ПМ2 Прикладное программирование.

РО 2.1. Создавать консольные приложения на Python.

Тема 2. Операторы, операнды, приоритет операций и базовые типы данных в Python.

Цель занятия:

Сформировать у студентов понимание базовых строительных элементов программ на Python: операторов и операндов, основных типов данных, а также научить использовать арифметические операции для построения простых выражений.

Учебные вопросы:

1. Понятие оператора и операнда в Python. Унарные и бинарные операторы.

2. Основные арифметические операции. Приоритет операций.

3. Переменные в Python. Оператор присвоения.

4. Простые типы данных в Python. Преобразование типов.

1. Понятие оператора и операнда в Python.

Что такое выражение?

В Python выражение — это комбинация операторов, операндов и функций, которая вычисляется в некоторое значение.

Примеры выражений:

```
5 + 3
x - 2
"Hello" + "World"
```

Оператор

Оператор — это символ или ключевое слово, которое выполняет действие над одним или несколькими операндами.

Виды операторов:

- Арифметические: +, -, *, /, //, %, **
- Сравнения: ==, !=, <, >, <=, >=
- Логические: and, or, not
- Присваивания: =, +=, -=, *= и т.д.
- Другие.

Примеры:

```
5 + 3    # оператор сложения
x = 10 # оператор присвоения
```

Операнд

Операнд — это значение или объект, над которым выполняется операция.

Операндами могут быть:

- Числа (5, 3.14)
- Переменные (x, y)
- Строки ("Hello")
- другие типы

Примеры:

```
5 + 3          # операнды: 5 и 3
x - 2          # операнды: x и 2
"Hello" + "World"  # операнды: "Hello" и "World"
```

Выражение в Python состоит из оператора и операндов

Связь оператора и операндов

Оператор применяет своё действие к операндам.

Без операндов оператор не может вычислить значение.

Пример:

```
a = 10        # присваивание: оператор =, операнды: a и 10
b = a * 2     # умножение: оператор *, операнды: a и 2
```

Типы выражений по количеству операндов

Тип	Количество операндов	Пример
Унарное	1	-x
Бинарное	2	a + b
Тернарное	3	x if condition else y

Итог:

- Оператор — выполняет действие
- Операнд — объект/значение, над которым действует оператор
- Выражение = комбинация операторов и операндов, которая вычисляется в значение

2. Основные арифметические операции. Приоритет операций.

Назначение арифметических операторов в Python — выполнение элементарных математических действий над числами или переменными, которым присвоено числовое значение.

Оператор	Описание	Пример	Результат
+	Сложение	7 + 3	10
-	Вычитание	7 - 3	4
*	Умножение	7 * 3	21
/	Деление (истинное)	7 / 3	2.3333333333333335
**	Возведение в степень	7**3	343
//	Целочисленное деление	7 // 3	2
%	Остаток от деления	7 % 3	1

Особенности операций:

- Деление **/** всегда возвращает число типа float.
- Целочисленное деление **//** отбрасывает дробную часть, возвращая целое число.
- Остаток **%** полезен для проверки четности числа (x % 2 == 0) или циклических операций.
- Возведение в степень **** позволяет вычислять степени и корни (x ** 0.5 — квадратный корень).
- Унарный минус (**-**) используется для изменения знака числа. Он работает только с одним операндом:

Примеры:

```
x = 10
y = 3

# Основные операции
print(x + y)   # 13
print(x - y)   # 7
print(x * y)   # 30
print(x / y)   # 3.3333...
print(x // y)  # 3
print(x % y)   # 1
print(x ** y)  # 1000

# Унарный минус
print(-x)      # -10
```

Приоритет арифметических операций

Оператор	Операция	Приоритет
()	Скобки	1
**	Степень	2
*	Умножение	3
/	Деление	3
//	Целочисленное деление	3
%	Деление по модулю	3
+	Сложение	4
-	Вычитание	4

Приоритет операций в Python:

- Скобки () — выполняются первыми.
- Возведение в степень **.
- Унарные +x, -x.
- Умножение, деление, целочисленное деление, остаток: *, /, //, %.
- Сложение и вычитание: +, -.

Приоритет операций аналогичен стандартным математическим правилам.

Примеры:

```
print(2 + 3 * 4)        # 14, умножение раньше сложения
print((2 + 3) * 4)      # 20, скобки меняют порядок
print(-2 ** 2)          # -4, унарный минус после степени
print((-2) ** 2)        # 4, скобки изменяют порядок
```

Округление чисел. Функция round()

Функция round() в Python используется для округления чисел до заданного количества знаков после запятой или до ближайшего целого.

