используя обратный слеш и некоторые буквы английского алфавита.

\\ - так можно вывести \

\’ или \” – выводит кавычки

\n – выводит следующий после текст с новой строки

\t – добавит табуляцию

Пример 3

print('Hello\nworld!')

Python значения, написанные через “,”, выводит через пробел, пробел можно заменить чем угодно, с помощью sep.

Пример 4

print('Hello,','Brad' , 'and Vlad', sep='=')

Также у функции print есть параметр end, по умолчанию он равен “\n”.

Пример 5

print('Hello,','Bard' , 'and Vad', end='novaya stroka')

**Параграф 2. Переменные.**

Переменные – это основа любого языка программирования. У переменных всегда есть названия. В Python название переменной не может начинаться с числа, также вы конечно можете назвать переменную кириллицей, а кто-то даже сможет мефодицей, но так не принято делать. Переменные хранят в себе значения, и как не странно их можно менять. Также во многих языках программирования присутствуют константы. Константы – это как переменные, но их нельзя менять. В Python нет констант, но заместо этого вы можете написать название переменной большими буквами, как в настоящем языке программирования. Оператором присвоения является знак “=”.

Переменные бывают нескольких видов:

* Целочисленные – int (от англ. integer)
* Числа с плавающей точкой – float
* Строковые – str (от англ. string)
* Смысловые – bool
* Список – list
* Словарь – dict

**Параграф 3. Операции над целочисленными переменными и переменными с плавающей точкой.**

Помимо основных операций присутствуют такие как целочисленное деление “//” и остаток от деления “%”.

Пример 6

#Основные мат. операции  
print('x+y=', x+y)  
print('x-y=', x-y)  
print('x\*y=', x\*y)  
print('x/y=', x/y)  
print('x%y=', x%y)#наход. остатка при /  
print('x//y', x//y)#наход целой части при /  
print('x\*\*y', x\*\*y)#возвести в степень

**Параграф 4. Операции над строками.**

Строки можно складывать и умножать.

Пример 7

a = "abc"  
b = 'xyz'  
c = a + b  
print("a =", a)  
print("b =", b)  
print("(a + b) =", a + b)  
print(c \* 3)

К строкам можно обращаться по индексу, индекс первого символа строки – 0, индекс последнего символа строки – -1. Также с помощью индексов можно извлекать срезы.

Пример 8

s = "Hello, beautiful world!"  
print(s[5], s[-3])  
print(s[3:5]) # срез  
print(s[:]) # полный срез  
print(s[3:1:5]) # срез с шагом

Если нужно узнать количество символов в строке, используется функция len().

Пример 9

Stroka = "1234"  
print(len(Stroka))

У каждого символа есть свой код в кодировке ASCII (American standard code for information interchange). Его можно узнать с помощью команды ord. Также из числа можно получить букву с помощью команды chr.

Пример 10

S = "Poulic"  
P = ord(S[0])  
print("P =", P)  
print(chr(P))

**Методы строк**

# Поиск подстроки в строке. Возвращает номер первого вхождения или -1

S.find(str, [start],[end])

# Поиск подстроки в строке. Возвращает номер последнего вхождения или -1

S.rfind(str, [start],[end])

# Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError

S.index(str, [start],[end])

# Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError

S.rindex(str, [start],[end])

# Замена шаблона

S.replace(шаблон, замена)

# Разбиение строки по разделителю

S.split(символ)

# Состоит ли строка из цифр

S.isdigit()

# Состоит ли строка из букв

S.isalpha()

# Состоит ли строка из цифр или букв

S.isalnum()

# Состоит ли строка из символов в нижнем регистре

S.islower()

# Состоит ли строка из символов в верхнем регистре

S.isupper()

# Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v'))

S.isspace()

# Начинаются ли слова в строке с заглавной буквы

S.istitle()

# Преобразование строки к верхнему регистру

S.upper()

# Преобразование строки к нижнему регистру

S.lower()

# Начинается ли строка S с шаблона str

S.startswith(str)

