

Python






Краткий план курса

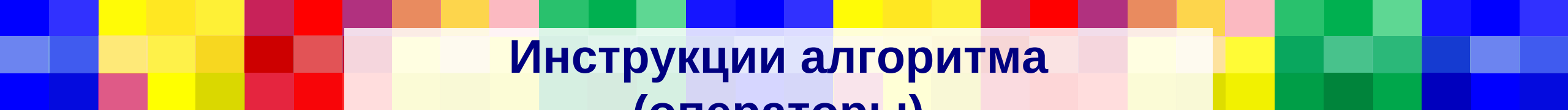
- 12 лекций
- 12 практик
- ДЗ

- Блок 1: Основы программирования на языке Python
- Блок 2: Объектно-ориентированное программирование на языке Python
- Блок 3: Программирование на языке Python в машинном обучении и анализе данных
- ?



Алгоритм

- Чтобы заставить компьютер сделать что-либо, инструкции должны быть записаны в виде компьютерной программы.
 - В информатике алгоритм - это конечная последовательность четко определенных, реализуемых компьютером инструкций для решения какой-то проблемы или для выполнения вычислений. Алгоритмы всегда однозначны и используются для вычислений, обработки данных, автоматизированных решений и других задач.
- 



Инструкции алгоритма (операторы)

- Последовательные
- Условные
- Циклы



Python

- Язык - это инструмент для представления (кодирования) ИНФОРМАЦИИ: данных или последовательности действий.
- Python - язык высокого уровня.
- Python – интерпретируемый язык программирования.



+++++

- Прост в изучении;
- компактный код;
- полностью поддерживается в качестве языка с открытым исходным кодом;
- надежная многоплатформенную поддержку;
- многочисленные среды разработки;
- Силен в исследовательских, и в инженерных задачах



Интерпретаторы

- Python IDE
 - PyCharm
 - Visual Studio Code
 - И др.
-
- Расширение файлов исходников .py

Ввод\Вывод результатов и комментирование

- `# print("Hello, students! Welcome to our course! ")`

`#single comment`

```
'''  
Multi-line comments example  
'''
```

- Строки игнорируются

Встроенные типы языка Python

- Тип данных (англ. Data type) - характеристика, определяющая:
- множество допустимых значений, которые могут принимать данные, принадлежащие к этому типу (например, объект типа Целое число может принимать только целочисленные значения в определенном диапазоне);
- набор операций, которые можно осуществлять над данными, принадлежащими к этому типу (например, объекты типа Целое число умеют складываться, умножаться и т.д.).
- Все типы в Python являются объектами. При создании объекта вызывается специальная функция - конструктор.



Переменная

- Переменная - это идентификатор, который указывает на определенную область памяти, где хранятся произвольные данные - созданный объект (значение переменной).

Правила именования переменных

- первый символ должен быть заглавной или строчной латинской буквой или нижним подчёркиванием _;
- остальные символы могут быть заглавными или строчными латинскими буквами, нижними подчёркиваниями и цифрами;
- нельзя использовать пробелы;
- имя переменной не должно совпадать ни с каким из зарезервированных в Python ключевых слов.

Типы данных

- Скалярные (неделимые).
 - Числа (целое, вещественное).
 - Логический тип.
 - NoneType.
- Структурированные (составные) / коллекции.
 - Последовательности: строка, список, кортеж, числовой диапазон.
 - Множества.
 - Отображения: словарь.



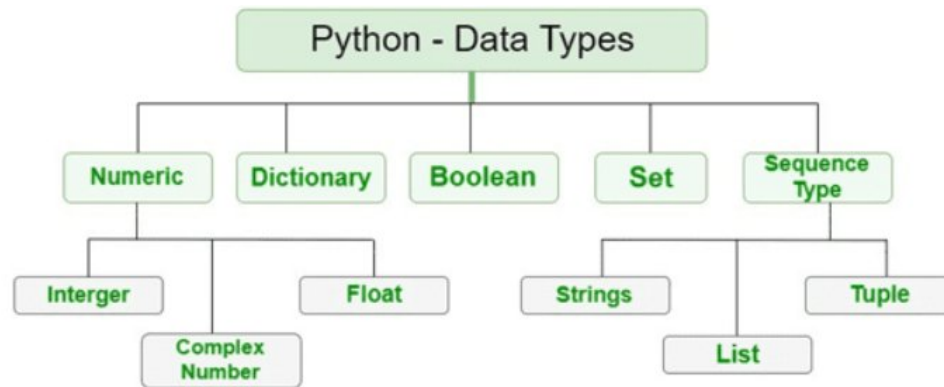
Типы данных

- Изменяемые (англ. Mutable): содержимое объекта можно изменить после создания (например, список);
- Неизменяемые(англ. Immutable): содержимое объекта нельзя изменить после создания (например, строка или число).

