**Вопросы для собеседования на Python для новичков**

**1. Является ли Python компилируемым языком или интерпретируемым языком?**

Python можно считать как компилируемым, так и интерпретируемым языком, но на разных этапах процесса выполнения.

1. **Компиляция**: когда вы пишете код на Python и запускаете его, интерпретатор Python сначала компилирует ваш исходный код (.py-файлы) в промежуточную форму, называемую байт-кодом (.pyc-файлы). Этот байт-код является низкоуровневым представлением вашего кода, но он всё равно не является непосредственно машинным кодом. Это то, что виртуальная машина Python (PVM) может понять и выполнить.
2. **Интерпретация**: после компиляции кода Python в байт-код он выполняется виртуальной машиной Python (PVM), которая является интерпретатором. PVM считывает байт-код и выполняет его построчно во время выполнения, поэтому на практике Python считается интерпретируемым языком.

Некоторые реализации, такие как **PyPy**, используют компиляцию Just-In-Time (JIT), при которой код Python компилируется в машинный код во время выполнения для более быстрого выполнения, стирая границы между интерпретацией и компиляцией.

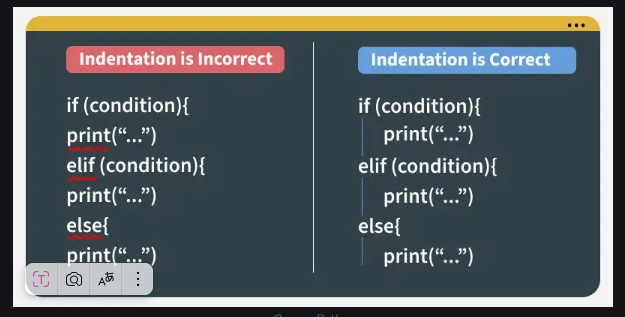
**2. Что такое динамически типизированный язык?**

[Типизированные языки](https://www.geeksforgeeks.org/what-is-a-typed-language/) — это языки, в которых мы определяем тип данных, и он будет известен машине во время компиляции или выполнения. Типизированные языки можно разделить на две категории:

* **Языки со статической типизацией:**в языках этого типа тип данных переменной известен во время компиляции, то есть программист должен указать тип данных переменной во время её объявления.
* **Языки с динамической типизацией:**это языки, которые не требуют определения типа данных для какой-либо переменной, поскольку он интерпретируется во время выполнения программы самой машиной. В этих языках интерпретаторы присваивают тип данных переменной во время выполнения программы в зависимости от её значения.

**3. Требуется ли в Python отступ?**

Да, [отступы](https://www.geeksforgeeks.org/indentation-in-python/) необходимы в Python. С помощью отступов в Python можно сообщить интерпретатору, что группа операторов относится к определённому блоку кода. Отступы облегчают чтение кода разработчиками на всех языках программирования, но в Python очень важно делать отступы в определённом порядке.



*Отступ Python*

**4. Что такое встроенные типы данных в Python?**

Ниже перечислены стандартные или встроенные[типы данных](https://www.geeksforgeeks.org/python-data-types/) в Python:

* **Числовой тип:**числовой тип данных в Python представляет данные, имеющие числовое значение. Числовое значение может быть целым, дробным, логическим**,**или даже комплексным числом.
* **Тип последовательности: T**ип последовательности в Python — это упорядоченная коллекция одинаковых или разных типов данных. В Python есть несколько типов последовательностей:
  + [Строка Python](https://www.geeksforgeeks.org/python-string/)
  + [Список Python](https://www.geeksforgeeks.org/python-lists/)
  + [Кортеж Python](https://www.geeksforgeeks.org/python-tuples/)
  + [Диапазон Python](https://www.geeksforgeeks.org/python-range-function/)
* **Типы сопоставления:**в Python хешируемые данные могут сопоставляться со случайными объектами с помощью объекта сопоставления. В настоящее время существует только один распространённый тип сопоставления — словарь, а объекты сопоставления являются изменяемыми.
  + [Словарь Python](https://www.geeksforgeeks.org/python-dictionary/)
* **Типы наборов:**в Python [набор](https://www.geeksforgeeks.org/sets-in-python/) — это неупорядоченная коллекция типов данных, которая является итерируемой, изменяемой и не содержит повторяющихся элементов. Порядок элементов в наборе не определён, хотя он может состоять из различных элементов.

**5. В чём разница между изменяемым типом данных и неизменяемым типом данных?**

* Изменяемые типы данных можно редактировать, то есть они могут изменяться во время выполнения программы. **Например,**список, словарь и т. д.
* Неизменяемые типы данных нельзя редактировать, то есть они не могут изменяться во время выполнения. **Например,** — строка, кортеж и т. д.

**6. Что такое область видимости переменной в Python?**

Область, в которой мы можем найти переменную, а также получить к ней доступ при необходимости, называется [областью видимости переменной](https://www.geeksforgeeks.org/python-scope-of-variables/).

