

УРОК 1

Типы
данных

1

Списки, кортежи,
множества и
словари

3

Обсуждения

5

2

Числа, строки и
логические
СИМВОЛЫ

4

Практика



1.

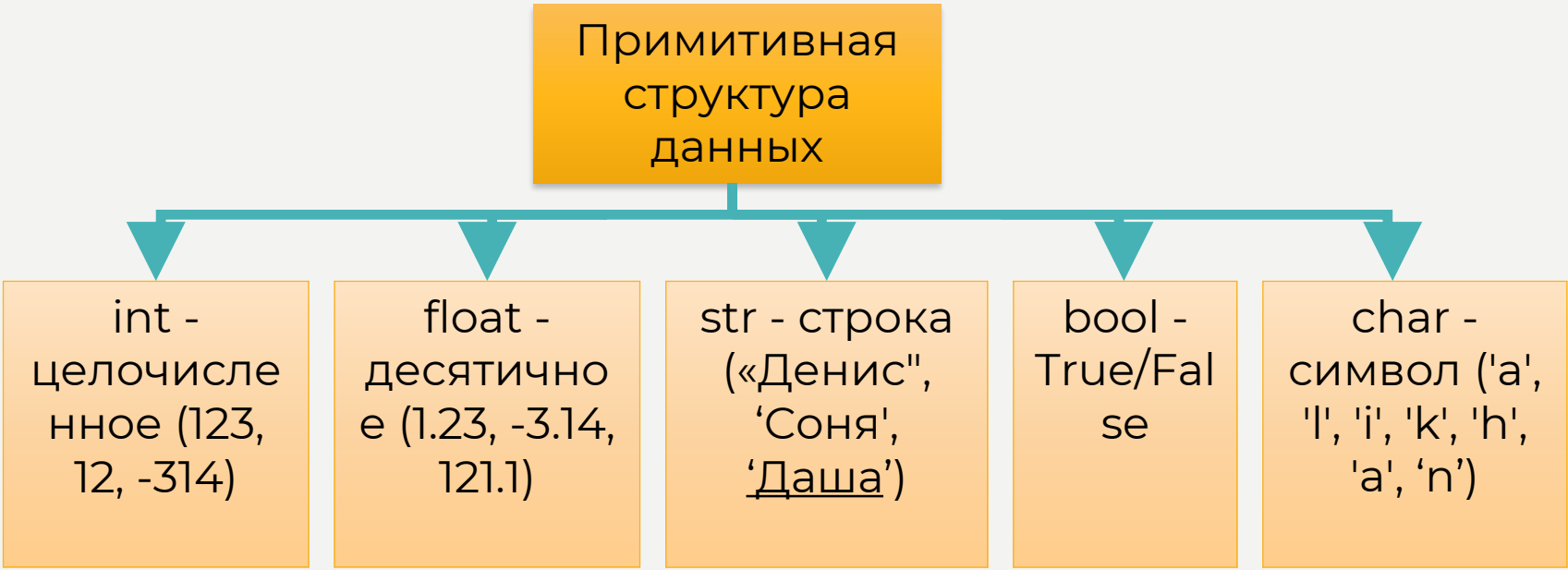
ТИПЫ ДАННЫХ



ПРИМИТИВНЫЕ СТРУКТУРЫ ДАННЫХ

Примитивная структура данных - это чистый и простой тип данных который может хранить значение определенного типа.

Примитивная структура данных



```
graph TD; A[Примитивная структура данных] --> B[int - целочисленное (123, 12, -314)]; A --> C[float - десятичное (1.23, -3.14, 121.1)]; A --> D[str - строка («Денис», 'Соня', 'Даша')]; A --> E[bool - True/False]; A --> F[char - символ ('a', 'l', 'i', 'k', 'h', 'a', 'n')];
```

int -
целочисле
нное (123,
12, -314)

float -
десятично
е (1.23, -3.14,
121.1)

str - строка
(«Денис»,
'Соня',
'Даша')

bool -
True/Fal
se

char -
символ ('a',
'l', 'i', 'k', 'h',
'a', 'n')

НЕПРИМИТИВНЫЕ СТРУКТУРЫ ДАННЫХ

Непримитивная структура данных - это тип данных, который содержит набор примитивных типов данных и может иметь различные форматы.

Не примитивные структуры данных

```
graph TD; A[Не примитивные структуры данных] --> B[list - массив = ['Codemode', 2023, 3.14]]; A --> C[tuple - кортеж = (4, 5, 6)]; A --> D[set - уникальная коллекция = ('Hello', 2023)]; A --> E[dict - словарь = {'name': 'codemode', 'age': 2}];
```

list -
массив =
['Codemode', 2023,
3.14]

tuple -
кортеж=
(4, 5, 6)

set -
уникальная
коллекция
= ('Hello',
2023)

dict -
словарь = {
"name":
"codemode",
"age": 2 }



2.

