Тема 1.1 Типы данных Python. Обзор. Простые типы данных. Числа. Строки. Переменные.

Определение типа данных.

Тип данных характеризует одновременно:

· множество допустимых значений, которые могут принимать данные, принадлежащие к этому типу;

· набор операций, которые можно осуществлять над данными, принадлежащими к этому типу.

Python поддерживает следующие типы данных:

· Числовые типы данных: int, float, complex;

· Строковый тип данных : str;

· Последовательности: list, tuple, range (список, кортеж, диапазон);

· Двоичные типы данных: bytes, bytearray, memoryview;

· Тип данных словарь: dict;

· Логические типы данных: bool; set, frozenset.

Как типы данных также рассматриваются функции, модули, файлы:

Для работы с данными используются переменные.

Переменная в Python указывает на область памяти компьютера, в которой хранятся данные.

Во многих языках программирования, прежде чем использовать переменные, их необходимо объявить, то есть указать, какого типа данные будут в них храниться. В языке Python такой необходимости нет. Тип данных переменной определяется автоматически в момент присвоения переменной конкретных данных.

Рассмотрим типы данных Python с примерами переменных.

Знак # обозначает, что далее до конца строки следует комментарий.

Оператор print используется для вывода данных на печать:

print(z) - на печать выводится переменная z.

В формате print(type(z)) на печать выводится тип данных переменной z.

Примеры.

Числовой тип данных

Числовой тип данных Python используется для хранения числовых значений. При этом тип данных int – содержит целые числа со знаком неограниченной длины, float – содержит числа с плавающей точкой с точностью до 15 десятичных знаков, комплексный – содержит комплексные числа.

```
#int - целые числа

z = 2147483647

print(z)

print(type(z))


# float - вещественные числа

y = 5.1

print(y)

print(type(y))


# complex - комплексные числа

z = 2+2i

print(z)

print(type(z))
```

Строковый тип данных (String)

Python поддерживает символы Unicode.

Строка представляет собой последовательность символов.

Строки заключаются в одинарные или двойные кавычки.

```
a = "строка в двойных кавычках"

b = 'строка в одинарных кавычках'


# str - строка

y = 'фламинго'

print(y)

print(type(y))
```

Другие типы данных

```
# bool - логический тип

z = True

z1 = False

print(z, z1)

print(type(z))

print(type(z1))


# list - список

arr = [1, 2, 3]

print(arr)

print(type(arr))
```

```
# tuple - кортеж

kar = (1, 2, 3)

print(kar)

print(type(kar))


# dict - словарь

dd = {"x": 5, "y": 45}

print(dd)

print(type(dd))


# set – множество

zm = set(["a", "f", "c"])

print(zm)

print(type(zm))


# frozenset - неизменяемое множество

zm = frozenset(["a", "f", "c"])

print(zm)

print(type(zm))


# NoneType - объект со значением None

no = None

print(no)

print(type(no))
```

```
# function - функция

def myFunc():pass

print(type(myFunc))


# module - модуль

import sys

print(type(sys))


# file - файл

f = open("zz.txt", "w")

print(type(f))
```

Типы данных **списки** и **словари** являются изменяемыми. Это значит, что их элементы можно изменять.

```
Arr = [1, 2, 3]

print(arr)

arr[0] = 5

print(arr)
```

Типы данных **числа, строки, кортежи** являются неизменяемыми.

Элементы этих типов данных изменять нельзя.

```
stroka = 'Елена'

print(stroka)

print(stroka[0])  # вывести первый символ - выводится 'Л'
```

```
#stroka[0] = 'Г'     # попытка изменить первый символ выдаст ошибку


stroka = 'вася'     # изменить значение строки целиком (переприсвоить) можно

print(stroka)
```

Строки, списки, кортежи являются последовательностями и

поддерживают обращение к их элементам по индексу.

Также к ним применимы операции получение среза, конкатенация (+), повторение (*), проверка на вхождение (in).

В примере ниже оператор цикла for применяется для поэлементного

перебора списка.

```
arr = [1,2,3]

for i in arr:     # индекс i –обращение к очередному элементу списка

    print(i)


stroka = 'Строка'

for i in stroka:

    print(i)
```