* **Локальная переменная Python:** Локальные переменные — это переменные, которые инициализируются внутри функции и являются уникальными для этой функции. К ним нельзя получить доступ за пределами функции.
* **Глобальные переменные Python:** Глобальные переменные — это переменные, которые определяются и объявляются вне какой-либо функции и не указываются ни в одной функции.
* **Область видимости на уровне модуля:** относится к глобальным объектам текущего модуля, доступным в программе.
* **Внешняя область видимости:**относится к любым встроенным именам, которые может вызывать программа. Упомянутое имя находится в конце списка объектов в этой области видимости.

**7. Как записать число в Python?**

Чтобы округлить число в меньшую сторону в Python, можно использовать функцию math.floor(), которая возвращает наибольшее целое число, меньшее или равное заданному числу.

* [floor()](https://www.geeksforgeeks.org/floor-ceil-function-python/)метод в Python возвращает наименьшее целое число, не превышающее x.
* Кроме того, метод ceil(x) в Python возвращает наибольшее целое число, меньшее или равное x.

**import** **math**

n = 3.7

F\_num = math.floor(n)

print(F\_num)

**Выходной сигнал**

3

**8. В чем разница между / и // в Python?**

/ обозначает точное деление (результат — число с плавающей запятой), а // обозначает деление с округлением (результат — целое число). Например:

*5//2 = 2*

*5/2 = 2.5*

**9. Разница между циклом for и циклом while в Python**

Цикл «for» обычно используется для перебора элементов различных типов коллекций, таких как [список](https://www.geeksforgeeks.org/python-lists/), [кортеж](https://www.geeksforgeeks.org/python-tuples/), [множество](https://www.geeksforgeeks.org/sets-in-python/) и [словарь](https://www.geeksforgeeks.org/python-dictionary/). Разработчики используют цикл «for», в котором есть как начальное, так и конечное условия. В то время как цикл «while» — это функция циклического выполнения, которая используется в любом другом языке программирования. Программисты используют цикл «while» в Python, в котором есть только конечное условие.

**10. Можно ли передать функцию в качестве аргумента в Python?**

Да, в функцию можно передать несколько аргументов, включая объекты, переменные (одного или разных типов данных) и функции. [Функции можно передавать](https://www.geeksforgeeks.org/passing-function-as-an-argument-in-python/)в качестве параметров другим функциям, поскольку они являются объектами. Функции высшего порядка — это функции, которые могут принимать другие функции в качестве аргументов.

**11. Что такое pass в Python?**

[Оператор pass](https://www.geeksforgeeks.org/python-pass-statement/)в Python — это пустая операция или заполнитель. Он используется, когда оператор синтаксически необходим, но мы не хотим выполнять какой-либо код. Он ничего не делает, но позволяет сохранить структуру нашей программы.

Пример использования ключевого слова Pass в функции:

Ключевое слово pass в функции используется, когда мы определяем функцию, но не хотим сразу реализовывать её логику. Это позволяет функции быть синтаксически корректной, даже если она пока не выполняет никаких действий.

**def** fun():

**pass** *# Placeholder, no functionality yet*

*# Call the function*

fun()

**12. Что такое break, continue и pass в Python?**

* [Оператор break](https://www.geeksforgeeks.org/python-break-statement/) используется для завершения цикла или оператора, в котором он присутствует. После этого управление переходит к операторам, которые присутствуют после оператора break, если таковые имеются.
* [Continue](https://www.geeksforgeeks.org/python-continue-statement/) — это также оператор управления циклом, как и оператор break. Оператор continue противоположен оператору break: вместо завершения цикла он заставляет выполнить следующую итерацию цикла.
* [Pass](https://www.geeksforgeeks.org/python-pass-statement/)означает отсутствие выполнения какой-либо операции, или, другими словами, это заполнитель в составном операторе, где должен быть пробел, и ничего не нужно писать.

**13. Как в Python передаются аргументы: по значению или по ссылке?**

[Передача аргументов в Python](https://www.geeksforgeeks.org/pass-by-reference-vs-value-in-python/) модель **ни “Передавать по значению”, ни “Передавать по ссылке”, но это “Передавать по ссылке на объект”.**

В зависимости от типа объекта, который вы передаёте в функцию, функция будет вести себя по-разному. Неизменяемые объекты передаются «по значению», а изменяемые — «по ссылке».

Вы можете увидеть разницу между передачей по значению и передачей по ссылке в приведённом ниже примере:

**def** call\_by\_value(x):

x = x \* 2

print("in function value updated to", x)

**return**

**def** call\_by\_reference(list):

list.append("D")

print("in function list updated to", list)

**return**

my\_list = ["E"]

num = 6

print("number before=", num)

call\_by\_value(num)

print("after function num value=", num)

print("list before",my\_list)

call\_by\_reference(my\_list)

print("after function list is ",my\_list)

**Выходной сигнал**

number before= 6

in function value updated to 12

after function num value= 6

list before ['E']

in function list updated to ['E', 'D']

after function list is ['E', 'D']

**14. Что такое лямбда-функция?**

A[лямбда-функция](https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/) является анонимной функцией. Эта функция может иметь любое количество параметров, но может содержать только один оператор.