Синтаксис функции: round(number, ndigits). 1

Параметры:

- number — число, которое нужно округлить;
- ndigits (необязательно) — число, до которого округляется данное число, по умолчанию 0.

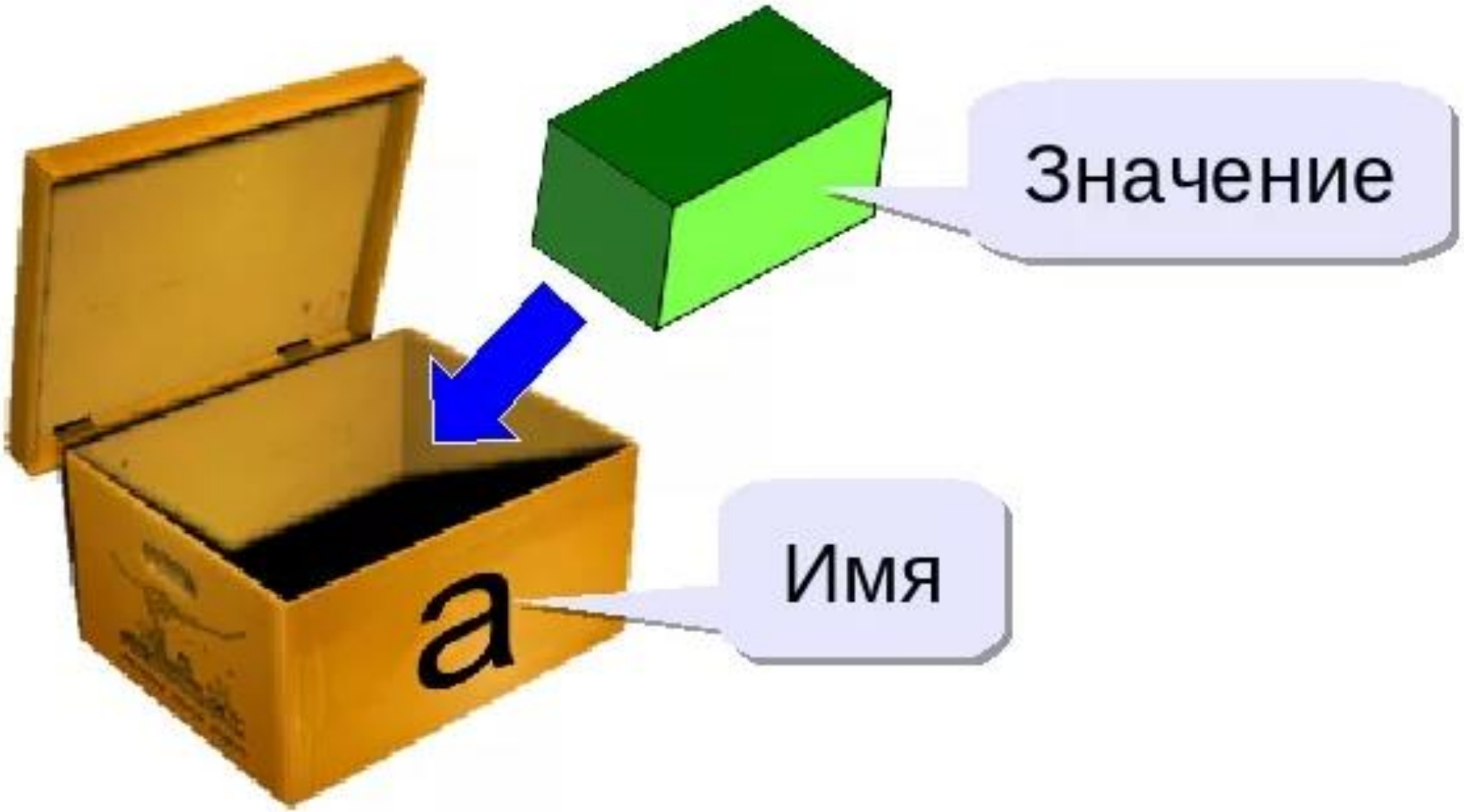
Пример:

