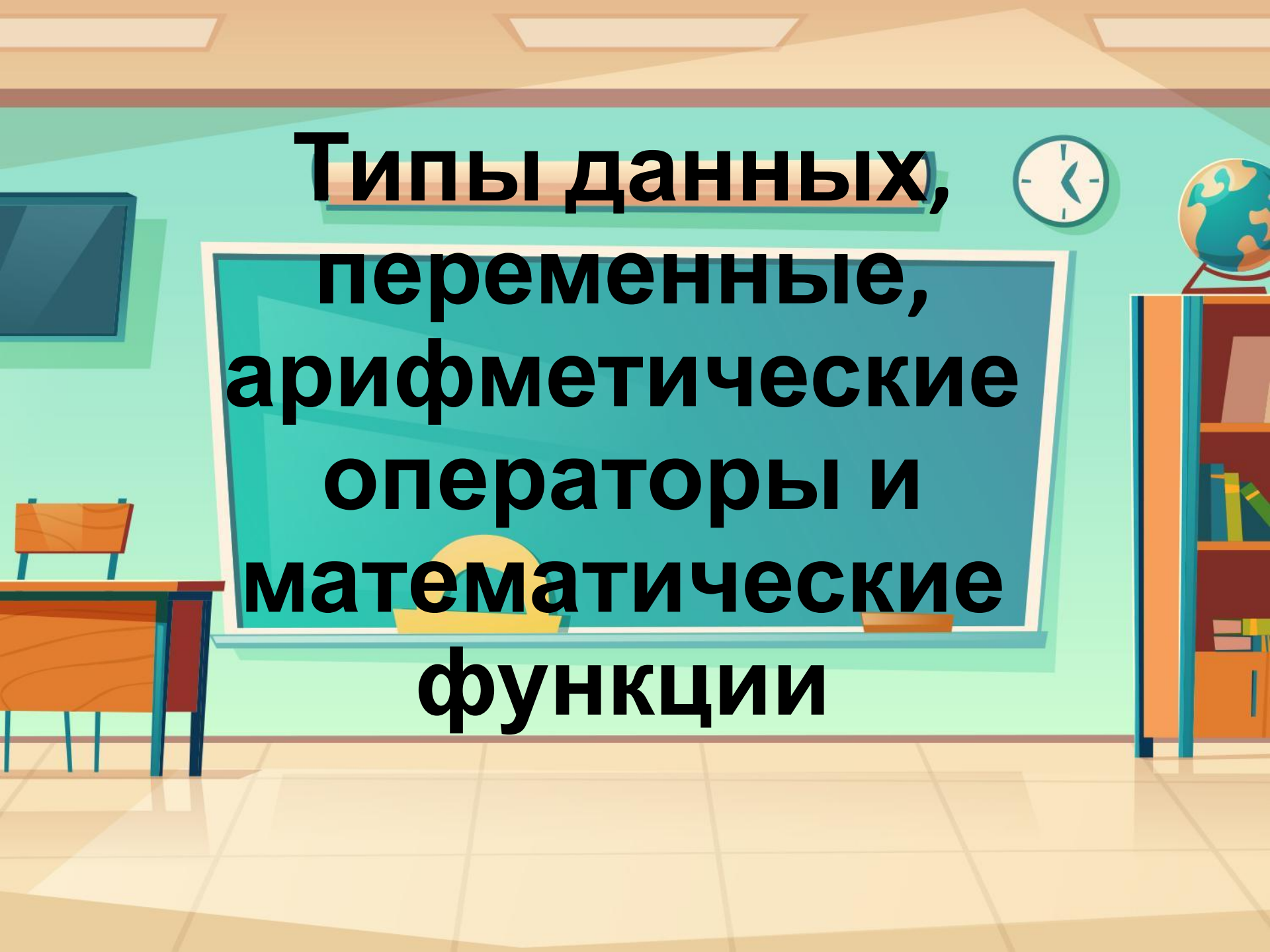