В этом примере мы определили лямбда-функцию(**upper**) для преобразования строки в верхний регистр с помощью функции upper().

s1 = 'GeeksforGeeks'

s2 = **lambda** func: func.upper()

print(s2(s1))

**Выходной сигнал**

GEEKSFORGEEKS

**15. Чем словарь отличается от списка?**

Список — это упорядоченная коллекция элементов, к которым можно получить доступ по индексу, а словарь — это неупорядоченная коллекция пар «ключ-значение», к которым можно получить доступ с помощью уникальных ключей. Списки идеально подходят для последовательных данных, а словари — для ассоциативных данных. Например, в списке можно хранить [10, 20, 30], а в словаре — {«a»: 10, «b»: 20, «c»: 30}.

**16. Что такое понимание списка? Приведите пример.**

[Понимание списков](https://www.geeksforgeeks.org/python-list-comprehension/) — это способ создания списков с использованием краткого синтаксиса. Он позволяет нам генерировать новый список, применяя **выражение**к каждому элементу в существующем **итерируемом**объекте (таком как **список** или **диапазон**). Это помогает нам писать более чистый и читаемый код по сравнению с традиционными методами циклирования.

**Например,** если у нас есть список целых чисел и мы хотим создать новый список, содержащий квадрат каждого элемента, мы можем легко сделать это с помощью генератора списков.

a = [2,3,4,5]

res = [val \*\* 2 **for** val **in** a]

print(res)

**Выходной сигнал**

[4, 9, 16, 25]

**17. Что такое \* args и \*\* kwargs?**

**\*args:**Специальный синтаксис *\*args* в определениях функцийиспользуется для передачи функции переменного количества аргументов. Программа на Python для иллюстрации \*args для переменного количества аргументов:

**def** myFun(\*argv):

**for** arg **in** argv:

print(arg)

myFun('Hello', 'Welcome', 'to', 'GeeksforGeeks')

**Выходной сигнал**

Hello

Welcome

to

GeeksforGeeks

**\*\*kwargs:**Специальный синтаксис \*\*kwargs в определениях функций используется для передачи списка аргументов переменной длины. Мы используем имя kwargs с двойной звездочкой \*\*.

**def** fun(\*\*kwargs):

**for** k, val **in** kwargs.items():

print("**%s** == **%s**" % (k, val))

*# Driver code*

fun(s1='Geeks', s2='for', s3='Geeks')

**Выходной сигнал**

s1 == Geeks

s2 == for

s3 == Geeks

**18. В чём разница между набором и словарем?**

Набор[Python Set](https://www.geeksforgeeks.org/sets-in-python/) — это неупорядоченный тип данных коллекции, который является итерируемым, изменяемым и не содержит повторяющихся элементов. Класс Python Set представляет математическое понятие множества.

**Синтаксис**: Определяется с помощью фигурных скобок {} или функции set().

*my\_set = {1, 2, 3}*

[Словарь](https://www.geeksforgeeks.org/python-dictionary/) в Python — это упорядоченный (начиная с Py 3.7) [неупорядоченный (начиная с Py 3.6 и ранее)] набор значений данных, используемый для хранения значений данных, как карта, которая, в отличие от других типов данных, содержащих только одно значение в качестве элемента, содержит **пару «ключ: значение»**. Пара «ключ: значение» используется в словаре для повышения его оптимизации.

**Синтаксис**: определяется с помощью фигурных скобок {} и пар «ключ-значение».

*my\_dict = {«a»: 1, «b»: 2, «c»: 3}*

**19. Как вы можете объединить два списка в Python?**

Мы можем объединить два списка в Python с помощью оператора + или метода**extend()**.

**1. Использование оператора +:**

Это создает новый список путем объединения двух списков вместе.

list1 = [1, 2, 3]

list2 = [4, 5, 6]

result = list1 + list2

print(result)

**Выходной сигнал**

[1, 2, 3, 4, 5, 6]

**2. Использование метода extend():**

Это позволяет добавить все элементы второго списка в первый на месте.

list1 = [1, 2, 3]

list2 = [4, 5, 6]

list1.extend(list2)

print(list1)

**Выходной сигнал**

[1, 2, 3, 4, 5, 6]

**20. Что такое docstring в Python?**

Строки документации Python (или [docstrings](https://www.geeksforgeeks.org/python-docstrings/)) предоставляют удобный способ связать документацию с модулями, функциями, классами и методами Python.

* **Объявление строк документации:** Строки документации объявляются с помощью «тройных одинарных кавычек» или «тройных двойных кавычек» непосредственно под объявлением класса, метода или функции. Все функции должны иметь строку документации.
* **Доступ к строкам документации:** Доступ к строкам документации можно получить с помощью метода \_\_doc\_\_ объекта или функции справки.