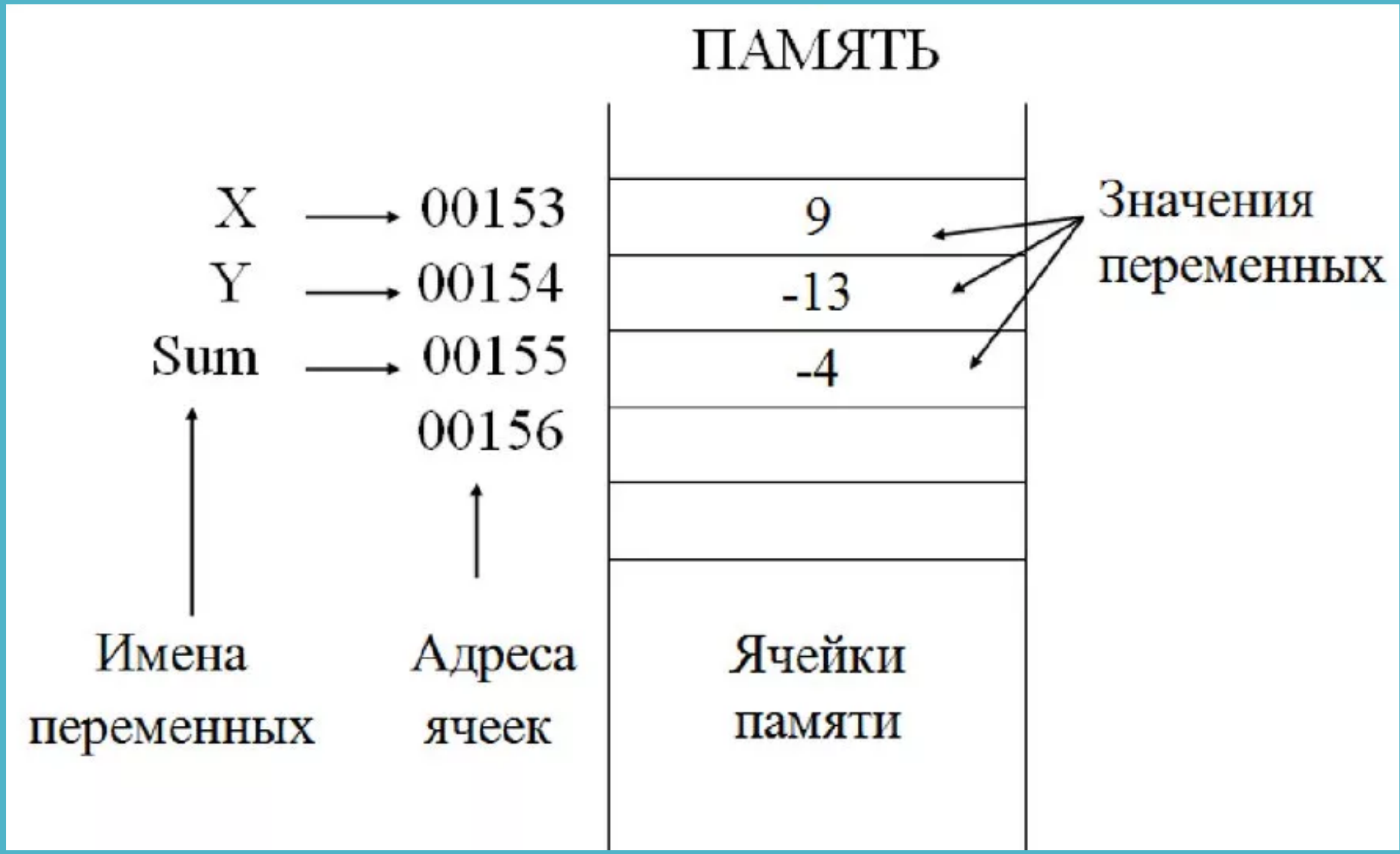
```
pi = 3.1415926535
e = 2.71828182
price = 5.99


print(round(pi, 4)) # 3.1416
print(round(e, 2))  # 2.72
print(round(price)) # 6
```

3. Переменные в Python. Оператор присвоения.

Переменная — это именованная ячейка памяти, в которой хранится значение.

С помощью переменных можно сохранять данные, изменять их и использовать в вычислениях.