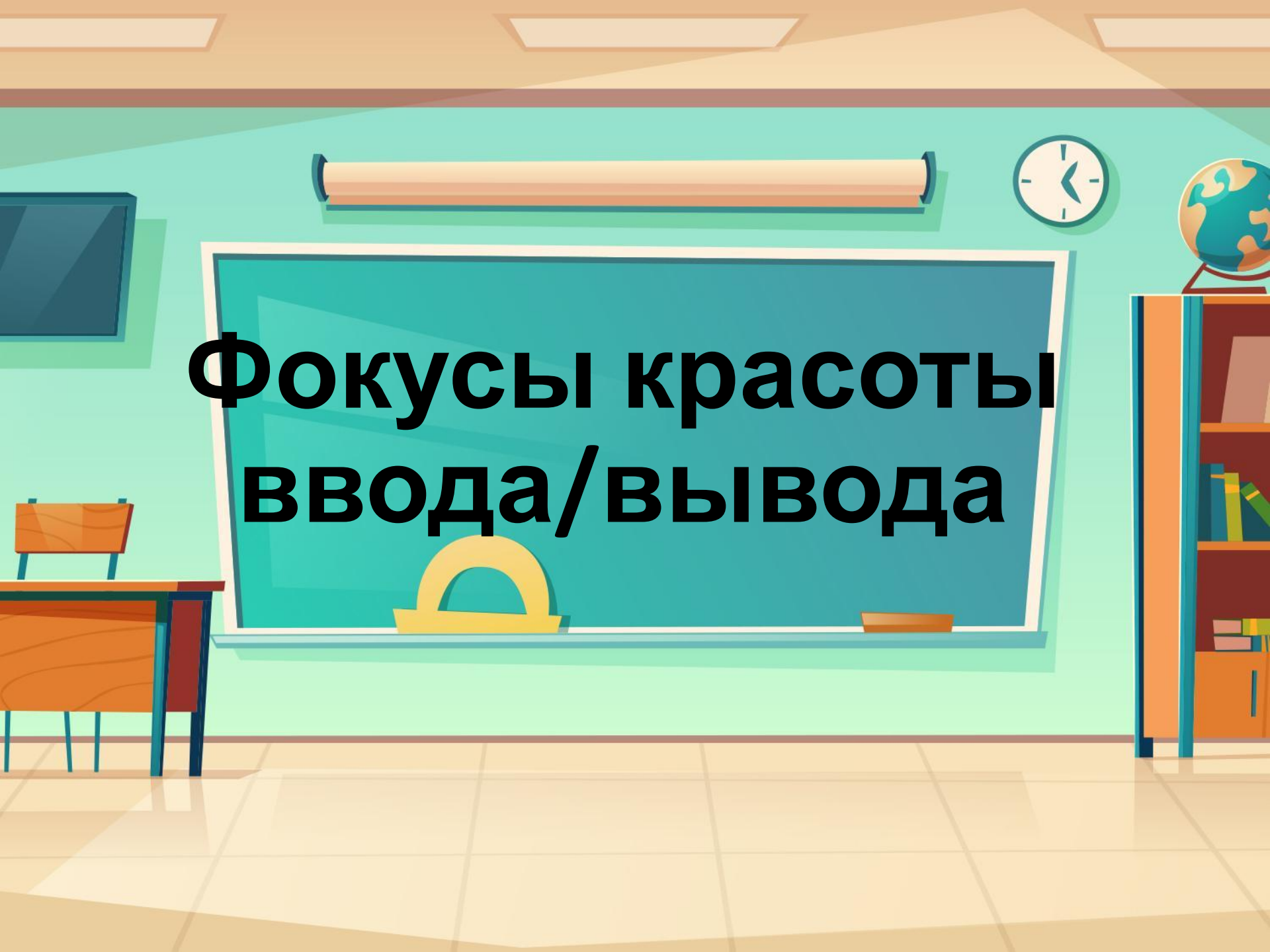