ЧИСЛА, СТРОКИ И ЛОГИЧЕСКИЕ СИМВОЛЫ

ЧИСЛА

1

Integer - это числовой тип данных, который обычно используется при работе с целыми числами. Он также распознает положительные и отрицательные значения (знаковые числа).



Пример: -1, -2, -3, 0, 1, 2, 3, 4

2

Тип данных Float представляет собой число, в котором присутствует запятая. Тип float обычно используется для обозначения точных значений, например, одинарной точности.



Пример: 3.14, 0.5, 2.5, 10.5

Код программы:



```
a = 10  
print (a, "Тип", type(a))  
b = 1.7  
print (b, "Тип", type(b))
```

БУЛЕВОЕ ЗНАЧЕНИЕ

Булево значение в Python — это тип данных, который может принимать только два значения: True (истина) и False (ложь). Булевы значения часто используются в логических операциях и условных выражениях, чтобы принимать решения в программе. Например, они могут определять, выполняется ли условие, или использоваться для управления потоком выполнения (например, в операторе if).



Пример: True или False

Код программы:



```
num1 = 6  
num2 = 5  
ouput_boolean = num1 == num2  
print(ouput_boolean)
```

Код программы:



```
num1 = 6  
num2 = 5  
ouput_boolean = num1 > num2  
print(ouput_boolean)
```

СТРОКА

Строка в Python — это последовательность символов, используемая для хранения текста. Строки неизменяемы, что означает, что после создания их нельзя изменить. Вы можете использовать строки для выполнения различных операций, таких как объединение, разделение и извлечение подстрок. Строки заключаются в одинарные ('), двойные ("), тройные одинарные (""") или тройные двойные ("""") кавычки.



Пример: "Интерес", 'Привет', '2', '-5' dll

Код программы:



```
single_quote_string = 'Это строка, используя одинарные кавычки'  
print(single_quote_string)
```

Код программы:



```
double_quote_string = "Это строка, используя двойные кавычки"  
print(double_quote_string)
```

Код программы:



```
multi_line_single = '''Это многострочная строка,  
которая использует одинарные кавычки'''  
print(multi_line_single)
```




3.

СПИСКИ, КОРТЕЖИ, МНОЖЕСТВА И СЛОВАРИ



ЛИСТЫ

Список (лист) в Python — это упорядоченная коллекция элементов, которую можно изменять. Списки позволяют хранить несколько значений (например, числа, строки или другие списки) в одном объекте. Вы можете добавлять, удалять и изменять элементы в списке.



- `list1 = ['Химия', 'Физика', 1993, 2021]`
- `list2 = [1, 2, 3, 4, 5]`
- `list3 = ["А", "Б", "В", "Г", "Д"]`

Код программы:

```
my_list = [1, 2, 3, 4, 5]
print("Список:", my_list)
# Доступ к элементам списка по индексу
first_element = my_list[0]
print("Первый элемент:", first_element)
# Изменение элемента списка
my_list[1] = 20
print("Измененный список:", my_list)
# Добавление элемента в конец списка
my_list.append(6)
print("Список после добавления элемента:", my_list)
# Вставка элемента в список по индексу
my_list.insert(2, 10)
print("Список после вставки элемента на индекс 2:", my_list)
# Удаление элемента по значению
my_list.remove(4)
print("Список после удаления элемента 4:", my_list)
# Удаление элемента по индексу
popped_element = my_list.pop(0)
print("Извлеченный элемент:", popped_element)
print("Список после удаления первого элемента:", my_list)
# Длина списка
list_length = len(my_list)
print("Длина списка:", list_length)
```

КОРТЕЖИ

Кортеж в Python — это упорядоченная коллекция элементов, которая, в отличие от списков, является неизменяемой. Это означает, что после создания кортежа вы не можете изменять его элементы. Кортежи могут содержать элементы разных типов и обычно используются для хранения связанных данных. Они заключаются в круглые скобки `()`, и для создания кортежа с одним элементом нужно добавлять запятую (например, `(1,)`).



- `tup1 = ('Сергей', 'Банан', 1993, 2021)`
- `tup2 = (1, 2, 3, 4, 5)`

Код программы:



```
# Создание кортежа
my_tuple = (1, 2, 3, 4, 5)
print("Кортеж:", my_tuple)

# Доступ к элементам кортежа по индексу
first_element = my_tuple[0]
print("Первый элемент кортежа:", first_element)

# Срез кортежа
sub_tuple = my_tuple[1:4] # Элементы с индексами от 1 до 3
print("Подкортеж:", sub_tuple)

# Создание списка, содержащего кортежи
list_of_tuples = [(1, 'apple'), (2, 'banana'), (3, 'cherry')]
print("Список кортежей:", list_of_tuples)

# Доступ к элементам списка кортежей
first_tuple = list_of_tuples[0]
print("Первый кортеж из списка:", first_tuple)

# Доступ к элементам кортежа из списка
fruit_name = list_of_tuples[1][1] # Доступ к элементу 'banana'
print("Название фрукта из второго кортежа:", fruit_name)
```

МНОЖЕСТВА

Множество в Python — это неупорядоченная коллекция уникальных элементов, которые не могут повторяться. Множества используются для хранения различных значений и выполнения операций, таких как объединение, пересечение и разность. Они заключаются в фигурные скобки ({}), или создаются с помощью функции `set()`. Множества удобны, когда нужно избавиться от дубликатов и быстро проверять наличие элемента.