Значение
a
Имя

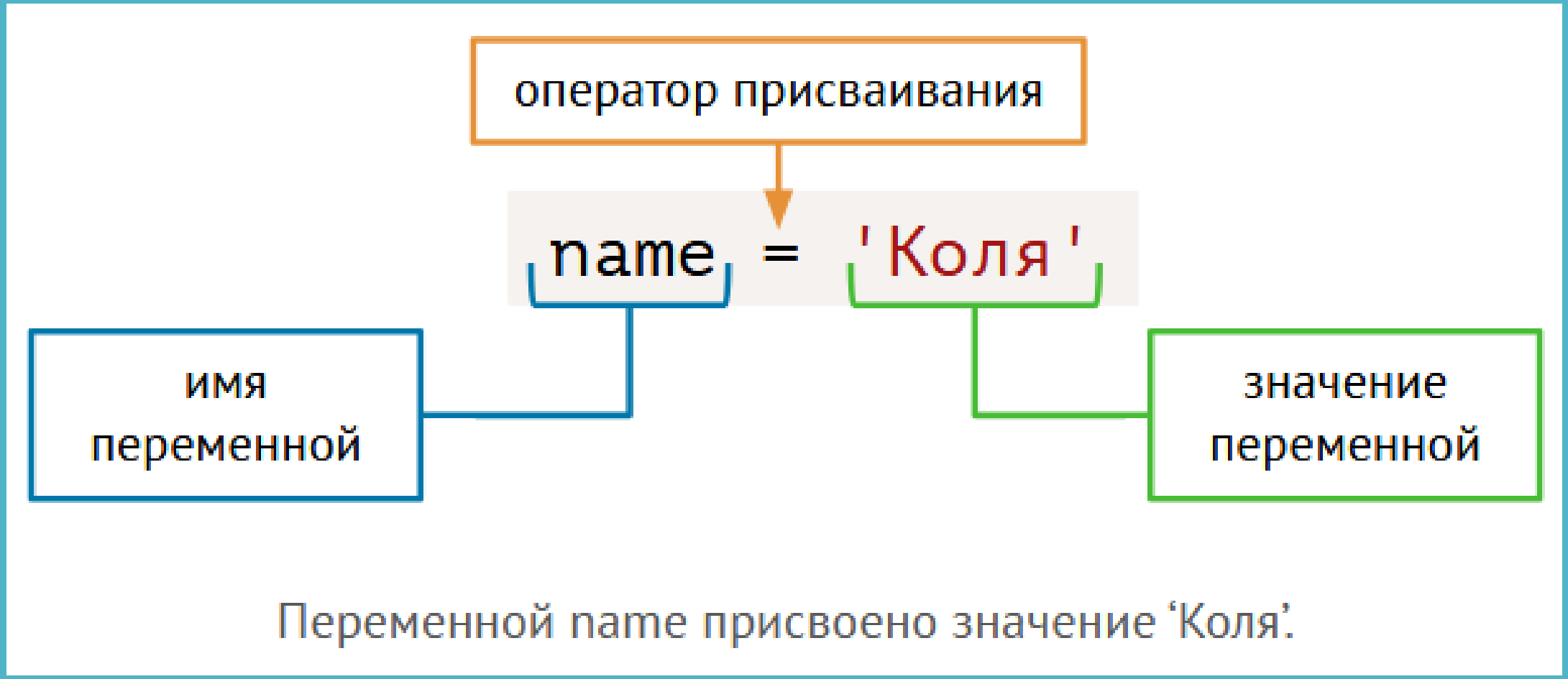

ПАМЯТЬ
X
Y
Sum
00153
00154
00155
00156
9
-13
-4
Значения переменных
Имена переменных
Адреса ячеек
Ячейки памяти

Особенности переменных в Python:

- Не нужно заранее объявлять тип переменной — Python определяет тип автоматически.
- Имя переменной должно начинаться с буквы или символа _, и может содержать буквы, цифры и _.
- Имена чувствительны к регистру: **x** и **X** — разные переменные.
- Нельзя использовать ключевые слова Python в качестве имени переменной (например, **if**, **for**, **def**).

Чтобы сохранить значение переменной, используется оператор присваивания «=».

Не путать с «равно»!



Множественное присвоение.

Python позволяет присваивать значения сразу нескольким переменным:

```
x, y, z = 1, 2, 3 # x = 1, y = 2, z = 3
a = b = c = 0  # все три переменные получат значение 0
```

Изменение значения переменной.

Можно использовать существующую переменную для вычислений и присвоения нового значения:

```
x = 10
y = 20
x = x + y   # теперь x = 30
x = x / y   # а теперь  1.5
x = 0       # а теперь 0
```

Правила именования переменных:

- Имя переменной может состоять только из цифр, латинских букв и знака
- подчеркивания.
- Имя переменной не может начинаться с цифр.

Рекомендации именования:

- Имя переменной должно описывать её суть.
- Лучше использовать snake_case (слова с маленькой буквы и разделять подчеркиванием).