Типы данных, переменные, арифметические операторы и математические функции


Фокусы красоты
ввода/вывода

Секрет 1

Для перехода на новую строку используйте \n

Например:

print(“Привет! \n Как ты?”)

Результат:

Привет!

Как ты?

Для сравнения:

print(“Привет! Как ты?”)

Результат:

Привет! Как ты?

Секрет 2

Для определения строки можно заключать текст, как в двойные кавычки “Текст”, так и в одинарные ‘Текст’

Например:

print(“Привет! Как ты?”)

Результат:

Привет! Как ты?

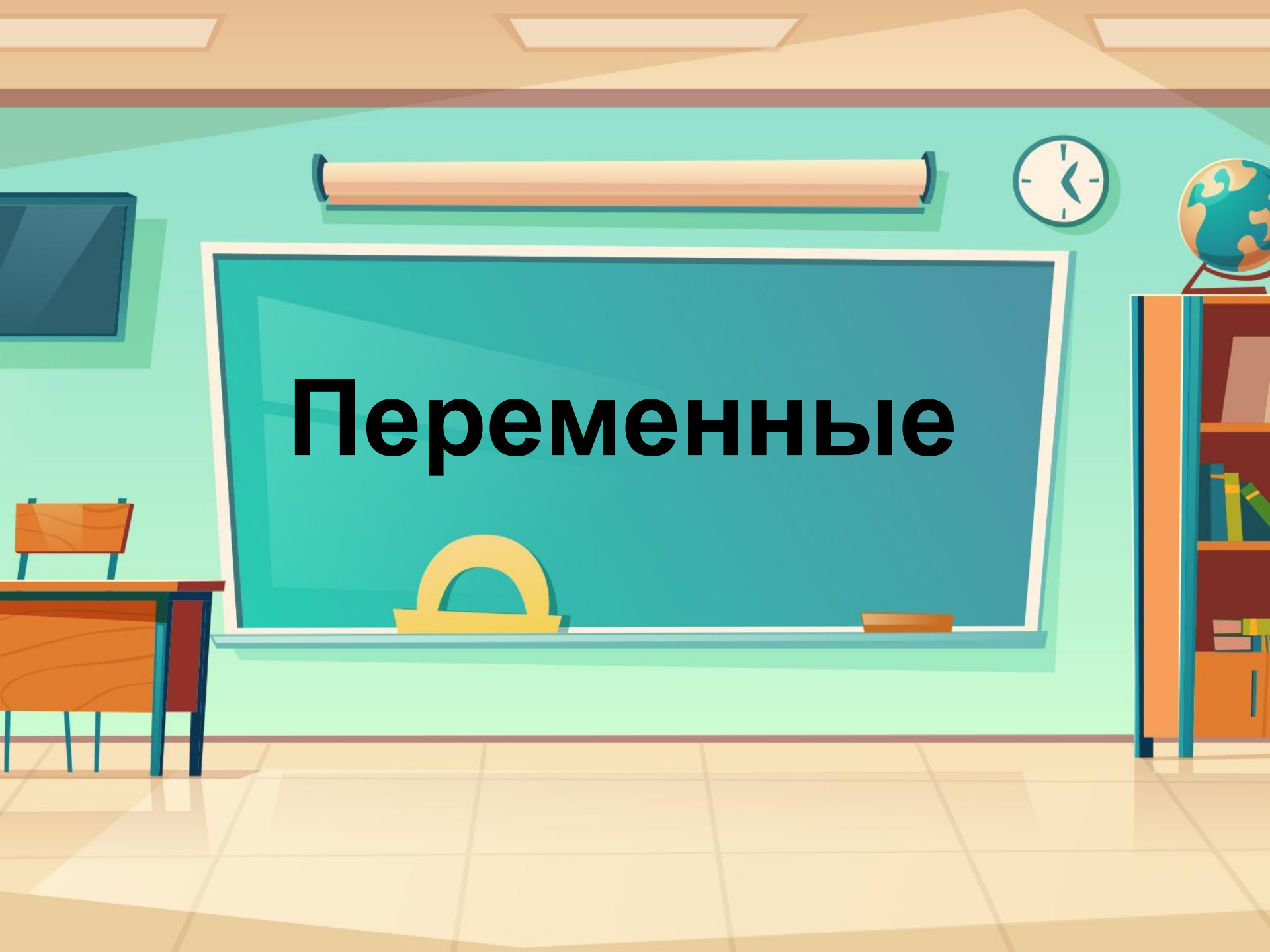
Для сравнения:

print(‘Привет! Как ты?’)

Результат:

Привет! Как ты?

Секрет 3
Функция input считывает только строку!
Например:
```
a=input()
```
Ввод:
6
Тип переменной a:
Строка


Переменные

Переменная - поименованная, либо адресуемая иным способом *область памяти*, адрес которой можно использовать для осуществления доступа к данным.

Данные, находящиеся в переменной (то есть по данному адресу памяти), **называются значением** этой переменной.

!!!!Странно, но значение переменной можно менять!!!!

Для объявления переменной необходимо задать ее имя и по необходимости тип данных. Для определения связи *между переменной и данными* используется знак присвоения «=».

Пример: sp=

Есть несколько правил именования переменных:

- **Желательно давать переменным осмысленные имена, говорящие о назначении данных, на которые они ссылаются.**

  Сложно придумать осмысленное имя, если ты сам не понимаешь в чем смысл.

- **Имя переменной не должно совпадать с командами языка (зарезервированными ключевыми словами).**

  Если нет фантазии, то не стоит пользоваться фантазией разработчиков языка!

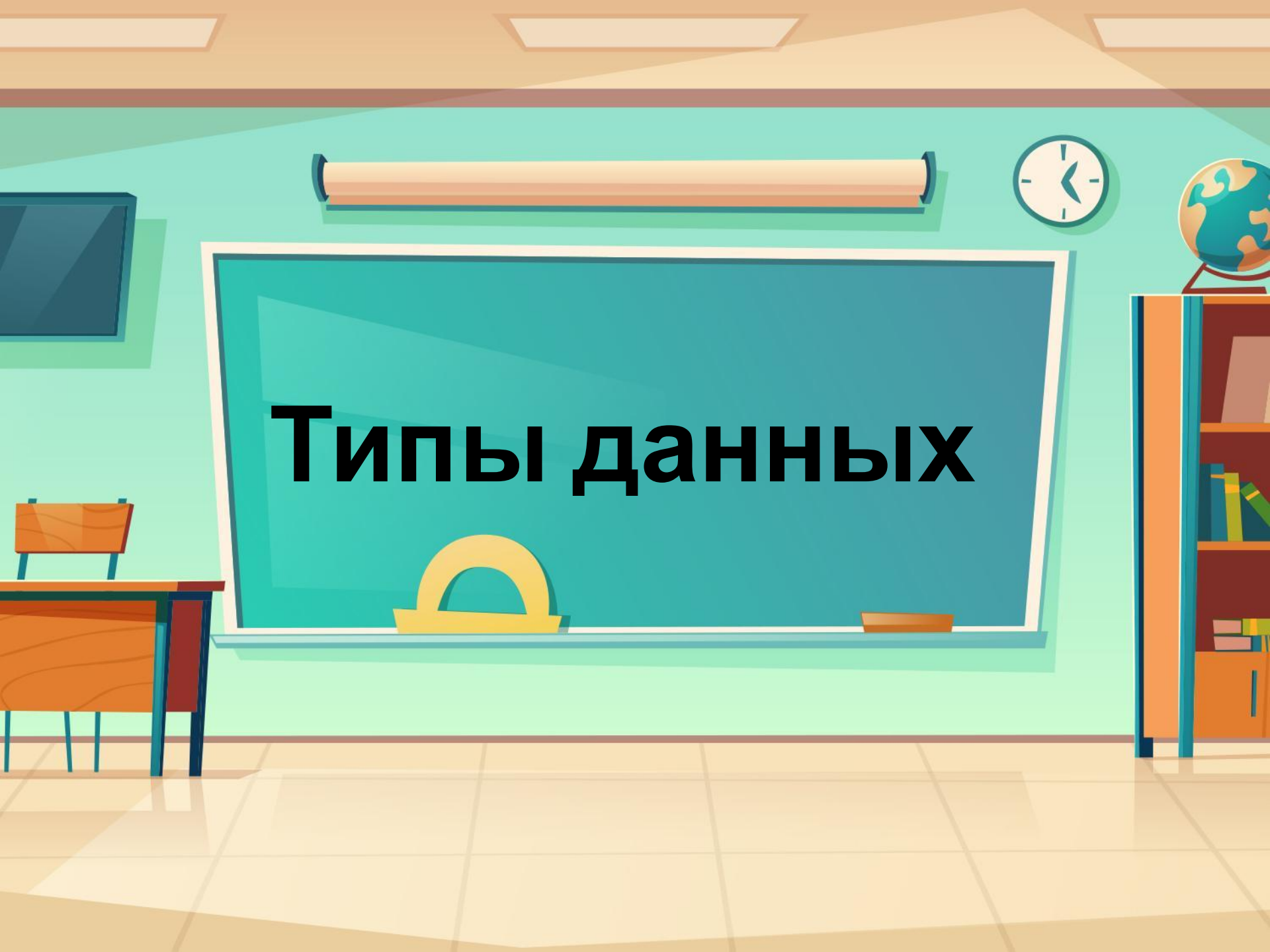
- **Имя переменной должно начинаться с буквы или символа подчеркивания (_), но не с цифры.**