- `set1 = {«Яблоко", «Банан", «Интерес"}`
- `set2 = {«АБВ", 34, Истина, 40, «Мужчина"}`

Код программы:



```
# Создание множества
my_set = {1, 2, 3, 4, 5}
print("Множество:", my_set)
# Добавление элемента в множество
my_set.add(6)
print("Множество после добавления 6:", my_set)
# Удаление элемента из множества
my_set.remove(2)
print("Множество после удаления 2:", my_set)
```

СЛОВАРИ

Словарь в Python — это неупорядоченная коллекция пар "ключ-значение". Он позволяет хранить данные в виде ассоциативных массивов, где каждый ключ уникален и связывается с определённым значением. Словари удобно использовать для быстрого поиска значений по ключам. Словари создаются с помощью фигурных скобок ({}), или функции dict(). Ключи могут быть строками, числами или кортежами, а значения могут быть любыми типами данных.



```
dic = {'Имя' : 'Сергей', 'Россия' : 23, 'Работа' : 'Дизайнер'}
```


Код программы:

```

# Создание словаря
my_dict = {
    'name': 'Alice',
    'age': 30,
    'city': 'New York'
}
print("Словарь:", my_dict)
# Доступ к значению по ключу
name = my_dict['name']
print("Имя:", name)
# Изменение значения по ключу
my_dict['age'] = 31
print("Обновленный словарь:", my_dict)
# Добавление новой пары ключ-значение
my_dict['job'] = 'Engineer'
print("Словарь после добавления новой пары:", my_dict)
# Удаление элемента по ключу
del my_dict['city']
print("Словарь после удаления города:", my_dict)
```

УРОК 2

Ввод с
клавиатуры

1

Списки, кортежи,
множества и
словари

3

Обсуждения

5

2

Математические
операции

4

Практика



1.

ВВОД С КЛАВИАТУРЫ



ВВОД С КЛАВИАТУРЫ

Для ввода текста с клавиатуры в Python используется встроенная функция `input()`. Эта функция позволяет пользователю ввести данные, которые затем можно использовать в программе.



```
name = input()
```

Код программы:



```
# Запрашиваем имя пользователя
name = input("Введите ваше имя: ")
# Выводим приветственное сообщение
print(f"Привет, {name}!")
```

ВВОД ЧИСЛЕННЫХ ЗНАЧЕНИЙ С КЛАВИАТУРЫ

Чтобы вводить целые или вещественные числа с клавиатуры, можно использовать уже знакомую нам функцию “input”, но в сочетании с рассмотренными в первом уроке функциями “float”, “int”.



```
int_name = int(input())  
float_name = float(input())
```

Код программы:



```
# Ввод числового значения и преобразование в тип float
number = float(input("Введите число: "))
# Выводим результаты
print(f"Вы ввели число: {number}")
```



2.

МАТЕМАТИЧЕСК ИЕ ОПЕРАЦИИ



МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ

Для работы с числовыми переменными доступны следующие математические операции (больше операций в методическом указании):



Сложение – $a + b$;

Вычитание – $a - b$;

Умножение – $a * b$;

Деление – a / b ;

Возведение в степень – $a ** b$;

Целочисленное деление – $a // b$;

Остаток от деления – $a \% b$.

Код программы:



```
# Ввод двух чисел пользователем
num1 = float(input("Введите первое число: "))
num2 = float(input("Введите второе число: "))
# Выполнение математических операций
addition = num1 + num2      # Сложение
subtraction = num1 - num2   # Вычитание
multiplication = num1 * num2 # Умножение
division = num1 / num2      # Деление
modulus = num1 % num2       # Остаток от деления
exponentiation = num1 ** num2 # Возведение в степень
# Вывод результатов
print(f"Сложение: {num1} + {num2} = {addition}")
print(f"Вычитание: {num1} - {num2} = {subtraction}")
print(f"Умножение: {num1} * {num2} = {multiplication}")
print(f"Деление: {num1} / {num2} = {division}")
print(f"Остаток от деления: {num1} % {num2} = {modulus}")
print(f"Возведение в степень: {num1} ** {num2} = {exponentiation}")
```