Оператор присваивания

- (пример)

Типы данных



+ complex

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey": "value" , "name": "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False

Целые числа

- `int()`
- **0b** (0B) – для двоичного представления;
- **0o** (0O) – для восьмеричного представления;
- **0x** (0X) – для шестнадцатеричного представления.



Вещественные числа

- `float()`



complex

- $T = 4 + 3j$
- `complex(3,4)`

Операции с числами

Operators	Meaning	Example	Result
+	Addition	$4 + 2$	6
-	Subtraction	$4 - 2$	2
*	Multiplication	$4 * 2$	8
/	Division	$4 / 2$	2
%	Modulus operator to get remainder in integer division	$5 \% 2$	1
**	Exponent	$5 ** 2 = 5^2$	25
//	Integer Division/ Floor Division	$5 // 2$ $-5 // 2$	2 -3

Operator	Meaning	Example
&	Bitwise AND	$x \& y = 0$ (0000 0000)
	Bitwise OR	$x y = 14$ (0000 1110)
~	Bitwise NOT	$\sim x = -11$ (1111 0101)
^	Bitwise XOR	$x \wedge y = 14$ (0000 1110)
>>	Bitwise right shift	$x >> 2 = 2$ (0000 0010)
<<	Bitwise left shift	$x << 2 = 40$ (0010 1000)

boolean

- true/false
- Строка=true, искл – пустая строка
- Число = true, искл - 0
- Список, множество, кортеж – true. искл – пустые объекты

- bool()

Operator	Meaning
== (double equal to)	Equal to
<	Less than
>	Greater than
!=	Not equal to
<=	Less than or equal to
>=	Greater than or equal to




set

- $S = \{1, 't', .1, \text{false}\}$
- `set()`
- Неупорядоченный
- Неизменяемый
- Гетерогенный
- Уникальные значения



list

- `L = [1, 'ttt', .4, false]`
 - `list()`
 - Упорядоченный
 - Изменяемый
 - Гетерогенный
 - Возможны дублированные значения
 - Индексация – прямая, обратная (отрицательные значения индексов)
- 



tuple

- T = (20, 'str', .3, [3, 'tt'])
- tuple()
- Упорядоченный
- Неизменяемый
- Гетерогенный
- Допускает повторяющиеся значения



dict

- $D = \{ 'a': 10, 'b': 30 \}$
- `D = dict()`
- Неупорядоченный
- Уникальный по ключу
- изменяемый



str

- Str() – неизменяемый тип данных



Больше про типы данных и методы работы с ними

- [5. Data Structures — Python 3.12.6 documentation](#)

If elif else

```
x = -2
if x == 5:
    print("x is 5")
elif x < 5:
    print("x is less than 5")
else:
    print("x is greater than 5")
print("Done!")
```

match

```
valueofS= int(input())
match valueofS:
    case 10:
        print("U entered 10")
    case 20:
        print("U entered 20")
    case 30:
        print("U entered 30")
    case _:
        print("don't enter 10 or 20 or 30")
```

while

```
while условие:  
    # выполнить эти операторы  
else:  
    # выполнить эти операторы
```

for

```
list = ["geeks", "for", "geeks"]  
for index in range(len(list)):  
    print(list[index])
```

```
n = 4  
for i in range(0, n):  
    print(i)
```

```
list = ["geeks", "for", "geeks"]  
for index in range(len(list)):  
    print(list[index])  
else:  
    print("Inside Else Block")
```

```
fruits = ["apple", "orange", "kiwi"]  
  
for fruit in fruits:  
    print(fruit)
```

Break, continue, pass

Оператор break

Оператор `break` позволяет досрочно прервать цикл:

- `break` прерывает текущий цикл и продолжает выполнение следующих выражений
- если используется несколько вложенных циклов, `break` прерывает внутренний цикл и продолжает выполнять выражения, следующие за блоком * `break` может использоваться в циклах `for` и `while`

Оператор continue

Оператор `continue` возвращает управление в начало цикла. То есть, `continue` позволяет «перепрыгнуть» оставшиеся выражения в цикле и перейти к следующей итерации.

Оператор pass

Оператор `pass` ничего не делает. Фактически, это такая заглушка для объектов.

Например, `pass` может помочь в ситуации, когда нужно прописать структуру скрипта. Его можно ставить в циклах, функциях, классах. И это не будет влиять на исполнение кода.