  Цифры – это всё для математика, но называть переменные, начиная с цифры – это даже для них слишком.

- **Имя переменной не должно содержать пробелы.**

  Разделяя пробелом переменную, ты разрушаешь ее мечты на существование!

SR u Sr u sR u sr – это разные переменные!


Типы данных

В Python существует множество различных типов данных, которые подразделяются на категории: **числа, последовательности, словари, наборы**:

- **boolean** - логическое значение True или False
- **int** - представляет целое число, например, 1, 4, 8, 50.
- **float** - представляет число с плавающей точкой, например, 1.2 или 34.76
- complex - комплексные числа
- **str** - строки, например "hello". В Python 3.x строки представляют набор символов в кодировке Unicode
- bytes - последовательность чисел в диапазоне 0-255
- byte array - массив байтов, аналогичен bytes с тем отличием, что может изменяться
- list - список
- tuple - кортеж
- set - неупорядоченная коллекция уникальных объектов
- frozen set - то же самое, что и set, только не может изменяться (immutable)
- dict - словарь, где каждый элемент имеет ключ и значение

Если просто объявить переменную и присвоить ей значение, то тип переменной будет определен автоматически!

Например: per=4 #будет иметь целочисленный тип
per='word' #будет иметь тип –строка

Если хочется узнать тип, то нужно использовать функцию type()!

type(имя_переменной) или type(выражение)
type(aa) или type(54)

```
aa=1>2
print(type(aa)) #bool
aa=5
print(type(aa)) #int
aa=6.7
print(type(aa)) #float
aa="dd"
print(type(aa)) #str
aa=(-1)**(0.5)
print(type(aa)) #complex
```

Преобразование выражения к определенному типу!

