Тема 2.2 ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА PYTHON

1. Введение в язык программирования Python

Язык Python поддерживается всеми операционными системами (существуют версии для Linux, Windows, MacOS) и позволяет решать сложные математические задачи, создавать графические изображения, разрабатывать веб-сайты, работать с реляционными базами данных. Он используется для решения большого количества как научных, так и бизнес-задач.

Программы могут разрабатываться в консольном режиме (такие программы имеют расширение .py) и с графическим интерфейсом (программы имеют расширение .pyw).

Программа на языке Python — это обычный текстовый файл, инструкции (команды) которого выполняются интерпретатором для каждой строки. Программу на языке Python можно создавать и редактировать с помощью любого редактора, например, Notepad ++, Eclipse, Nano и др. Язык Python отличает скорость и простота скриптов.

Все данные языка, в том числе простые типы данных (числа, строки) являются ***объектами***. В переменной хранится не сам объект, а ссылка на него, то есть адрес ячейки памяти, в которой хранится объект.

Структура проекта на языке Python состоит из отдельных модулей (рис.1).

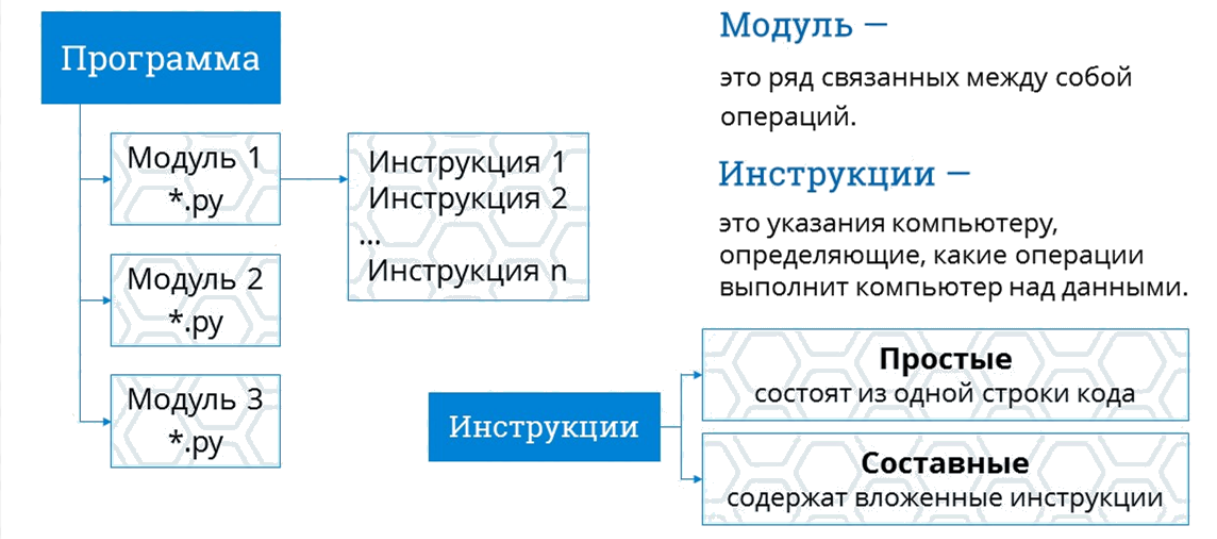

Рисунок 1 – Структура программы

Модуль — это любой файл с кодом. Количество таких модулей не ограничено. Один модуль может быть вложен в другой модуль, то есть применяется иерархическая структура модулей. Модули могут группироваться в пакеты. Модули могут разрабатываться самим программистом, а могут использоваться уже существующие в стандартной библиотеке языка. Один из модулей является главным, из него запускается проект на выполнение.

Модули, в свою очередь, состоят из более простых структурных единиц. В модулях содержится код на языке Python, состоящий из ***инструкций***. Инструкции представляют собой указания компьютеру. Они определяют, какие операции выполнит компьютер с данными. Инструкции в языке Python делятся на простые и составные. Простые инструкции описываются одной строкой кода, составные же – содержат вложенные инструкции.