**21. Как в Python обрабатывается обработка исключений?**

Существует 3 основных ключевых слова: try, except и finally, которые используются для перехвата исключений и соответствующей обработки механизма восстановления. Try — это блок кода, который отслеживает ошибки. Блок except выполняется при возникновении ошибки.

Преимущество последнего блока заключается в том, что он выполняет код после попытки исправить ошибку. Этот блок выполняется независимо от того, произошла ошибка или нет. Наконец, блок используется для выполнения необходимых действий по очистке объектов/переменных.

**Пример:**попытка разделить число на ноль приведёт к исключению.

n = 10

**try**:

res = n / 0 *# This will raise a ZeroDivisionError*

**except** **ZeroDivisionError**:

print("Can't be divided by zero!")

**Выходной сигнал**

Can't be divided by zero!

**Объяснение:** В этом примере деление числа на 0 вызывает исключение [**ZeroDivisionError**](https://www.geeksforgeeks.org/zerodivisionerror-float-division-by-zero-in-python/). Блок try содержит код, который может вызвать исключение, а блок except обрабатывает исключение, выводя сообщение об ошибке вместо остановки программы.

1 / 3

**22. В чем разница между массивами Python и списками?**

[Массивы](https://www.geeksforgeeks.org/python-arrays/) (при работе с модулем array в Python) используются специально для хранения набора числовых элементов одного типа. Это делает их более эффективными для хранения больших объёмов данных и выполнения числовых вычислений, при которых сохраняется согласованность типов.

**Синтаксис:** Необходимо импортировать array модуль для использования массивов.

**Пример:**

**from** **array** **import** array

arr = array('i', [1, 2, 3, 4]) *# Array of integers*

[Списки](https://www.geeksforgeeks.org/python-lists/) более гибкие, чем массивы, поскольку могут содержать элементы разных типов (целые числа, строки, объекты и т. д.). Они встроены в Python и не требуют импорта дополнительных модулей.

Списки поддерживают множество операций, которые могут изменять список.

**Пример:**

lst = [1, 'hello', 3.14, [1, 2, 3]]

*узнайте больше о* [*Разница между списком и массивом в Python*](https://www.geeksforgeeks.org/difference-between-list-and-array-in-python/)

**23. Что такое модули и пакеты в Python?**

A[модуль](https://www.geeksforgeeks.org/python-modules/)представляет собой отдельный файл, содержащий код Python (функции, переменные, классы), который может быть повторно использован в других программах. Вы можете думать об этом как о библиотеке кода. Например: **математика**это встроенный модуль, который предоставляет математические функции, такие как sqrt(), pi и т.д.

**import** **math**

print(math.sqrt(16))

[Пакет](https://www.geeksforgeeks.org/python-packages/)— это набор связанных между собой модулей, хранящихся в каталоге. Он помогает упорядочивать и группировать модули для более удобного управления. Например, пакет [**numpy**](https://www.geeksforgeeks.org/introduction-to-numpy/)содержит несколько модулей для выполнения числовых операций.

Чтобы создать пакет, каталог должен содержать специальный файл с именем **\_\_init\_\_.py**.

**Промежуточные Вопросы для интервью на Python**

**24. В чем разница между функциями xrange и range?**

[range() и xrange()](https://www.geeksforgeeks.org/range-vs-xrange-in-python/) — это две функции, которые можно использовать для выполнения определённого количества итераций в циклах for в Python.

* В Python 3 нет функции xrange, но функция range работает как xrange.
* В Python 2
  + **range()**— возвращает объект range, который является неизменяемым типом последовательности, генерирующим числа по запросу.
  + **xrange()** — эта функция возвращает объект-генератор, который можно использовать для отображения чисел только с помощью цикла. Конкретный диапазон отображается только по запросу и поэтому называется отложенным вычислением.

**25. Что такое понимание по словарю? Приведите пример**

[Понимание словаря](https://www.geeksforgeeks.org/python-dictionary-comprehension/) — это синтаксическая конструкция, упрощающая создание словаря на основе существующего итерируемого объекта.

*# Python code to demonstrate dictionary*

*# comprehension*

*# Lists to represent keys and values*

keys = ['a','b','c','d','e']

values = [1,2,3,4,5]

*# but this line shows dict comprehension here*

myDict = { k:v **for** (k,v) **in** zip(keys, values)}

*# We can use below too*

*# myDict = dict(zip(keys, values))*

print (myDict)

**Выходной сигнал**

{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

**26. Возможно ли создание кортежей в Python? Если да, то как, а если нет, то почему?**

[Генераторы кортежей](https://www.geeksforgeeks.org/why-no-python-tuple-comprehension/) не поддерживаются напрямую, но существующие в Python функции, такие как выражения-генераторы и функция tuple(), предоставляют гибкие альтернативы для создания кортежей из итерируемых данных.

*(i для i в (1, 2, 3))*

В Python невозможно создать кортеж, потому что в итоге получится генератор, а не кортеж.