Определите, какие имена правильные:

1. zarplata = 1000
2. age = 25
3. 5element = True
4. familia = 45
5. month = ’May’
6. qqq = ‘Spartak’
7. is_number = True

Комбинированные операторы присвоения.

Комбинированные (сокращённые) операторы — это операторы, которые одновременно выполняют арифметическое действие и присваивание результата переменной.

Они являются краткой записью операций вида:

x = x + 5

Записывают короче:

x += 5

То есть переменная изменяется сама на основании своего прежнего значения.

Сокращенная запись операций

```
a += b   # a = a + b
a -= b   # a = a - b
a *= b   # a = a * b
a /= b   # a = a / b
a //= b  # a = a // b
a %= b   # a = a % b
```

```
a += 1
```

увеличение на 1

4. Простые типы данных в Python.

Простые типы — это типы данных, которые хранят **одно значение**.

Они не являются коллекциями и называются ещё значимыми типами.

Числовые типы

int — целые числа.

Хранят целые значения, положительные или отрицательные.

```
a = 10

b = -5

c = 0
```

float — числа с плавающей точкой.

Хранят дробные числа.

```
x = 3.14
y = -0.5
z = 1.0
```

Логический тип — bool

Представляет два значения: True (Истина) или False(Ложь).

```
flag = True
is_ok = False
```

Строки — str

- Хранят текст.
- Можно использовать одинарные, двойные и тройные кавычки.
- У строк есть множество методов

```
name = "Alice"
text = 'Hello, Python!'
```

NoneType — None

Специальное значение «ничего», используется для обозначения отсутствия значения.

```
result = None
```

Проверка типа данных

Проверить тип данных можно с помощью функции **type()**

```
x = 10
print(type(x))  # <class 'int'>

y = 3.14
print(type(y))  # <class 'float'>

z = "Hello"
print(type(z))  # <class 'str'>
```

Преобразование типов данных: явное и неявное.

Неявное преобразование типов в Python

Неявное преобразование типов — это когда Python автоматически приводит данные к нужному типу без указания программиста.

Это происходит тогда, когда такой шаг логичен, безопасен и однозначен.

1. Неявное преобразование в числовых операциях.

Python автоматически расширяет типы в сторону более «сложного» или «широкого» типа:

int → float

Примеры:

```
a = 10      # int
b = 3.5     # float
c = a + b   # int + float → float
print(c)    # 13.5
```

2. Логический тип (bool) в арифметике.

Тип **bool** — это подтип **int**.

Поэтому:

True ведёт себя как **1**

False ведёт себя как **0**

И это тоже неявное преобразование.

Примеры:

```
print(True + 1)   # 2
print(False + 5)  # 5
print(True * 10)  # 10
```

3. Неявное преобразование в функции **print()**.

Функция **print()** принимает любой тип и сама преобразует его к строке для вывода.

Пример:

```
print(10)
print(3.14)
print(True)
```

Происходит автоматическое преобразование:

- 10 → "10"
- 3.14 → "3.14"
- True → "True "

То есть print() вызывает внутреннее преобразование в строку, но тип самой переменной не меняется.

Явное (ручное) преобразование типов.

Явное преобразование — это когда мы сами указываем Python, к какому типу нужно привести данные.

Для этого используются встроенные функции:

- **int()** — в целое число
- **float()** — в число с плавающей точкой
- **str()** — в строку
- **bool()** — в логическое значение

Примеры явного преобразования.

1. Строка → число

```
a = "123"
b = int(a)        # 123 (int)
c = float(a)      # 123.0 (float)
```

2. Число → строка

```
x = 10
s = str(x)       # "10"
```

3. Число → логическое значение

```
bool(0)    # False
bool(5)    # True
```

4. Строка → логическое значение

```
bool("")      # False
bool("text")  # True
```

Контрольные вопросы:

- Что такое оператор и что такое операнд?
- Чем унарный оператор отличается от бинарного?
- Как работает унарный минус?
- Почему выражение 2 + 3 * 4 даёт 14, а не 20?
- Какой оператор имеет самый высокий приоритет?
- В чём разница между int и float?
- Какие значения может принимать тип bool?
- Что произойдёт при выполнении выражения 1 + 2.0 и почему?
- Что делает функция int() при передаче строки?
- Чем целочисленное деление (//) отличается от обычного (/)?

Домашнее задание:

https://ru.hexlet.io/programs/python-basics-free

Материалы лекций:

https://github.com/ShViktor72/education2025

ПМ2_Прикладное_программирование

Обратная связь:

colledge20education23@gmail.com