**Вопросы по Python**

**1. Типы данных, операторы, переменные, константы**

* **Типы данных:** Основные встроенные типы в Python:
  + **Числовые:** int (целые числа), float (числа с плавающей точкой), complex (комплексные числа).
  + **Текстовый:** str (строки, последовательности символов).
  + **Логический:** bool (булев тип, True или False).
  + **Последовательности:** list (изменяемый список), tuple (неизменяемый кортеж), range (диапазон чисел).
  + **Множества:** set (неупорядоченное множество уникальных элементов), frozenset (неизменяемое множество).
  + **Словари:** dict (неупорядоченная коллекция пар "ключ-значение").
  + **NoneType:** None (специальный тип, означающий отсутствие значения).
* **Операторы:**
  + **Арифметические:** +, -, \*, / (деление), // (целочисленное деление), % (остаток от деления), \*\* (возведение в степень).
  + **Сравнения:** == (равно), != (не равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно).
  + **Логические:** and (логическое И), or (логическое ИЛИ), not (логическое НЕ).
  + **Присваивания:** =, +=, -=, \*=, /=.
* **Переменные:** Именованная область в памяти для хранения данных. В Python переменные создаются в момент присваивания им значения. Тип переменной определяется автоматически (динамическая типизация).

Generated python

my\_variable = 10 # my\_variable теперь типа int

my\_variable = "Hello" # my\_variable теперь типа str

* **Константы:** В Python нет строгих констант, как в других языках. По соглашению (PEP 8), имена констант пишутся заглавными буквами, чтобы показать программистам, что это значение не следует изменять.

Generated python

PI = 3.14159

MAX\_CONNECTIONS = 10

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**2. Операторы ветвления, выбора, цикла**

* **Операторы ветвления и выбора (if, elif, else):** Используются для выполнения кода в зависимости от истинности условия.

Generated python

age = 18

if age < 18:

print("Доступ запрещен")

elif age == 18:

print("Вам 18, доступ разрешен")

else:

print("Добро пожаловать")

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Операторы цикла:**
  + **for:** Используется для перебора элементов в итерируемом объекте (список, строка, словарь и т.д.).

Generated python

for fruit in ["apple", "banana", "cherry"]:

print(fruit)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* + **while:** Выполняет блок кода, пока условие истинно.

Generated python

count = 0

while count < 5:

print(count)

count += 1

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**3. Массивы одномерные и многомерные**

В Python стандартным аналогом массива является **список (list)**. Для высокопроизводительных научных вычислений используется библиотека **NumPy**.

* **Одномерный массив (список):** Простая последовательность элементов.

Generated python

my\_list = [10, 20, 30, 40]

print(my\_list[1]) # Выведет 20

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Многомерный массив (список списков):** Список, элементами которого являются другие списки.

Generated python

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print(matrix[1][2]) # Выведет 6 (вторая строка, третий элемент)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**4. Строки. Набор методов для работы со строками**

**Строка (str)** — это неизменяемая последовательность символов Unicode.

* **Основные методы:**
  + s.upper(), s.lower(): Преобразование в верхний/нижний регистр.
  + s.strip(): Удаление пробельных символов в начале и в конце строки.
  + s.split(delimiter): Разбиение строки на список по разделителю.
  + delimiter.join(list): Сборка строки из списка с использованием разделителя.
  + s.replace(old, new): Замена подстроки old на new.
  + s.find(substring): Поиск подстроки, возвращает индекс первого вхождения или -1, если не найдено.
  + s.startswith(prefix), s.endswith(suffix): Проверка, начинается/заканчивается ли строка с указанного префикса/суффикса.
  + s.isdigit(), s.isalpha(), s.isalnum(): Проверка, состоит ли строка только из цифр, букв, или букв и цифр.

**5. Функции (методы)**

**Функция** — это именованный, многократно используемый блок кода, который выполняет определенную задачу. Определяется с помощью ключевого слова def.  
**Метод** — это функция, которая принадлежит объекту (классу) и вызывается через него (object.method()).

Generated python

# Определение функции

def greet(name):

return f"Hello, {name}!"

# Вызов функции

message = greet("Alice")

print(message) # Выведет "Hello, Alice!"

# Пример метода

my\_list = [1, 2, 3]

my\_list.append(4) # append - это метод объекта-списка

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**6. Файловый ввод-вывод. Работа с файлами**

Для работы с файлами используется функция open(). Рекомендуется использовать менеджер контекста with, который автоматически закрывает файл после завершения работы.

* **Чтение файла:**

Generated python

with open('myfile.txt', 'r', encoding='utf-8') as f:

content = f.read() # Читает весь файл в одну строку

# lines = f.readlines() # Читает все строки в список

# line = f.readline() # Читает одну строку

print(content)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Запись в файл:** (режим 'w' перезаписывает файл, 'a' — дописывает в конец)

Generated python

with open('output.txt', 'w', encoding='utf-8') as f:

f.write("Это первая строка.\n")

f.write("Это вторая строка.\n")

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**7. Класс как абстрактный тип. Создание и использование классов**

**Класс** — это чертеж или шаблон для создания объектов. Он определяет набор атрибутов (данных) и методов (функций), которые будут у его объектов. Класс является абстрактным типом данных, созданным программистом.

* **Создание класса:**

Generated python

class Dog:

# Конструктор, инициализирует объект

def \_\_init\_\_(self, name, breed):

self.name = name # Атрибут экземпляра

self.breed = breed # Атрибут экземпляра

# Метод экземпляра

def bark(self):

return f"{self.name} говорит: Гав!"