Тип_к_которому_надо_преобразовать
(переменная_или_выражение)

Например:

```
aa=3.5
bb=int(aa)
print(bb)
```

Вывод:

```
3
```

Можно преобразовывать с помощью функций: int(), float(), bool(), str()


Арифметические операторы

Арифметические операторы

Сложение двух чисел:

- +

Пример: print(4+ 3) # 7

Вычитание двух чисел:

- -

Пример: print(22 - 2) # 20

Умножение двух чисел:

- *

Пример: print(4* 2) # 8

Деление двух чисел:

- /

Пример: print(6 /2) # 3

Арифметические операторы

Целочисленное деление (выводится только целая часть):

- //

Пример: print(11//3) # 3

Возведение в степень:

- **

Пример: print(2**3) # возводится 2 в 3-ю степень = 8

Остаток от деления:

- %

Пример: print(11% 3) # 2

Арифметические операторы с присвоением

Присвоение результата сложения:

- **+=**

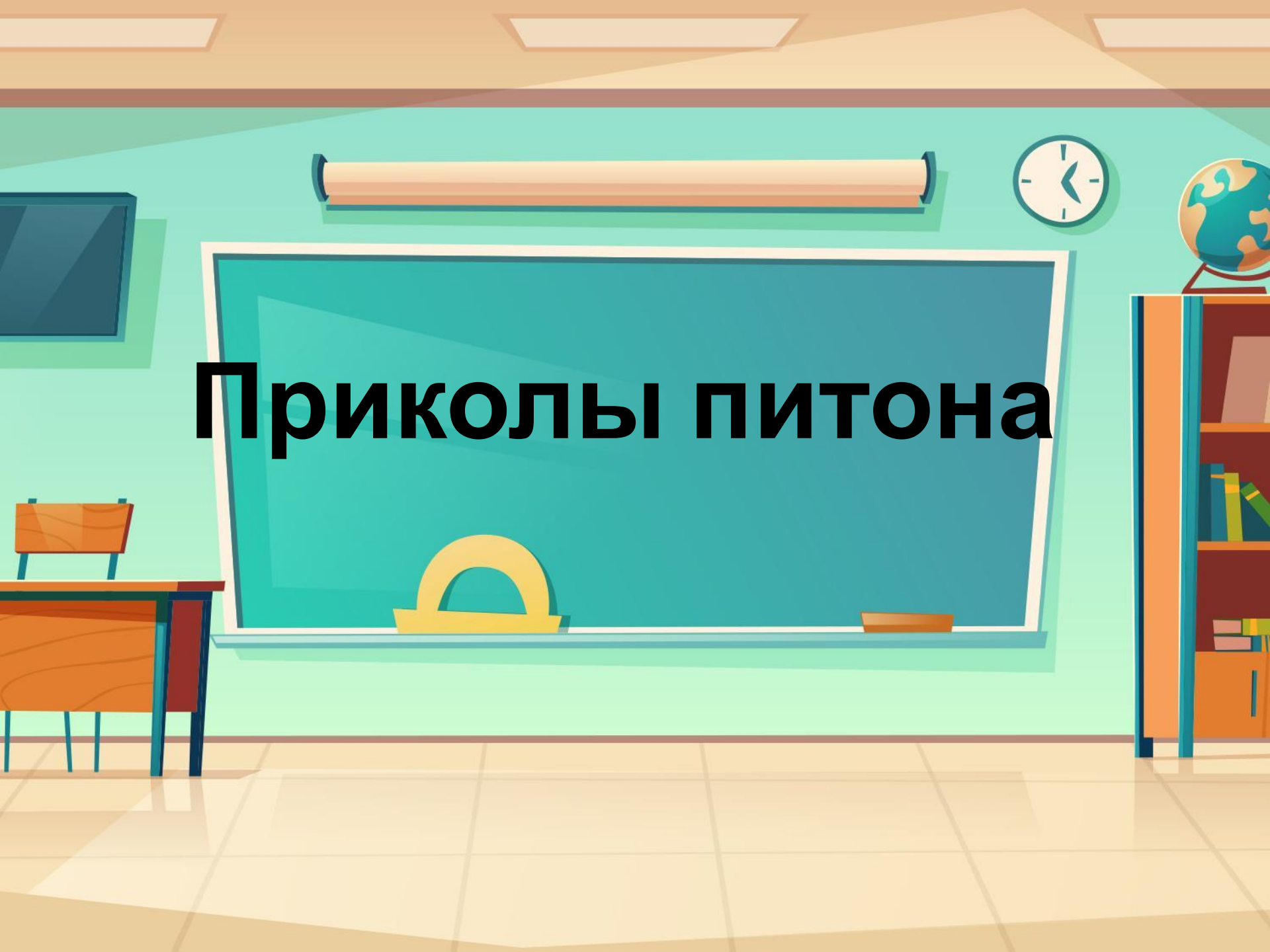
Пример:

```
aa=5
print(aa) #5
aa+=4
print(aa) #9
```