Программа на Python, с точки зрения интерпретатора, состоит из логических строк. Одна логическая строка, как правило, располагается в одной физической, но

длинные логические строки можно явно (с помощью обратной косой черты) или неявно (внутри скобок) разбить на несколько физических:

print (a, " - очень длинная строка, которая не помещается в", \ 80, "знакоместах")

или

```
if (a == 1 and b == 2 and
    c == 3 and d == 4):
        print('spam' * 3)
```

Конец строки является концом инструкции (точка с запятой не требуется).

В Python отступ очень важен. Вложенные инструкции объединяются в блоки по величине отступов.

Когда основная инструкция завершается двоеточием, вслед за которым располагается вложенный блок кода, Python использует отступ для указания блока кода. Используют ***отступ - 4 пробела***.

Основная инструкция:

Вложенный блок инструкций

Python предоставляет возможность комментирования документации внутри кода. ***Комментарии*** следует начинать с символа #, а Python отобразит остальную часть строки в виде комментария:

 # Это комментарий

2. Основные элементы языка Python

Любая конструкция языка программирования начинается с алфавита. Из символов алфавита создаются лексемы (token). *Лексема* — это минимальная единица языка, которая имеет определенное самостоятельное значение.

Различают следующие виды лексем:

- ключевые (зарезервированные) слова (keywords);
- идентификаторы (identifiers);
- литералы (константы);
- операции;
- знаки препинания.

В языке Python используется несколько десятков *ключевых (зарезервированных) слов* (например, int, list, input, print, float и др.).

Идентификаторы (имена) используются для обозначения переменных, функций, которые создает программист, и других объектов. Имена переменных могут быть любыми. Однако есть несколько общих правил их написания:

1) Желательно давать переменным осмысленные имена, говорящие о назначении данных, на которые они ссылаются.

2) Имя переменной не должно совпадать с командами языка (зарезервированными ключевыми словами).

3) Имя переменной должно начинаться с буквы или символа подчеркивания _, но не с цифры.

4) Имя переменной не должно содержать пробелы.

Имена переменных используются для доступа к данным. Данные в языке Python представлены в форме объектов. То есть объект — это участок памяти с определенным значением и возможными операциями его обработки. Каждый объект имеет свой тип, например int (целое число), str (строка) и др.

В языке Python существуют базовые объектные типы (встроенные в язык) и разрабатываемые пользователем средствами самого языка или другими средствами.

Отметим, что переменные хранят не сам объект, а ссылку на объект, то есть адрес объекта в памяти компьютера.

В программе на языке Python связь между данными и переменными устанавливается с помощью ***оператора присваивания***, который обозначается знаком (=).

<имя переменной> = <значение переменной>

Выполняется оператор стандартным образом: сначала вычисляется выражения справа от знака равенства, а затем полученное значение записывается в переменную, указанную слева от знака равенства.

3. Типы данных в Python

Язык Python имеет множество встроенных типов данных. Все типы делятся на простые и составные (рис.2).