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**8. Использование Объектов и классов**

**Объект (экземпляр)** — это конкретная реализация класса. Он создается путем вызова класса как функции.

* **Использование:**

Generated python

# Создание объектов (экземпляров) класса Dog

my\_dog = Dog("Рекс", "Овчарка")

another\_dog = Dog("Шарик", "Дворняга")

# Доступ к атрибутам объекта

print(my\_dog.name) # Выведет "Рекс"

# Вызов методов объекта

print(another\_dog.bark()) # Выведет "Шарик говорит: Гав!"

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**9. Списки, кортежи, словари**

* **Списки (list):** Упорядоченная, **изменяемая** коллекция. Элементы могут быть разных типов. Создаются с помощью [].  
  my\_list = [1, "hello", 3.14]
* **Кортежи (tuple):** Упорядоченная, **неизменяемая** коллекция. После создания его нельзя изменить. Создаются с помощью (). Часто используются для данных, которые не должны меняться.  
  my\_tuple = (1, "hello", 3.14)
* **Словари (dict):** Неупорядоченная (в Python < 3.7) или упорядоченная (в Python >= 3.7) коллекция пар **ключ-значение**. **Изменяемая**. Ключи должны быть уникальными и неизменяемыми. Создаются с помощью {}.  
  my\_dict = {"name": "Иван", "age": 30}

**10. Компоненты для работы с текстом**

Это может относиться к строковым методам (см. вопрос 4) или к графическим компонентам (виджетам) для отображения и ввода текста в GUI-приложениях, например, в библиотеке **Tkinter**:

* **Label:** Отображает статичный текст.
* **Entry:** Однострочное поле для ввода текста.
* **Text:** Многострочное поле для ввода и редактирования текста.
* **Button:** Кнопка, на которой может быть текст.

**11. Модули. Определение и подключение модулей**

**Модуль** — это файл с расширением .py, содержащий определения функций, классов и переменных. Модули позволяют организовывать код и повторно его использовать.

* **Подключение модулей:**
  + import module\_name: Импортирует весь модуль. Доступ к содержимому через module\_name.function\_name.

Generated python

import math

print(math.pi)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* + from module\_name import name1, name2: Импортирует только указанные имена из модуля.

Generated python

from math import pi, sqrt

print(pi)

print(sqrt(16))

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* + import module\_name as alias: Импортирует модуль под псевдонимом.

Generated python

import numpy as np

arr = np.array([1, 2, 3])

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**12. Как можно принять результат ввода на клавиатуре?**

С помощью встроенной функции input(). Она считывает строку с клавиатуры и возвращает ее.

Generated python

name = input("Введите ваше имя: ")

print(f"Привет, {name}!")

# Важно: input() всегда возвращает строку. Для получения числа нужна конвертация.

age\_str = input("Введите ваш возраст: ")

age\_int = int(age\_str)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**13. Как проверить, что все символы строки относятся к алфавитно-цифровым?**

Используя строковый метод isalnum(). Он возвращает True, если все символы в строке — буквы или цифры, и строка не пуста.

Generated python

print("Python3".isalnum()) # True

print("Python 3".isalnum()) # False (из-за пробела)

print("Python-3".isalnum()) # False (из-за дефиса)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**14. В чем разница между списком и кортежем?**

|  |  |  |
| --- | --- | --- |
| Характеристика | Список (list) | Кортеж (tuple) |
| **Изменяемость** | **Изменяемый** (Mutable) | **Неизменяемый** (Immutable) |
| **Синтаксис** | [1, 2, 3] | (1, 2, 3) |
| **Производительность** | Медленнее, потребляет больше памяти | Быстрее, потребляет меньше памяти |
| **Использование** | Когда коллекция должна меняться | Когда данные не должны меняться, ключ в словаре |

**15. Объясните функцию range**

Функция range() создает неизменяемую последовательность чисел. Она очень эффективна по памяти, так как не хранит все числа одновременно.

* **Синтаксис:** range(start, stop, step)
  + stop: Обязательный параметр. Последовательность идет до stop - 1.
  + start: Необязательный. Начальное число (по умолчанию 0).
  + step: Необязательный. Шаг (по умолчанию 1).

Generated python

for i in range(5): # от 0 до 4

print(i, end=' ') # 0 1 2 3 4

print()

for i in range(2, 6): # от 2 до 5

print(i, end=' ') # 2 3 4 5

print()

for i in range(0, 10, 2): # от 0 до 9 с шагом 2

print(i, end=' ') # 0 2 4 6 8

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**16. Что такое локальные и глобальные переменные в Python?**

* **Глобальные переменные:** Объявлены вне всех функций. Они доступны для чтения из любой части кода, включая функции.
* **Локальные переменные:** Объявлены внутри функции. Они существуют и доступны только внутри этой функции.

Generated python

global\_var = "Я глобальная"

def my\_func():

local\_var = "Я локальная"

print(local\_var)

print(global\_var) # Можно читать глобальную переменную

my\_func()

print(global\_var)

# print(local\_var) # Ошибка: NameError, локальная переменная недоступна снаружи

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

Чтобы изменить глобальную переменную из функции, нужно использовать ключевое слово global.

**17. В чем разница между func и func()?**

* **func:** Это ссылка на сам объект функции. Вы можете передать эту ссылку в другую функцию, присвоить переменной и т.д.
* **func():** Это **вызов** функции. Это выражение будет выполнено, и на его место будет подставлен результат, который вернула функция (с помощью return).

Generated python

def greet():

return "Hello!"

a = greet # a - это ссылка на функцию greet

b = greet() # b - это результат вызова функции, т.е. строка "Hello!"

print(a) # <function greet at 0x...>

print(b) # Hello!

print(a()) # Вызов функции через ссылку a -> Hello!