**27. Проводите различие между списком и кортежем?**

Давайте проанализируем[различия между списком и кортежем](https://www.geeksforgeeks.org/python-difference-between-list-and-tuple/):

**Список**

* Списки - это изменяемый тип данных.
* Списки потребляют больше памяти
* Список лучше подходит для выполнения операций, таких как вставка и удаление.
* Выполнение итераций требует много времени

**Кортеж**

* Кортежи - это неизменяемый тип данных.
* Кортеж потребляет меньше памяти по сравнению со списком
* Тип данных кортежа подходит для доступа к элементам
* Выполнение итераций происходит сравнительно быстрее

**28. В чём разница между поверхностной и глубокой копией?**

Ниже приведена таблица,[показывающая разницу](https://www.geeksforgeeks.org/difference-between-shallow-and-deep-copy-of-a-class/)между поверхностным и глубоким копированием:

| **Неглубокая копия** | **Глубокое Копирование** |
| --- | --- |
| Неглубокая копия хранит ссылки объектов на исходный адрес в памяти. | Глубокое копирование хранит копии значения объекта. |
| Неглубокое копирование отражает изменения, внесённые в новый/скопированный объект в исходном объекте. | Глубокое копирование не отражает изменения, внесённые в новый/скопированный объект в исходном объекте. |
| Неглубокое копирование сохраняет копию исходного объекта и указывает ссылки на объекты. | Глубокая копия сохраняет копию исходного объекта и рекурсивно копирует объекты. |
| Неглубокая копия выполняется быстрее. | Глубокое копирование выполняется сравнительно медленнее. |

**29. Какой метод сортировки используется функциями sort() и sorted() в Python?**

Python использует алгоритм сортировки [**Tim Sort**](https://www.geeksforgeeks.org/timsort/). Это устойчивая сортировка, наихудший случай которой имеет сложность O(N log N). Это гибридный алгоритм сортировки, созданный на основе сортировки слиянием и сортировки вставками и хорошо работающий со многими видами реальных данных.

**30. Кто такие декораторы?**

[Декораторы](https://www.geeksforgeeks.org/decorators-in-python/) — это мощный и гибкий способ изменять или расширять поведение функций или методов без изменения их фактического кода. Декоратор — это, по сути, функция, которая принимает другую функцию в качестве аргумента и возвращает новую функцию с расширенными возможностями.

Декораторы часто используются в таких сценариях, как ведение журнала, аутентификация и запоминание, позволяя нам добавлять дополнительные функции к существующим функциям или методам простым и многократно используемым способом.

**31. Как вы отлаживаете программу на Python?**

**1. Использование pdb (отладчика Python):**

pdb — это встроенный модуль, который позволяет устанавливать точки останова и пошагово выполнять код. Вы можете запустить отладчик, добавив import pdb; pdb.set\_trace() в код в том месте, где вы хотите начать отладку.

**import** **pdb**

x = 5

pdb.set\_trace() *# Debugger starts here*

print(x)

**Выходной сигнал**

> /home/repl/02c07243-5df9-4fb0-a2cd-54fe6d597c80/main.py(4)<module>()

-> print(x)

(Pdb)

**2. Использование модуля ведения журнала:**

Для более продвинутой отладки модуль ведения журнала предоставляет гибкий способ записи сообщений с различными уровнями важности (INFO, DEBUG, WARNING, ERROR, CRITICAL).

**import** **logging**

logging.basicConfig(level=logging.DEBUG)

logging.debug("This is a debug message")

**Выходной сигнал**

DEBUG:root:This is a debug message

**32. Что такое итераторы в Python?**

В Python [итераторы](https://www.geeksforgeeks.org/iterators-in-python/)используются для перебора группы элементов, таких как список. Итераторы — это коллекции элементов, которые могут быть представлены в виде списка, кортежа или словаря. Итератор Python реализует \_\_itr\_\_ и метод next() для перебора сохранённых элементов. Обычно для перебора коллекций (списков, кортежей) в Python используются циклы.

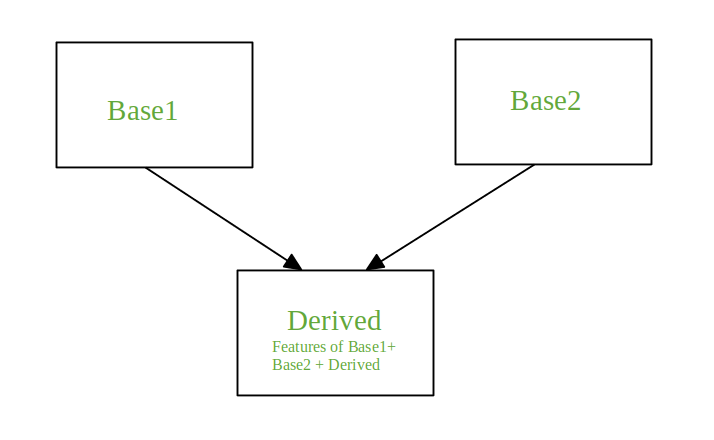
**33. Что такое генераторы в Python?**

В Python [генератор](https://www.geeksforgeeks.org/generators-in-python/)— это способ, определяющий, как реализовать итераторы. Это обычная функция, за исключением того, что она возвращает выражение в функции. Она не реализует методы \_\_itr\_\_ и \_\_next\_\_ и снижает другие накладные расходы.