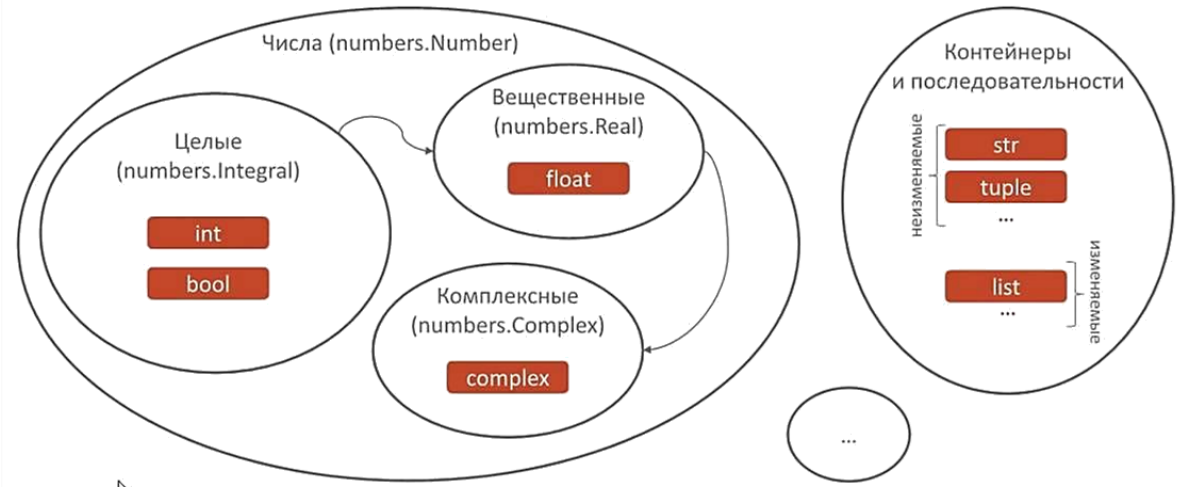

Рисунок 2 – Типы данных

Переменные простых типов состоят из единственного значения, к ним относятся *числа* и *логические переменные*.

Числа в Python делятся на:

- целые,
- действительные (с плавающей точкой размером в 64 бита),
- комплексные (с плавающей точкой размером в 128 бит).

Переменные составных типов состоят из набора значений и, в свою очередь, делятся на *неизменяемые*, для которых нельзя изменять значения элементов, добавлять или изымать элементы, и *изменяемые*, для которых всё это можно делать. К неизменяемым типам относятся строки и кортежи, к изменяемым — списки, словари и множества.

Основными типами данных в Python являются:

- **int** – целочисленные значения – положительные и отрицательные целые числа, а также 0 (например, 4, 687, -45, 0);
- **float** – вещественные (дробные) значения (например, 1.45, -3.789654, 0.00453). Для разделения целой и дробной частей здесь используется точка, а не запятая;
- **bool** – логические значения — значения могут принимать одно из двух значений: True (истина) или False (ложь);
- **str** – символьная строка или единичный символ - набор символов, заключенных в кавычки (например, "ball", "Whatisyourname?", 'd', '6589'). Кавычки в Python могут быть одинарными или двойными.

В языке Python применяется динамическая типизация данных. Это значит, что не нужно объявлять типы переменных, как это делается во многих языках программирования, поскольку их тип определяется автоматически в процессе присваивания им значений. Также, автоматически освобождается память от тех данных, которые становятся ненужными.

Этот тип определяется значением, расположенным справа от оператора присваивания (=).

>>>x = 44 # переменная x имеет тип int и значение 44

В одной строке можно присвоить одинаковые значения нескольким переменным, например:

>>> y = z = 1.10 # переменные y и z имеют тип float и значение 1.10

Если необходимо поменять значения местами, чтобы x = 1.10, а y = 44**,** тогда достаточно сделать так:

>x,y = y,x

Так же полезно запомнить, что для проверки типа любого значения и любой переменной можно использовать функцию type():

```
>>> a=5
>>> b=17.1
>>> c=4+2j
>>> type(a); type(b); type(c)
<class 'int'>
<class 'float'>
<class 'complex'>
```

4. Ввод и вывод данных

Ввод данных осуществляется при помощи оператора **input()**:

a = input()

В скобках функции можно указать сообщение-комментарий к вводимым данным:

a = input ("Введите количество: ")

Функция **input** воспринимает входные данные, как ***поток символов***.

```
a = input('Введите первое число: ')
b = input('Введите второе число: ')
print(a + b)
```

Ход работы программы:

```
Введите первое число: 10
Введите второе число: 4
104
```

Поэтому, чтобы принять целочисленное значение, следует воспользоваться функцией **int()**:

```
a = int(input('Введите первое число: '))
b = int(input('Введите второе число: '))
print(a + b)
```

Вывод программы:

```
Введите первое число: 10
Введите второе число: 4
14
```

Вывод данных осуществляется при помощи оператора **print()**:

```
a = 1
b = 2
print(a)
print(a + b)
print('сумма = ', a + b)
```

Внутри круглых скобок через запятую пишется то, что хотим вывести.