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**18. Объясните, как работает функция map**

Функция map(function, iterable) применяет указанную function к каждому элементу iterable (например, списка) и возвращает специальный map-объект (итератор), содержащий результаты.

Generated python

numbers = [1, 2, 3, 4]

# Увеличить каждое число на 10

new\_numbers\_iterator = map(lambda x: x + 10, numbers)

# Чтобы увидеть результат, нужно преобразовать итератор, например, в список

new\_numbers\_list = list(new\_numbers\_iterator)

print(new\_numbers\_list) # [11, 12, 13, 14]

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**19. Переменные в Python передаются по ссылке или по значению?**

В Python используется механизм, который называется **"передача по ссылке на объект"** (pass-by-object-reference) или **"передача по присваиванию"** (pass-by-assignment).

* **Для неизменяемых типов (int, str, tuple):** Эффект похож на передачу по значению. Вы не можете изменить исходный объект внутри функции.
* **Для изменяемых типов (list, dict):** Эффект похож на передачу по ссылке. Если вы изменяете сам объект внутри функции (например, добавляете элемент в список), эти изменения отразятся на исходном объекте.

Generated python

def modify\_list(my\_list):

my\_list.append(100) # Изменяем сам объект

def try\_to\_modify\_number(num):

num = 0 # Создаем новую локальную переменную num

data\_list = [1, 2]

modify\_list(data\_list)

print(data\_list) # [1, 2, 100] -> список изменился

data\_num = 5

try\_to\_modify\_number(data\_num)

print(data\_num) # 5 -> число не изменилось

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**20. Как развернуть список?**

Есть три основных способа:

1. **Метод .reverse():** Разворачивает список "на месте" (модифицирует исходный список).

Generated python

my\_list = [1, 2, 3]

my\_list.reverse()

print(my\_list) # [3, 2, 1]

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

1. **Функция reversed():** Возвращает итератор, который перебирает элементы в обратном порядке. Не изменяет исходный список.

Generated python

my\_list = [1, 2, 3]

reversed\_list = list(reversed(my\_list))

print(reversed\_list) # [3, 2, 1]

print(my\_list) # [1, 2, 3]

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

1. **Срез (slicing):** Создает новую копию списка в обратном порядке. Самый "питонический" способ.

Generated python

my\_list = [1, 2, 3]

reversed\_list = my\_list[::-1]

print(reversed\_list) # [3, 2, 1]

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**21. Что означает self в классе?**

self — это общепринятое имя для первого параметра в методах класса. Он представляет **экземпляр (объект)**, для которого вызывается метод. Через self метод получает доступ к атрибутам и другим методам этого конкретного объекта.

Generated python

class Car:

def \_\_init\_\_(self, color):

self.color = color # self.color - это атрибут конкретного объекта

def show\_color(self):

# self здесь ссылается на тот объект, у которого вызвали метод

print(f"Мой цвет: {self.color}")

red\_car = Car("красный")

blue\_car = Car("синий")

red\_car.show\_color() # self внутри метода будет red\_car

blue\_car.show\_color() # self внутри метода будет blue\_car

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**22. Как объединить списки в Python?**

1. **Оператор +:** Создает новый список, содержащий элементы обоих списков.

Generated python

list1 = [1, 2]

list2 = [3, 4]

combined\_list = list1 + list2

print(combined\_list) # [1, 2, 3, 4]

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

1. **Метод .extend():** Добавляет все элементы другого списка в конец исходного (модифицирует "на месте").

Generated python

list1 = [1, 2]

list2 = [3, 4]

list1.extend(list2)

print(list1) # [1, 2, 3, 4]

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**23. В чем разница между списками и массивами?**

В контексте стандартного Python:

* **Список (list):** Встроенный тип. Может хранить элементы разных типов данных. Более гибкий, но менее эффективный по памяти.
* **Массив (array):** Из модуля array. Хранит элементы **только одного типа** (например, только целые числа). Более эффективен по памяти, чем список, если хранить много однотипных элементов.

В контексте **NumPy**:

* **NumPy Array:** Гораздо мощнее и быстрее, чем списки и array. Основа для научных вычислений. Поддерживает многомерность и векторизованные операции.

**24. Назовите изменяемые и неизменяемые объекты**

* **Изменяемые (Mutable):** Их содержимое можно изменить после создания.
  + list (список)
  + dict (словарь)
  + set (множество)
* **Неизменяемые (Immutable):** Их содержимое нельзя изменить после создания.
  + int, float (числа)
  + str (строка)
  + tuple (кортеж)
  + bool (булев тип)
  + frozenset (неизменяемое множество)

**25. Как округлить число до трех десятичных знаков?**

С помощью встроенной функции round().

Generated python

number = 3.14159265

rounded\_number = round(number, 3)

print(rounded\_number) # 3.142

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

*Важно:* Для финансовых расчетов, где важна точность, следует использовать модуль Decimal.

**26. Где быстрее поиск: в словарях или списках?**

**В словарях поиск значительно быстрее.**

* **Словарь (dict):** Поиск по ключу выполняется за **O(1)** в среднем (константное время) благодаря хэшированию. Время поиска почти не зависит от размера словаря.
* **Список (list):** Поиск элемента (value in my\_list) требует перебора элементов один за другим. Это **O(n)** (линейное время), где n — количество элементов. Время поиска растет прямо пропорционально размеру списка.

**27. В чем разница между модулем и пакетом?**

* **Модуль:** Один файл \*.py.
* **Пакет:** Это способ структурирования модулей. Пакет — это директория, которая содержит:
  + Несколько файлов-модулей.
  + Специальный файл \_\_init\_\_.py (может быть пустым), который указывает Python, что эта директория является пакетом.