# Заканчивается ли строка S шаблоном str

S.endswith(str)

# Сборка строки из списка с разделителем S

S.join(список)

# Символ в его код ASCII

ord(символ)

# Код ASCII в символ

chr(число)

# Переводит первый символ строки в верхний регистр, а все остальные в нижний

S.capitalize()

# Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)

S.center(width, [fill])

# Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)

S.count(str, [start],[end])

# Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам

S.expandtabs([tabsize])

# Удаление пробельных символов в начале строки

S.lstrip([chars])

# Удаление пробельных символов в конце строки

S.rstrip([chars])

# Удаление пробельных символов в начале и в конце строки

S.strip([chars])

# Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки

S.partition(шаблон)

# Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку

S.rpartition(sep)

# Переводит символы нижнего регистра в верхний, а верхнего – в нижний

S.swapcase()

# Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний

S.title()

# Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями

S.zfill(width)

# Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar

S.ljust(width, fillchar=" ")

# Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar

S.rjust(width, fillchar=" ")

Для полного усвоения рекомендуется по пробовать на деле все эти методы и вам наверно кажется что большая их часть не пригодится, но это не так.

**Параграф 5. Списки и операции над ними.**

Список – последовательность данных разных типов. У списка тоже есть нумерация, такая же как у строки. Команда len также справедлива для списков. Можно делать срезы списков. Метод index принимает значение и возвращает индекс этого значения. Метод count принимает значение, а возвращает количество его повторений. Метод sort сортирует список. Метод append принимает значение добавляет его в конец списка. Метод remove удаляет полученное значение из списка. Метод pop удаляет значение по индексу. Метод extend склеивает два списка, тоже самое что и сложение двух списков. Метод insert принимает индекс и вставляет значение. Также существуют функции: min, max, sum. Можно проверять есть ли значение в списке с помощью in или not in.

Пример 11

l1 = [1, 2, 4, 3, 5, 4, 3, 6, 1, 2, 8, 7, 9]  
l2 = ['Ann', 'Paul', 'Rodion', 'Max', 'Matt', 'Billy', 'Thomas']  
print(l2.index('Paul'))  
print(l1.index(2, 4)) # ищем индекс двойки но начиная с четвертого индекс  
print(l1.count(3))  
l1.sort()  
print(l1)  
l2.sort(reverse=True) # в обратном порядке  
l2.append('Tom')  
print(l2)  
l1.remove(1)  
print(l1)  
name0 = l2.pop(0)  
print(l2)  
print(name0)  
l1.extend([1, 2])  
print(l1)  
l1.insert(0, l2) # мы получили вложенный список  
print(l1)  
print(l1[0][3]) # таким образом я обратился к вложенному списку и выведу то что находится в нем под индексом 3

**Параграф 6. Словари.**

Словари – это не упорядоченные списки. Словарь содержит пары ключ : значение. К элементам словаря нужно обращаться по ключу. Словари могут пригодиться при де кодировке или кодировке. В целом словари самый бес полезный с моей точки зрения тип, но упомянуть их стоит. Запись словарей выглядит так.

Пример 12

dct1 = {'P': 'Paul', 'A': 'Ann', 'R': 'Rodion'}

словарь можно создать и так.

Пример 13

l1 = ['P', 'A', 'R']  
l2 = ['Paul', 'Ann', 'Rodion']  
dct2 = dict.fromkeys(l1, l2) # пример случая , когда хотел одно, а получилось другое  
print(dct2)  
dct2['P'] = 'Paul'  
dct2['R'] = 'Rodion'  
dct2['A'] = 'Ann'  
print(dct2) # теперь все так как и хотел

**Параграф7. Переменные логического типа.**

Переменные логического типа хранят в себе одно из двух значений True или False.

На этом все про логические переменные.

**Итоги главы**

В этой главе мы познакомились со всеми переменными и с тем, что с ними можно делать.

Основами работы яп python.

Задания по этой главе есть в конце пособия, настоятельно рекомендую их хотя бы просмотреть.