Если функция содержит хотя бы один оператор yield, она становится генератором. Ключевое слово yield приостанавливает текущее выполнение, сохраняя его состояние, а затем возобновляет его при необходимости.

**34. Поддерживает ли Python множественное наследование?**

Когда класс наследуется более чем от одного базового класса, это называется множественным наследованием. Производный класс наследует все свойства базового класса.



*Множественное наследование*

Python поддерживает множество [наследований](https://www.geeksforgeeks.org/inheritance-in-python/), в отличие от Java.

**35. Что такое полиморфизм в Python?**

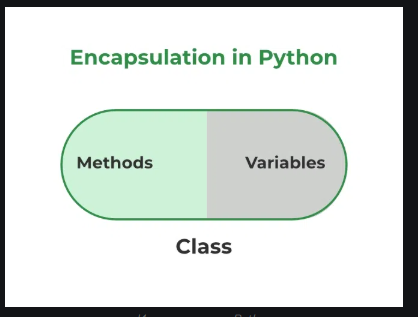
[Полиморфизм](https://www.geeksforgeeks.org/polymorphism-in-python/) означает способность принимать различные формы. Полиморфизм позволяет обращаться к разным классам так, как если бы они были экземплярами одного и того же класса, с помощью общего интерфейса. Это означает, что метод в родительском классе может быть переопределён методом с таким же именем в дочернем классе, но дочерний класс может предоставить свою собственную реализацию. Это позволяет одному и тому же методу работать по-разному в зависимости от объекта, который его вызывает. Полиморфизм — это переопределение, а не перегрузка; он позволяет методам работать с объектами разных классов, которые могут иметь собственные атрибуты и методы, обеспечивая гибкость и возможность повторного использования кода.

**36. Как определить инкапсуляцию в Python?**

[Инкапсуляция](https://www.geeksforgeeks.org/encapsulation-in-python/)— это процесс сокрытия внутреннего состояния объекта и требование, чтобы все взаимодействия осуществлялись через методы объекта. Этот подход:

* Обеспечивает лучший контроль над данными.
* Предотвращает случайное изменение данных.
* Продвигает модульное программирование.

Python обеспечивает инкапсуляцию с помощью **открытых**, **защищённых** и **закрытых** атрибутов.



*Инкапсуляция в Python*

**37. Как вы выполняете абстракцию данных в Python?**

[Абстракция данных](https://www.geeksforgeeks.org/data-abstraction-in-python/) предоставляет только необходимые сведения и скрывает реализацию от внешнего мира. Основное внимание уделяется раскрытию только основных функций и сокрытию сложной реализации за интерфейсом. В Python это можно реализовать с помощью интерфейсов и абстрактных классов.

**38. Как осуществляется управление памятью в Python?**

Python использует собственное пространство кучи для [управления](https://www.geeksforgeeks.org/memory-management-in-python/)памятью. По сути, все объекты и структуры данных хранятся в собственном пространстве кучи. Даже программист не может получить доступ к этому пространству, так как интерпретатор заботится об этом пространстве. В Python также есть встроенный сборщик мусора, который перерабатывает всю неиспользуемую память, освобождает её и делает доступной для пространства кучи.

**39. Как удалить файл с помощью Python?**

Мы можем удалить файл с помощью Python, используя следующие подходы:

1. Python Удаляет файл с помощью [os. remove](https://www.geeksforgeeks.org/python-os-remove-method/)
2. Удалить файл в Python с помощью [модуля send2trash](https://www.geeksforgeeks.org/how-to-delete-files-in-python-using-send2trash-module/)
3. Python Удаляет файл с помощью [os.rmdir](https://www.geeksforgeeks.org/python-os-rmdir-method/)

**40. Что такое нарезка в Python?**

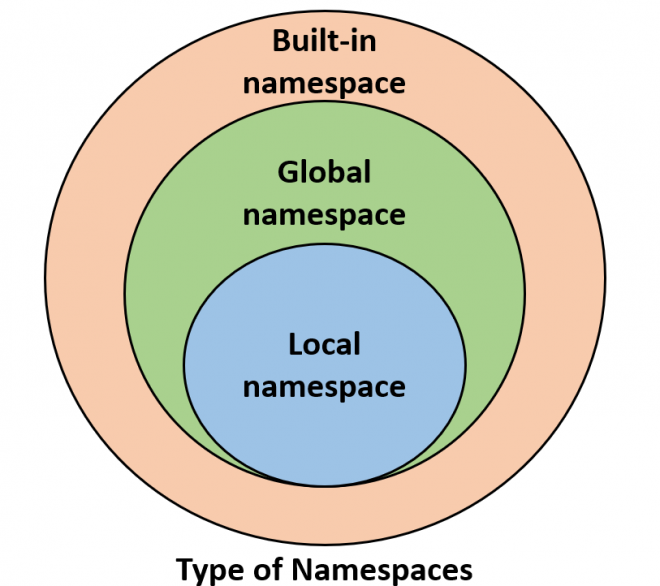
[Срез в Python](https://www.geeksforgeeks.org/python-slice-function/) — это строковая операция для извлечения части строки или части списка. С помощью этого оператора можно указать, где начать срез, где закончить и задать шаг. Срез списка возвращает новый список из существующего.