**28. Как увеличить и уменьшить целое число в Python?**

В Python нет операторов инкремента (++) и декремента (--). Используются операторы присваивания с операцией.

Generated python

x = 10

x += 1 # Увеличить на 1

print(x) # 11

y = 5

y -= 1 # Уменьшить на 1

print(y) # 4

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**29. Как проверить, существует ли значение в списке?**

Используя оператор in.

Generated python

my\_list = [10, 20, "hello", 30]

if 20 in my\_list:

print("20 найдено в списке.")

if "world" not in my\_list:

print("'world' не найдено в списке.")

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**30. В чем разница между append и extend?**

Оба метода изменяют список "на месте".

* **append(element):** Добавляет element в конец списка как **один** элемент.
* **extend(iterable):** Добавляет **каждый** элемент из iterable в конец списка.

Generated python

list\_a = [1, 2]

list\_b = [3, 4]

# Используем append

list\_a.append(list\_b)

print(list\_a) # [1, 2, [3, 4]] -> вложенный список

#---------------------

list\_c = [1, 2]

list\_d = [3, 4]

# Используем extend

list\_c.extend(list\_d)

print(list\_c) # [1, 2, 3, 4] -> элементы добавлены по одному

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**31. В чем разница между pass, continue и break?**

Все они используются внутри циклов (а pass и в других местах).

* **break:** **Полностью прерывает** выполнение цикла и передает управление коду, идущему после цикла.
* **continue:** **Пропускает** оставшуюся часть текущей итерации и немедленно переходит к **следующей итерации** цикла.
* **pass:** Ничего не делает. Это оператор-заглушка. Он используется там, где синтаксически требуется инструкция, но вам не нужно выполнять никаких действий (например, в пустых функциях, классах или блоках if/except).

**32. Какие типы данных поддерживаются в питоне? (5)**

1. **int** (целое число)
2. **float** (число с плавающей точкой)
3. **str** (строка)
4. **list** (список)
5. **bool** (булев тип)

(Также tuple, dict, set, NoneType и др.)

**33. Как выполняется обработка исключений в Python?**

С помощью конструкции try...except.

* **try:** В этот блок помещается код, который может вызвать ошибку (исключение).
* **except ExceptionType:** Если в блоке try возникает исключение указанного типа, выполняется код в этом блоке.
* **else:** (необязательно) Выполняется, если в блоке try **не произошло** исключений.
* **finally:** (необязательно) Выполняется **всегда**, независимо от того, было исключение или нет. Используется для освобождения ресурсов (например, закрытия файла).

Generated python

try:

x = int(input("Введите число: "))

result = 10 / x

except ValueError:

print("Ошибка: введено не число!")

except ZeroDivisionError:

print("Ошибка: деление на ноль!")

else:

print(f"Результат: {result}")

finally:

print("Выполнение завершено.")

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**34. Tkinter. Создание окна приложения. Кнопки. Позиционирование**

**Tkinter** — стандартная библиотека Python для создания GUI.

* **Создание окна:**

Generated python

import tkinter as tk

window = tk.Tk() # Создаем главное окно

window.title("Мое приложение") # Устанавливаем заголовок

window.geometry("300x200") # Устанавливаем размер

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Создание кнопки:**

Generated python

def on\_button\_click():

print("Кнопка нажата!")

button = tk.Button(window, text="Нажми меня!", command=on\_button\_click)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Позиционирование (менеджеры геометрии):**
  + **.pack():** Упаковывает виджеты друг за другом (сверху вниз или слева направо). Просто и быстро.  
    button.pack()
  + **.grid(row=, column=):** Размещает виджеты в виде таблицы. Мощный и гибкий.  
    button.grid(row=0, column=0)
  + **.place(x=, y=):** Размещает виджеты по точным координатам. Используется редко.  
    button.place(x=50, y=50)

Generated python

# Запуск главного цикла приложения

window.mainloop()

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**35. Tkinter. Текстовая метка Label, Поле ввода Entry и другие элементы**

* **Label (Метка):** Отображает текст или изображение.

Generated python

label = tk.Label(window, text="Это статичная метка")

label.pack()

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Entry (Поле ввода):** Однострочное поле для ввода текста пользователем.

Generated python

entry = tk.Entry(window, width=30)

entry.pack()

# Получить текст из поля: entry.get()

# Вставить текст в поле: entry.insert(0, "Начальный текст")

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Другие элементы:**
  + Text: Многострочное поле ввода.
  + Checkbutton: Галочка (флажок).
  + Radiobutton: Переключатель (выбор одного из нескольких).
  + Frame: Рамка для группировки других виджетов.

**36. Работа с базой SQLite**

Python имеет встроенный модуль sqlite3 для работы с базой данных SQLite.

**Основные шаги:**

1. **Подключиться** к базе данных (sqlite3.connect()).
2. Создать объект **курсора** (connection.cursor()).
3. **Выполнить** SQL-запрос с помощью курсора (cursor.execute()).
4. **Получить** результаты (cursor.fetchone(), cursor.fetchall()).
5. **Сохранить изменения** (connection.commit()).
6. **Закрыть** соединение (connection.close()).