Арифметические операторы с присвоением

Еще кучка операторов с присвоением:

- **-= Присвоение результата вычитания**
- ***= Присвоение результата умножения**
- **/= Присвоение результата деления**
- **//= Присвоение результата целочисленного деления**
- ****= Присвоение результата возведения в степень**
- **%= Присвоение результата остатка от деления**


Приколы питона

Пример кода:

```
first_number = 2.0001
second_number = 5
third_number = first_number / second_number
print(third_number)
```

Ожидаемый результат:

0.40002

Фактический результат:

0.4000200000000004

Ну, чего ты? Просто используй round(значение_для_округления, кол-во_знаков_после_запятой)

Старый код	Новый код
first_number = 2.0001 second_number = 5 third_number = first_number / second_number print(third_number)	first_number = 2.0001 second_number = 5 third_number = first_number / second_number print(round(third_number,5))
Результат:	Результат:
0.40002000000000004	0.40002


Математические
функции

Жизнь такая штука, что всегда надо что-то в нее превносить, чтобы не умирать со скуки!

Для проведения вычислений с действительными числами язык Питон содержит много дополнительных функций, собранных в **библиотеку** (модуль), которая называется **math**.

Странно, но ее нужно подключить командой

import math

Да-да-да, просто перед применением любой функции из библиотеки напиши: import math, а лучше в самом начале кода)

Для использования любой функции из библиотеки нужно написать math.имя_функции (аргументы).

Мне книжку по матану надо программировать, там формул… Я ж на один ввод math потрачу годы

Просто пропиши from math import *
Вместо import math
Только это сссссссекрет

Функции библиотеки math (Округление)

round(x)	Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа.
round(x, n)	Округляет число x до n знаков после точки. Это стандартная функция, для ее использования не нужно подключать модуль math.
floor(x)	Округляет число вниз («пол»), при этом floor(1.5) == 1, floor(-1.5) == -2
ceil(x)	Округляет число вверх («потолок»), при этом ceil(1.5) == 2, ceil(-1.5) == -1
abs(x)	Модуль (абсолютная величина). Это — стандартная функция.

Функции библиотеки math (Корни, логарифмы)

sqrt(x)	Квадратный корень. Использование: sqrt(x)
log(x)	Натуральный логарифм. При вызове в виде log(x, b) возвращает логарифм по основанию b.
e	Основание натуральных логарифмов e = 2,71828...

Функции библиотеки math (Тригонометрия)

sin(x)	Синус угла, задаваемого в радианах
cos(x)	Косинус угла, задаваемого в радианах
tan(x)	Тангенс угла, задаваемого в радианах
asin(x)	Арксинус, возвращает значение в радианах
acos(x)	Арккосинус, возвращает значение в радианах
atan(x)	Арктангенс, возвращает значение в радианах
atan2(y, x)	Полярный угол (в радианах) точки с координатами (x, y).
degrees(x)	Преобразует угол, заданный в радианах, в градусы.
radians(x)	Преобразует угол, заданный в градусах, в радианы.
pi	Константа π = 3.1415...


Лабораторная работа

Просто посчитай это!
Не спрашивай зачем, так надо!

$$b = x + \frac{\sqrt[3]{zy}}{y + \cos x}$$