***Синтаксис:***

*substring = s[начало : конец : шаг]*

**41. Что такое пространство имен в Python?**

Пространство имен[в Python](https://www.geeksforgeeks.org/namespaces-and-scope-in-python/)— это контейнер, в котором имена (переменные, функции, объекты) сопоставляются с объектами. Проще говоря, пространство имен — это место, где определяются и хранятся имена, и оно помогает избежать конфликтов имен, гарантируя уникальность имен в заданной области.



*Типы пространств имен*

**Типы пространств имен**:

1. **Встроенное пространство имён**: содержит все встроенные функции и исключения, такие как print(), int() и т. д. Они доступны в каждой программе Python.
2. **Глобальное пространство имен**: содержит имена всех объектов, функций и переменных в программе на верхнем уровне.
3. **Локальное пространство имен**: относится к именам внутри функции или метода. Каждый вызов функции создает новое локальное пространство имен.

**Вопросы и ответы для Собеседования на Продвинутом Python**

**42. Что такое PIP?**

[PIP](https://www.geeksforgeeks.org/python-pip/) — это аббревиатура от Python Installer Package, которая обеспечивает удобный интерфейс для установки различных модулей Python. Это инструмент командной строки, который может искать пакеты в интернете и устанавливать их без участия пользователя.

**43. Что такое функция zip?**

Функция Python [zip()](https://www.geeksforgeeks.org/zip-in-python/) возвращает объект zip, который отображает аналогичный индекс для нескольких контейнеров. Он принимает iterable, преобразует его в итератор и агрегирует элементы на основе переданных итераций. Он возвращает итератор кортежей.

***Синтаксис:*** *zip (\* повторяющиеся файлы)*

**44. Что такое маринование и расконсервирование?**

* **Сериализация:**модуль pickle преобразует любой объект Python в поток байтов (а не в строковое представление). Затем этот поток байтов можно сохранить в файле, отправить по сети или сохранить для последующего использования. Функция, используемая для сериализации, — pickle.dump().
* **Распаковка:**Процесс извлечения исходного объекта Python из потока байтов (сохраняемого во время распарковки) называется распаковкой. Функция, используемая для распаковки, — pickle.load().

**45. В чём разница между @classmethod, @staticmethod и методами экземпляра в Python?**

1. Метод экземпляра работает с экземпляром класса, имеет доступ к атрибутам экземпляра и принимает self в качестве первого параметра. Пример:

*метод определения (self):*

2. Метод класса напрямую воздействует на сам класс, а не на экземпляр, он принимает cls в качестве первого параметра и определяется с помощью[@classmethod.](https://www.geeksforgeeks.org/classmethod-in-python/)

***Пример:*** *Метод определения @classmethod (cls):*

3. Статический метод не работает с экземпляром или классом и не принимает в качестве аргумента self или cls и определяется с помощью [@staticmethod.](https://www.geeksforgeeks.org/python-staticmethod/)

***Пример:*** *@staticmethod def метод(): выровняйте его и не выделяйте жирным шрифтом и не используйте маркированные списки*

**46. Что такое \_\_init\_\_() в Python и какую роль в нём играет self?**

[Метод \_\_init\_\_()](https://www.geeksforgeeks.org/__init__-in-python/) в Python эквивалентен конструкторам в терминологии ООП. Это зарезервированный метод в классах Python, который вызывается автоматически при создании нового объекта. Этот метод используется для инициализации атрибутов объекта значениями. Хотя \_\_init\_\_() инициализирует объект, он не выделяет память. Выделение памяти для нового объекта осуществляется методом \_\_new\_\_(), который вызывается перед \_\_init\_\_(). selfПараметр в \_\_init\_\_() относится к создаваемому экземпляру класса, поскольку он обеспечивает доступ к атрибутам и методам экземпляра. *self* должен быть явно объявлен в качестве первого параметра во всех методах экземпляра, включая \_\_init\_\_().