Generated python

import sqlite3

# 1. Подключение (файл будет создан, если не существует)

conn = sqlite3.connect('mydatabase.db')

# 2. Создание курсора

cursor = conn.cursor()

# 3. Выполнение SQL

cursor.execute('''

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY,

name TEXT NOT NULL,

age INTEGER

)

''')

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Alice", 30))

cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Bob", 25))

# 5. Сохранение изменений

conn.commit()

# 3, 4. Выборка данных

cursor.execute("SELECT \* FROM users WHERE age > ?", (28,))

user = cursor.fetchone() # Получить одну строку

print(user) # (1, 'Alice', 30)

# 6. Закрытие соединения

conn.close()

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**37. Python полностью поддерживает ООП?**

**Да**, Python является объектно-ориентированным языком программирования. Он поддерживает все ключевые принципы ООП:

* **Инкапсуляция:** Группировка данных (атрибутов) и методов для работы с ними в одном объекте (классе).
* **Наследование:** Возможность создавать новый класс на основе существующего, перенимая его свойства и методы.
* **Полиморфизм:** Возможность использовать объекты разных классов с одинаковым интерфейсом.
* **Абстракция:** Сокрытие сложной реализации и предоставление простого интерфейса.

Более того, в Python **"всё является объектом"**, включая числа, строки и даже функции.

**38. Что может быть ключом в словаре?**

Ключом в словаре может быть любой **неизменяемый (immutable)** и **хэшируемый (hashable)** тип данных.

* **Можно использовать:** int, float, str, tuple, bool.
* **Нельзя использовать:** list, dict, set (так как они изменяемые).

**39. Для чего используется функция \_\_init\_\_?**

\_\_init\_\_ — это специальный метод в классе, который называется **конструктором**. Он автоматически вызывается в момент создания нового объекта (экземпляра) класса. Его основная задача — **инициализировать** атрибуты объекта, то есть задать им начальные значения.

Generated python

class Person:

def \_\_init\_\_(self, name, age): # Конструктор

print("Создается новый объект Person")

self.name = name # Инициализация атрибута name

self.age = age # Инициализация атрибута age

# При вызове Person(...) автоматически вызывается \_\_init\_\_

p1 = Person("Иван", 25)

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**40. Что такое синхронный код и асинхронный код?**

* **Синхронный код (Synchronous):** Задачи выполняются **строго последовательно**, одна за другой. Программа "блокируется" и ждет завершения текущей задачи, прежде чем перейти к следующей. Это стандартный способ выполнения кода.  
  *Пример: Сначала скачать файл (программа ждет), потом обработать его.*
* **Асинхронный код (Asynchronous):** Позволяет программе начать выполнение долгой задачи (например, сетевой запрос или чтение большого файла) и, **не дожидаясь ее завершения**, переключиться на выполнение других задач. Когда долгая задача будет завершена, программа вернется к обработке ее результата. Это повышает производительность в задачах, связанных с вводом-выводом (I/O-bound tasks). В Python для этого используются async и await.  
  *Пример: Начать скачивание файла А, начать скачивание файла B, и пока они скачиваются, выполнять какие-то вычисления.*

**41. Что такое генератор и чем отличается от итератора?**

* **Итератор:** Это объект, который реализует протокол итерации, то есть имеет метод \_\_next\_\_(), возвращающий следующий элемент последовательности, и метод \_\_iter\_\_(), возвращающий сам себя. Когда элементы заканчиваются, \_\_next\_\_() вызывает исключение StopIteration.
* **Генератор:** Это **простой способ создания итератора**. Генератор — это функция, которая использует ключевое слово yield для "ленивой" выдачи значений по одному. При каждом вызове yield выполнение функции приостанавливается, а ее состояние сохраняется.

**Ключевое отличие:**

* Любой генератор является итератором.
* Но не любой итератор является генератором (итератор можно создать и как класс с методами \_\_iter\_\_ и \_\_next\_\_).
* Генераторы пишутся проще и компактнее, чем классы-итераторы.

Generated python

# Функция-генератор

def simple\_generator(n):

for i in range(n):

yield i # Приостанавливает выполнение и возвращает значение

# Использование генератора

gen = simple\_generator(3)

print(next(gen)) # 0

print(next(gen)) # 1

print(next(gen)) # 2

# print(next(gen)) # вызовет StopIteration

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

**Вопросы по Java**

**42. Назначения языка Java. Средства разработки приложений Java.**

* **Назначение:** Java — это строго типизированный, объектно-ориентированный язык программирования общего назначения. Его главная особенность — принцип **"Write Once, Run Anywhere" (WORA)**, что означает, что скомпилированный Java-код может выполняться на любой платформе, где установлена **виртуальная машина Java (JVM)**.
  + **Основные области применения:**
    - Крупные корпоративные (Enterprise) системы.
    - Backend для веб-приложений (Spring, Jakarta EE).
    - Разработка под Android.
    - Системы для обработки больших данных (Big Data).
    - Научные и финансовые приложения.
* **Средства разработки:**
  + **JDK (Java Development Kit):** Набор инструментов, включающий компилятор (javac), JVM, стандартные библиотеки и утилиты для разработки.
  + **IDE (Интегрированные среды разработки):**
    - **IntelliJ IDEA** (считается одной из самых мощных).
    - **Eclipse**.
    - **NetBeans**.
  + **Системы сборки:** Maven, Gradle.

**43. Типы данных и операции над ними (язык Java)**

В Java есть две категории типов данных:

1. **Примитивные типы (Primitive Types):**
   * **Целочисленные:** byte (8 бит), short (16), int (32), long (64).
   * **С плавающей точкой:** float (32), double (64).
   * **Символьный:** char (16-битный Unicode).
   * **Логический:** boolean (true или false).
2. **Ссылочные типы (Reference Types):**
   * Ссылаются на объекты в памяти.
   * **Классы** (например, String, Integer, любой пользовательский класс).
   * **Массивы** (например, int[]).
   * **Интерфейсы**.