По умолчанию функция **print()** принимает несколько аргументов, выводит их через пробел, после чего ставит перевод строки. Это поведение можно изменить, используя именованные параметры **sep** (разделитель) и **end** (окончание).

В частности, если end сделать пустой строкой, то print не перейдет на новую строчку, и следующий print продолжит вывод прямо на этой же строке.

```
print('При')
print('вет!')
# эти две строки кода выведут "При" и "вет!" на отдельных строках
print('При', end='')
print('вет!') # эти две строки кода выведут "Привет!"
print('Раз', 'два', 'три') # выведет "Раз два три"
print('Раз', 'два', 'три', sep='--') # выведет "Раз--два--три"
```

Для форматированного вывода используется format:

```
x = 11
print ( "{:4d}".format(x) )
```

В результате выведется число 11, а перед ним два пробела, так как указано использовать для вывода четыре знакоместа.

Экранирующая последовательность. Если внутри кавычек встречается символ \ – обратная косая черта, обратный слеш, бэкслеш, он вместе с идущим после него символом образует экранирующую последовательность (escape sequence) и воспринимается интерпретатором как *единый специальный символ*.

В частности, \n — символ начала новой строки, \t — табуляция.

5. Операторы и выражения

Операции над объектами выполняются с помощью соответствующих операторов. Объекты, над которыми выполняются операции, называют *операндами*.

Оператор в Python — это символ, который выполняет операцию над одним или несколькими операндами. Каждый оператор может выполнять операции над строго определенными для него типами операндов.

В зависимости от типа объектов, над которыми выполняются операции, операторы группируются в арифметические, логические, сравнения, присваивания и др.

Операторы Python бывают 7 типов:

- Арифметические операторы
- Операторы сравнения
- Операторы присваивания
- Логические операторы
- Операторы принадлежности
- Операторы тождественности
- Битовые операторы

Арифметические операторы предназначены для выполнения операций над числами (таблица 1).

Таблица 1 - Арифметические операторы

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
**	Возведение в степень
//	Целочисленное деление. 25 // 6 в результате будет 4
%	Остаток от целочисленного деления. 7 % 2 в результате будет 1

Операторы сравнения сравнивают значения объекта, который находится слева от оператора, со значением объекта, который расположен справа от этого оператора. Если условие выполняется, возвращается значение True, иначе — False. Состав и обозначения операторов сравнения приведены в таблице 2.

Таблица 2 - Операторы сравнения

Оператор	Описание
==	Равно
!=	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

Оператор присваивания (таблица 3) присваивает значение переменной. Он может манипулировать значением до присваивания. Есть простой оператор присваивания и операторы с использованием арифметических операторов.

Таблица 3 - Операторы присваивания

Оператор	Описание
=	присваивание
+=	сложение и присваивание. x+= 8 (эквивалентно x = x + 8)
-=	вычитание и присваивание. x- = 8 (эквивалентно x = x - 8)
*=	умножение и присваивание. x*= 8 (эквивалентно x = x * 8)
/=	деление и присваивание. x/= 8 (эквивалентно x = x / 8)
**=	возведение в степень и присваивание. x**= 8 (эквивалентно x = x ** 8)

Логические операторы (таблица 4) выполняются над данными логического типа, имеют два значения: True (истинное) и False (ложно).

Таблица 4 - Логические операторы

Оператор	Описание
and	Логический оператор "И"
or	Логический оператор "ИЛИ"
not	Логический оператор "НЕ"

Операторы принадлежности (таблица 5) проверяют, является ли значение частью последовательности. Они предназначены для проверки наличия элемента в составных типах данных, таких, как строки, списки, кортежи или словари.

Таблица 5 - Операторы принадлежности

Оператор	Описание
in	Проверяет, является ли значение членом последовательности
not in	проверяет, НЕ является ли значение членом последовательности

Операторы тождественности (таблица 6) проверяют, являются ли операнды одинаковыми (занимают ли они одну и ту же позицию в памяти).

Таблица 6 - Операторы тождественности

Оператор	Описание
is	Возвращает истину, если элемент присутствует в последовательности, иначе возвращает ложь.
is not	Возвращает истину, если элемента нет в последовательности.