**class** **MyClass**:

**def** \_\_init\_\_(self, value):

self.value = value *# Initialize object attribute*

**def** display(self):

print(f"Value: **{**self.value**}**")

obj = MyClass(10)

obj.display()

**Выходной сигнал**

Value: 10

**47. Напишите код для отображения текущего времени?**

**import** **time**

currenttime= time.localtime(time.time())

print ("Current time is", currenttime)

**48. Что такое спецификаторы доступа в Python?**

В Python символ «\_» используется для определения контроля доступа к конкретному элементу данных или функции-члену класса. Класс в Python имеет три типа [модификаторов доступа в Python](https://www.geeksforgeeks.org/access-modifiers-in-python-public-private-and-protected/):

* **Модификатор открытого доступа:** элементы класса, объявленные как открытые, легко доступны из любой части программы. Все элементы данных и функции-члены класса по умолчанию являются открытыми.
* **Модификатор защищенного доступа:**Члены класса, объявленные как защищенные, доступны только для класса, производного от него. Все элементы данных класса объявляются как защищенные путем добавления одного символа подчеркивания «\_» перед элементами данных этого класса.
* **Модификатор закрытого доступа:**члены класса, объявленные закрытыми, доступны только внутри класса. Модификатор закрытого доступа является самым безопасным модификатором доступа. Члены данных класса объявляются закрытыми путем добавления символа двойного подчеркивания «\_\_» перед членом данных этого класса.

**49. Что такое модульные тесты в Python?**

[Модульное тестирование](https://www.geeksforgeeks.org/unit-testing-python-unittest/) — это первый уровень тестирования программного обеспечения, на котором тестируются самые маленькие тестируемые части программного обеспечения. Это используется для проверки того, что каждый модуль программного обеспечения работает так, как задумано. Фреймворк модульного тестирования — это фреймворк Python в стиле xUnit. Для модульного тестирования используется метод «белого ящика».

**50. Глобальная блокировка интерпретатора Python (GIL)?**

[Глобальная блокировка интерпретатора Python](https://www.geeksforgeeks.org/what-is-the-python-global-interpreter-lock-gil/) (GIL) — это тип блокировки процесса, который используется Python при работе с процессами. Как правило, Python использует только один поток для выполнения набора написанных операторов. Производительность однопоточного и многопоточного процессов в Python будет одинаковой из-за GIL в Python. Мы не можем реализовать многопоточность в Python, потому что у нас есть глобальная блокировка интерпретатора, которая ограничивает потоки и работает как один поток.

**51. Что такое аннотации функций в Python?**

[Аннотация функции](https://www.geeksforgeeks.org/function-annotations-python/) — это функция, которая позволяет добавлять метаданные к параметрам функции и возвращаемым значениям. Таким образом, вы можете указать тип входных параметров функции и тип возвращаемого значения.

Аннотации функций — это произвольные выражения Python, которые связаны с различными частями функций. Эти выражения вычисляются во время компиляции и не используются в среде выполнения Python. Python не придаёт этим аннотациям никакого значения. Они используются при интерпретации сторонними библиотеками, например, mypy.

**52. Что такое группы исключений в Python?**

Последняя функция Python 3.11, [группы исключений](https://www.geeksforgeeks.org/exception-groups-in-python/). Группа исключений может обрабатываться с помощью нового синтаксиса except\*. Символ \* указывает на то, что каждое предложение except\* может обрабатывать несколько исключений.

ExceptionGroup — это набор/группа различных типов исключений. Без создания множества исключений мы можем группировать различные исключения, которые затем можно будет извлекать по одному по мере необходимости. Порядок, в котором исключения хранятся в группе исключений, не имеет значения при их вызове.

try:

raise ExceptionGroup('Example ExceptionGroup', (

TypeError('Example TypeError'),

ValueError('Example ValueError'),

KeyError('Example KeyError'),

AttributeError('Example AttributeError')

))

except\* TypeError:

...

except\* ValueError as e:

...

except\* (KeyError, AttributeError) as e:

...

**53. Что такое оператор переключения в Python?**

Начиная с версии 3.10, в Python реализована функция [переключения по условию](https://www.geeksforgeeks.org/switch-case-in-python-replacement/), называемая «структурным сопоставлением шаблонов». Вы можете реализовать эту функцию с помощью ключевых слов match и case. Обратите внимание, что символ подчеркивания используется для определения случая по умолчанию в операторе switch в Python.

**Примечание**: до версии Python 3.10 Python не поддерживал операторы сопоставления.

**match** term:

**case** pattern-1:

action-1

**case** pattern-2:

action-2

**case** pattern-3:

action-3

**case** **\_**:

action-default

**54. Что такое оператор Walrus?**

[Оператор «морж»](https://www.geeksforgeeks.org/walrus-operator-in-python-3-8/) позволяет присваивать значение переменной в рамках выражения. Это может быть полезно, когда вам нужно использовать значение несколько раз в цикле, но вы не хотите повторять вычисления.

Оператор присваивания представлен синтаксисом `:=` и может использоваться в различных контекстах, включая циклы while и операторы if.

**Примечание:**версии Python до 3.8 не поддерживают оператор Walrus.

numbers = [1, 2, 3, 4, 5]

**while** (n := len(numbers)) > 0:

print(numbers.pop())

**Выходной сигнал**

5

4

3

2

1