* **Операции:** Аналогичны операциям в Python/C++: арифметические (+, -, \*, /, %), сравнения (==, !=, >, <), логические (&&, ||, !), побитовые, присваивания.

**44. Объектно-ориентированное программирование (Java)**

Java — это классический ООП-язык. Он построен на четырех основных принципах:

1. **Инкапсуляция:** Сокрытие внутреннего состояния объекта и предоставление доступа к нему через публичные методы (геттеры и сеттеры). Данные и методы, работающие с ними, объединены в класс.
2. **Наследование:** Механизм, позволяющий одному классу (потомку) наследовать поля и методы другого класса (предка). Используется ключевое слово extends.
3. **Полиморфизм:** Способность объекта использовать методы родительского класса или интерфейса в своей собственной реализации. Позволяет работать с объектами разных классов через общий интерфейс. Проявляется через переопределение методов (@Override).
4. **Абстракция:** Выделение наиболее значимых характеристик объекта и игнорирование несущественных. В Java реализуется через абстрактные классы и интерфейсы.

**45. Работа со строками (Java)**

В Java строки представлены классом String.

* **Ключевая особенность:** Объекты класса String являются **неизменяемыми (immutable)**. Любая операция, которая "изменяет" строку, на самом деле создает новый объект String.
* **Основные методы:**
  + length(): возвращает длину строки.
  + charAt(int index): возвращает символ по индексу.
  + equals(Object another): сравнивает строки по содержимому (в отличие от ==, который сравнивает ссылки).
  + substring(int beginIndex, int endIndex): возвращает подстроку.
  + toUpperCase(), toLowerCase(): преобразование регистра.
  + concat(String str): объединяет строки (чаще используется оператор +).
  + split(String regex): разбивает строку по регулярному выражению.
* Для эффективного построения изменяемых строк используются классы **StringBuilder** (быстрый, но не потокобезопасный) и **StringBuffer** (потокобезопасный, но медленнее).

**46. Геттеры – сеттеры в Java**

**Геттеры (getters)** и **сеттеры (setters)** — это методы, используемые для реализации принципа **инкапсуляции**. Они предоставляют контролируемый доступ к приватным полям класса.

* **Поля класса** обычно объявляются с модификатором private, чтобы к ним нельзя было обратиться напрямую извне.
* **Геттер (get...)** — это публичный метод, который **возвращает** значение приватного поля.
* **Сеттер (set...)** — это публичный метод, который **устанавливает** новое значение для приватного поля. В сеттере можно добавить логику проверки (валидации) нового значения.

Generated java

public class Person {

private String name; // Приватное поле

private int age;

// Геттер для поля name

public String getName() {

return this.name;

}

// Сеттер для поля name

public void setName(String name) {

this.name = name;

}

// Геттер для поля age

public int getAge() {

return this.age;

}

// Сеттер с валидацией

public void setAge(int age) {

if (age > 0) { // Проверка

this.age = age;

}

}

}

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Java

IGNORE\_WHEN\_COPYING\_END

**Вопросы по Android Studio**

**47. Что такое Activity? Жизненный цикл Activity.**

**Activity** — это один из основных компонентов Android-приложения, представляющий собой **один экран с пользовательским интерфейсом**. Приложение обычно состоит из нескольких Activity, связанных между собой.

**Жизненный цикл Activity** — это последовательность состояний, через которые проходит Activity от создания до уничтожения. Управляется системой Android через вызов колбэк-методов:

1. onCreate(): Вызывается при создании Activity. Здесь происходит инициализация: создание UI, привязка данных.
2. onStart(): Activity становится видимой для пользователя.
3. onResume(): Activity выходит на передний план и готова к взаимодействию с пользователем.
4. onPause(): Activity теряет фокус (например, поверх нее появилось диалоговое окно), но все еще видна. Здесь следует остановить ресурсоемкие операции.
5. onStop(): Activity полностью перекрыта другой Activity и не видна пользователю.
6. onDestroy(): Activity уничтожается. Здесь происходит освобождение всех ресурсов.
7. onRestart(): Вызывается после onStop(), когда Activity снова становится видимой.

**48. Что такое AndroidManifest. Кратко объясните, что такое AndroidManifest и каковы его функции.**

**AndroidManifest.xml** — это главный конфигурационный файл Android-приложения. Он предоставляет системе Android всю необходимую информацию о приложении.

**Основные функции:**

* **Объявление компонентов:** Перечисляет все Activity, Service, BroadcastReceiver, ContentProvider приложения.
* **Указание точки входа:** Определяет, какая Activity будет запускаться при старте приложения.
* **Запрос разрешений:** Декларирует разрешения, необходимые приложению (например, доступ к интернету, камере, контактам).
* **Объявление аппаратных и программных требований:** Указывает, какие функции устройства нужны для работы приложения (например, Bluetooth, NFC).
* **Декларация библиотек:** Указывает используемые библиотеки.

**49. Что такое Intent? Кратко объясните, что такое Intent и каковы его функции.**

**Intent** — это объект-сообщение, который используется для запроса действия от другого компонента приложения (как своего, так и чужого). Это основной механизм для взаимодействия между Activity, Service и BroadcastReceiver.

**Типы Intent:**

1. **Явный (Explicit) Intent:** Указывает конкретный компонент, который должен быть запущен (например, new Intent(this, SecondActivity.class)). Используется для навигации внутри одного приложения.
2. **Неявный (Implicit) Intent:** Не указывает конкретный компонент, а описывает действие, которое нужно выполнить (например, "открыть веб-страницу", "отправить e-mail"). Система Android сама находит подходящее приложение для выполнения этого действия.