Побитовые операторы (таблица 7) предназначены для работы с данными в битовом (двоичном) формате. Эти операторы работают над операндами бит за битом. Предположим, что у нас есть два числа a = 60; и b = 13. В двоичном формате они будут иметь следующий вид:

a = 0011 1100

b = 0000 1101

Таблица 7 – Битовые операторы

Оператор	Описание
&	Бинарный "И" оператор, копирует бит в результат только если бит присутствует в обоих операндах
|	Бинарный "ИЛИ" оператор копирует бит, если тот присутствует в хотя бы в одном операнде
^	Бинарный "Исключительное ИЛИ" оператор копирует бит только если бит присутствует в одном из операндов, но не в обоих сразу
~	Бинарный комплиментарный оператор. Является унарным (то есть ему нужен только один операнд) меняет биты на обратные, там где была единица становиться ноль и наоборот
<<	Побитовый сдвиг влево. Значение левого операнда "сдвигается" влево на количество бит указанных в правом операнде
>>	Побитовый сдвиг вправо. Значение левого операнда "сдвигается" вправо на количество бит указанных в правом операнде

Операторы используются в выражениях. Понятие выражения в программировании аналогичное понятию выражения в математике.

Выражение в языке программирования состоит из операндов, операторов и круглых скобок и определяет порядок выполнения операций над данными. Операнды выражения — это переменные, константы, функции, методы. Самое простое выражение состоит из одного операнда, например константы или переменной.

В зависимости от типа операндов и операций, используемых в выражении, различают выражения: арифметические (результат арифметического типа),

логические (результат логического типа) и строчные (результат строчной типа). Для каждого типа операций существуют четкие правила их записи и исполнения.

В состав Python встроены функции:

abs(x) - Модуль числа x

int(x) – приведение вещественного числа x к целому, отбрасывание дробной части;

round(x) – округление вещественного числа x к ближайшему целому

pow(x, y) - полный аналог записи x ** y

len(x) - Возвращает число элементов в указанном объекте

max() - Максимальный элемент последовательности.

min() - Минимальный элемент последовательности.

sum() - Сумма элементов последовательности

Помимо стандартных выражений для работы с числами (а в Python их не так уж и много), в составе Python есть несколько полезных модулей. Например, *модуль math* предоставляет более сложные математические функции, *модуль random* реализует генератор случайных чисел и функции случайного выбора и др.

Загрузка модулей в Python осуществляется с помощью оператора **import**.

```
import math
# далее используем какую-либо функцию:
t = math.sin(math.pi/6)
print (math.sqrt(64))  # 8.0
```

Если название модуля слишком длинное, или оно вам не нравится по каким-то другим причинам, то для него можно создать псевдоним с помощью ключевого слова **as**:

```
import matplotlib.pyplot as plt
plt.plot(x)
```

Модуль math – один из наиважнейших в Python. Этот модуль предоставляет обширный функционал для работы с числами. Наиболее употребительные функции и константы модуля math:

trunc(X) - Усечение значения X до целого

ceil() - округление числа до ближайшего наибольшего целого

floor() - округление числа до ближайшего наименьшего целого

sqrt(X) - Квадратный корень из X

exp(X) - Экспонента числа X

log(X), log2(X), log10(X) - Натуральный, двоичный и десятичный логарифмы X

log(X, n) - Логарифм X по основанию n

sin(X), cos(X), tan(X) - Синус, косинус и тангенс X, X указывается в радианах

factorial(X) - Факториал числа X

pi - Выдаётся число pi

e - Выдаётся число e

Более подробное описание этих функций модуля math можно найти на сайте с документацией языка Питон или по ссылке https://pythonworld.ru/moduli/modul-math.html .

Функции для работы с псевдослучайными числами собраны в **модуле random**. Для получения псевдослучайных чисел в заданном диапазоне используют функции:

random() – случайное вещественное число из полуинтервала [0,1);

randint(a,b) – случайное целое число из отрезка [a,b].