«Московский физико-технический институт»
Физтех-школа радиотехники и компьютерных технологий

Конспект лекций по курсу «Алгоритмы на Python3»

(по видеозаписям лекций 2017-2018 учебного года)

Верстка:

Насыров А. Р.

Преподаватель:

Хирьянов Т. Ф.

г. Долгопрудный, 2026

СОДЕРЖАНИЕ

I ЛЕКЦИЯ 1. СИНТАКСИС PYTHON

I.1 Hello, World!

Простейшая программа на Python – это пустая программа: если мы создадим пустой файл, то это будет корректной программой на Python.

Более сложная программа – это программа, содержащая команду `print`:

```
print
```

Это программа тоже ничего не делает, она лишь упоминает функцию `print`, которая предназначена для вывода информации на экран, однако это тоже корректная программа на Python.

Следующий шаг – это программа, которая вызывает функцию `print`:

```
print()
```

Здесь мы вызываем функцию `print` без аргументов, и она просто выводит пустую строку. Это уже более осмысленная программа, которая выполняет действие.

Если мы хотим вывести что-то конкретное, например, строку "Hello, World! то мы можем написать:

```
print("Hello, World!")
```

I.2 Концепция присваивания

Однако хотелось бы использовать память компьютера для хранения данных, а не только для выполнения команд. Для этого в Python есть переменные. Переменная – это именованная область памяти, которая может хранить данные. Например, мы можем создать переменную `x` и присвоить ей строку "Hello, World!":

```
x = "Hello, World!"
print(x)
```

В данном примере `x` – это имя, которое ссылается на строку "Hello, World!"в памяти. Это объект типа `str` (строка). Чтобы узнать тип объекта, мы можем использовать функцию `type`:

```
print(type(x))
```

Зачем нужно давать имена объектам? Это позволяет нам обращаться к данным, которые хранятся в памяти, используя эти имена. То есть мы ссылаемся на объекты через их имена.

Есть тонкость: справа может быть не только объект, но и выражение. Например:

```
x = 1 + 2 + 3 * 2
print(x)
```

В этом случае действия интерпретатора следующие: он сначала вычисляет выражение `1 + 2 + 3 * 2` – для этого ему понадобятся временные неименованные объекты, которые будут удалены после вычислений благодаря сборщику мусора (очистке памяти), удаляющий объекты, на которые нет ссылок. После чего результат вычисления (число 9) будет сохранен в памяти, и имя `x` будет ссылаться на этот объект.

Что будет с объектом после двух и боллее ссылок? Он будет удален, когда на него не будет ссылок. Например:

```
x = "Hello, World!"
print(type(x))
x = 1 + 2 + 3 * 2
```

В этом примере строка "Hello, World!"будет удалена из памяти после того, как мы присвоим `x` новое значение в результате операции связывания.

I.3 Обмен двух переменных значениями

Рассмотрим простейший алгоритм обмена значениями двух переменных `a` и `b`:

```
a = 2
b = 5
tmp = a
a = b
b = tmp
```

После выполнения этого кода переменная tmp останется висеть в памяти далее по коду, так как на нее есть ссылка.

Однако в Python есть более простой способ обмена значениями двух переменных через две переменных:

```
a = 2
b = 5
tmp1 = a
tmp2 = b
a = tmp1
b = tmp2
```

I.4 Множественное присваивание

```
x, y, z = 1, 2, 3
```

В этом случае мы пользуемся понятием кортежа. Кортеж – это упорядоченный набор объектов, который может содержать объекты разных типов.

Тогда обмен значениями двух переменных можно записать так:

```
a = 2
b = 5
a, b = b, a
```

При этом время на копирование объектов не тратится, так как мы просто создаем ссылки на эти объекты.

I.5 Арифметические операции

1. Возведение в степень: $x^y \equiv x ** y$ ($\sqrt{3} = 3\ ** 0.5,\ a^{b^c} = a ** b\ ** c$)
2. Унарный минус: $-x \equiv -x$
3. Умножение: $x \cdot y \equiv x * y$
4. Деление: $x : y \equiv x/y$
5. Целочисленное деление: $x \div y \equiv x\ //\ y$
6. Остаток от деления: $x\ \bmod y \equiv x\ \%\ y$
7. Сложение и вычитание: $x + y \equiv x + y,\ x - y \equiv x - y$

I.6 Цикл while

```
while условие:
    оператор 1
    if условие 2:
        break
    оператор 2
else:
    после всех итераций цикла
```