**Функции:**

* Запуск другой Activity.
* Запуск Service.
* Отправка широковещательных сообщений (Broadcast).
* Передача данных между компонентами.

**50. Диалоговые окна Android. Какие диалоговые окна могут использоваться в Андроид приложении?**

**Диалоговое окно** — это небольшое окно, которое появляется поверх текущей Activity для информирования пользователя или запроса какого-либо действия.

* **AlertDialog:** Самый распространенный тип. Может содержать заголовок, сообщение, до трех кнопок (положительная, отрицательная, нейтральная), а также собственный макет (custom layout) или список для выбора.
* **DatePickerDialog и TimePickerDialog:** Предоставляют стандартный интерфейс для выбора даты и времени.
* **ProgressDialog (устарел):** Показывал прогресс выполнения операции. Сейчас рекомендуется использовать ProgressBar прямо в макете.
* **DialogFragment:** Рекомендуемый способ создания диалоговых окон. Это фрагмент, который отображается как диалог. Он правильно обрабатывает события жизненного цикла, например, поворот экрана.

**51. Виды Layout. Ключевые отличия и свойства. Какие виды разметок экрана(layout) существуют в android.**

**Layout (контейнер разметки)** — это ViewGroup, который определяет, как будут располагаться дочерние элементы (View) на экране.

* **ConstraintLayout:** Самый мощный и гибкий. Позволяет создавать сложные интерфейсы с плоской иерархией. Элементы располагаются относительно друг друга или родительского контейнера с помощью "связей" (constraints). **Рекомендуется Google как основной.**
* **LinearLayout:** Располагает элементы в одну линию — либо **горизонтально**, либо **вертикально** (задается свойством android:orientation).
* **RelativeLayout:** Располагает элементы относительно друг друга (например, "справа от кнопки А", "ниже текстового поля Б") или относительно родительского контейнера.
* **FrameLayout:** Самый простой контейнер. Элементы располагаются в виде стопки, друг на друге. Обычно используется для отображения одного элемента или для перекрытия элементов.
* **CoordinatorLayout:** Расширенный FrameLayout, который позволяет координировать поведение дочерних View (например, для создания всплывающей кнопки над прокручиваемым списком).

**52. Исполняемые файлы на Android**

* **APK (Android Package Kit):** Это архивный файл, который содержит все необходимое для установки и запуска приложения: скомпилированный код (.dex файлы), ресурсы, сертификаты и AndroidManifest.xml. APK — это то, что пользователь устанавливает на свое устройство.
* **AAB (Android App Bundle):** Это формат публикации приложения в Google Play. AAB содержит весь код и ресурсы, но Google Play использует его для генерации оптимизированных APK для каждого конкретного устройства (с учетом архитектуры процессора, языка, разрешения экрана). Это уменьшает размер скачиваемого приложения.
* **DEX (Dalvik Executable):** Скомпилированный байт-код Java/Kotlin, который исполняется на виртуальной машине Android (ART или ранее Dalvik).

**53. Что такое View? Приведите примеры ViewGroup.**

* **View:** Это базовый строительный блок для всех компонентов пользовательского интерфейса. View — это прямоугольная область на экране, которая отвечает за отрисовку и обработку событий.
  + **Примеры View:** TextView (текст), Button (кнопка), ImageView (изображение), EditText (поле ввода), ProgressBar.
* **ViewGroup:** Это специальный вид View, который может содержать другие View (и ViewGroup). Его задача — определять, как располагаются дочерние элементы.
  + **Примеры ViewGroup:** ConstraintLayout, LinearLayout, RelativeLayout, FrameLayout, RecyclerView, ScrollView.

**54. Структура проекта в Android Studio**

Проект в Android Studio имеет модульную структуру. Основной модуль — app.

* **app/manifests/AndroidManifest.xml:** Конфигурационный файл приложения.
* **app/java/:** Исходный код на Java или Kotlin, разделенный по пакетам. Здесь же находятся тесты.
* **app/res/ (ресурсы):**
  + drawable/: Изображения, иконки, XML-фигуры.
  + layout/: XML-файлы с разметкой экранов (Activity, Fragment).
  + mipmap/: Иконки приложения для разных разрешений экрана.
  + values/:
    - strings.xml: Строковые ресурсы.
    - colors.xml: Цвета.
    - dimens.xml: Размеры (отступы, шрифты).
    - styles.xml и themes.xml: Стили и темы приложения.
  + menu/: XML-файлы для описания меню.
* **Gradle Scripts:** Файлы сборки Gradle.
  + build.gradle (Project): Настройки для всего проекта.
  + build.gradle (Module: app): Настройки для конкретного модуля app (здесь подключаются зависимости, указывается версия SDK).

**55. Что такое SDK Manager и AVD Manager.**

Оба являются инструментами, доступными в Android Studio.

* **SDK Manager (Software Development Kit Manager):**
  + Менеджер комплектов для разработки.
  + Позволяет **скачивать, устанавливать и управлять** различными версиями Android SDK, инструментами сборки (build-tools), платформами (platforms) и другими необходимыми библиотеками.
* **AVD Manager (Android Virtual Device Manager):**
  + Менеджер виртуальных устройств Android.
  + Позволяет **создавать, настраивать и запускать** эмуляторы Android (виртуальные устройства). Вы можете создать эмулятор для любой версии Android и с любыми характеристиками (размер экрана, объем памяти и т.д.) для тестирования вашего приложения.

**56. Создание простого меню**

1. **Создать XML-файл меню:**
   * В папке res создайте папку menu.
   * Внутри res/menu создайте XML-файл, например, main\_menu.xml.
   * Опишите пункты меню с помощью тегов <item>:

Generated xml

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:id="@+id/action\_settings"

android:title="Настройки"

android:icon="@drawable/ic\_settings"

app:showAsAction="ifRoom" />

<item

android:id="@+id/action\_about"

android:title="О программе"

app:showAsAction="never" />

</menu>

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Xml

IGNORE\_WHEN\_COPYING\_END

1. **"Надуть" (Inflate) меню в Activity:**
   * Переопределите метод onCreateOptionsMenu в вашей Activity:

Generated java

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main\_menu, menu);

return true;

}

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Java

IGNORE\_WHEN\_COPYING\_END

1. **Обработать нажатия на пункты меню:**
   * Переопределите метод onOptionsItemSelected:

Generated java

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

if (id == R.id.action\_settings) {

// Действие для "Настроек"

return true;

}

if (id == R.id.action\_about) {

// Действие для "О программе"

return true;

}

return super.onOptionsItemSelected(item);

}

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Java

IGNORE\_WHEN\_COPYING\_END

**57. Папка res/values. Использование ресурсов приложения**

Папка res/values/ содержит XML-файлы, которые определяют простые значения, такие как строки, цвета и размеры.

* strings.xml: Хранит все текстовые строки.
* colors.xml: Хранит цветовые коды.
* dimens.xml: Хранит размеры (например, 16dp для отступов, 18sp для шрифтов).
* styles.xml / themes.xml: Определяет стили и темы для виджетов и всего приложения.

**Использование ресурсов:**

* **В XML-разметке:** @string/app\_name, @color/colorPrimary, @dimen/margin\_normal.
* **В Java/Kotlin коде:** getResources().getString(R.string.app\_name), getResources().getColor(R.color.colorPrimary).

**Преимущества:**

* **Централизация:** Все значения в одном месте.
* **Локализация:** Легко переводить приложение, создавая папки values-ru, values-fr и т.д. с переведенными strings.xml.
* **Переиспользование:** Один и тот же ресурс можно использовать в разных частях приложения.

**58. Инструментальные среды разработки мобильных приложений для операционной системы Android**

1. **Android Studio:** **Официальная** и основная IDE для разработки под Android. Основана на IntelliJ IDEA, предоставляет полный набор инструментов: редактор кода, отладчик, эмуляторы, анализаторы производительности, инструменты для проектирования UI.
2. **IntelliJ IDEA:** С плагином для Android может использоваться для разработки, но Android Studio имеет более специализированный и полный набор функций.
3. **Кросс-платформенные фреймворки:**
   * **Flutter (с Dart):** Разработан Google. Позволяет создавать нативные приложения для Android и iOS из одной кодовой базы. Использует свою IDE (Android Studio с плагином Flutter) или VS Code.
   * **React Native (с JavaScript/TypeScript):** Разработан Facebook. Аналогично Flutter, позволяет создавать кросс-платформенные приложения.
   * **.NET MAUI (с C#):** Развитие Xamarin.Forms от Microsoft.
   * **Kotlin Multiplatform Mobile (KMM):** Позволяет писать общую бизнес-логику на Kotlin для Android и iOS, а UI оставлять нативным для каждой платформы.

**59. Адаптеры и списки в Android**

Для отображения динамических списков данных (например, список контактов, новостная лента) используется связка **контейнера-списка** и **адаптера**.

* **Контейнер списка:**
  + **RecyclerView:** Современный, гибкий и производительный компонент для отображения прокручиваемых списков. Является стандартом де-факто.
  + ListView (устарел): Более простая, но менее эффективная версия.
* **Адаптер (Adapter):** Это "мост" между данными и RecyclerView. Его задачи:
  + Принять источник данных (например, List<User>).
  + Создавать View (макеты) для каждого элемента списка.
  + Привязывать данные из конкретного объекта к конкретному View.

**Принцип работы RecyclerView с Adapter:**

1. RecyclerView запрашивает у Адаптера, сколько всего элементов в списке (getItemCount).
2. Для элементов, которые должны появиться на экране, RecyclerView просит Адаптер создать для них "держатель" (ViewHolder) и макет (onCreateViewHolder).
3. Затем RecyclerView просит Адаптер заполнить этот макет данными для конкретной позиции (onBindViewHolder).
4. Когда пользователь прокручивает список, RecyclerView переиспользует ставшие невидимыми View для отображения новых элементов, что экономит память и повышает производительность.

**60. Работа с графикой и изображениями (Андроид)**

* **Хранение:** Изображения (PNG, JPG, WebP) и XML-описания графики (Shape Drawable, Vector Drawable) хранятся в папке **res/drawable**. Иконки приложения — в **res/mipmap**.
* **Отображение:** Для показа изображений в макете используется виджет **ImageView**.

Generated xml

<ImageView

android:id="@+id/my\_image\_view"

android:layout\_width="100dp"

android:layout\_height="100dp"

android:src="@drawable/my\_image" />

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Xml

IGNORE\_WHEN\_COPYING\_END

* **Vector Drawable:** XML-формат для описания векторной графики. Преимущество — масштабируется без потери качества и имеет малый размер файла. Идеален для иконок.
* **Загрузка изображений из сети:** Для загрузки изображений из интернета не рекомендуется делать это вручную из-за сложности кеширования, обработки ошибок и работы с потоками. Используются специализированные библиотеки:
  + **Glide:** Мощная и популярная библиотека, проста в использовании, эффективно управляет памятью и кешем.
  + **Picasso:** Еще одна популярная и простая библиотека.
  + **Coil:** Современная библиотека, написанная на Kotlin и использующая корутины.