Здравствуйте! Я, Каплуненко Дмитрий Дмитриевич, заинтересовался вашей вакансией на случай возможности работы в удаленном режиме.

Здесь отвечу на ваши теоретические вопросы:

1. Чем, по-вашему, Python превосходит остальные языки программирования? Для каких задач он подходит лучше всего? Для чего он не подходит?

Ответ: на сегодняшний момент Python является одним из самых популярных языков программирования, привлекательным для решения большого спектра задач. Перечислю основные преимущества использования:

1. Простота синтаксиса и читаемость кода: разработчик может писать код быстрее, а поддержка данного кода впоследствии упрощается (когда этот код должен прочесть, например, кто-то другой для кросс-валидации).
2. Обширный депозитарий стандартных библиотек: репозитарий Python (тот, который pip install <..>) имеет множество модулей для решения различных задач, таких как работа с файлами, сетевое программирование, парсинг данных, обработка изображений, научные вычисления и многое другое, что делает его универсальным инструментом быстрой разработки разнородных приложений.
3. Наличии сторонних библиотек, которые легко создавать. Эти библиотеки поддерживаются сообществом Python, что позволяет разработчикам использовать готовые решения для ряда задач, что существенно сокращает время и усилия, затрачиваемые на разработку.
4. Разнообразие поддерживаемых парадигм программирования: процедурное, объектно-ориентированное, функциональное и др. Это позволяет разработчикам выбрать наиболее подходящий стиль программирования для конкретной задачи.
5. Кроссплатформенность: Python доступен на различных платформах, таких как Windows, macOS, Linux, и может быть использован для разработки приложений, работающих на различных операционных системах. Есть, конечно, нюансы с применением библиотек – не все поддерживаются во всех ОС, но, как правило, таких меньшинство и вполне можно быстро найти адекватную замену.

Python подходит лучше всего для множества задач, таких как веб-разработка, научные исследования, анализ данных, машинное обучение, автоматизация задач, разработка игр, сетевое программирование (включая хакинг и пен-тестинг) и многое другое.

Python не подходит, т.е. не является оптимальным выбором в следующих случаях:

1. При работе с высокопроизводительными вычислениями: Python является интерпретируемым языком, что может снижать его производительность при выполнении высокопроизводительных вычислений, таких как интенсивные вычисления в реальном времени, обработка больших объемов данных или научные расчеты. В таких случаях, языки программирования с более низким

уровнем абстракции, такие как C++ или Fortran, могут быть более подходящим выбором.

2. Встроенные системы и микроконтроллеры (Arduino, STM32, и т.п.): Python может быть не оптимальным выбором для разработки встроенных систем, таких как системы на микроконтроллерах или встраиваемых системах с ограниченными ресурсами. В таких случаях, языки программирования с более низким потреблением ресурсов, такие как C (C++ или WIRED) или Assembly, могут быть предпочтительнее.
3. Компиляция в нативный код: Если необходимо создавать приложения, компилирующиеся в нативный машинный код для достижения максимальной производительности, то Python может не быть наилучшим выбором. Языки программирования, такие как C или C++, обеспечивают большую гибкость и контроль над процессом компиляции и оптимизации, что может быть важно в некоторых случаях.
4. Ограничения на производительность: Если производительность является критическим фактором для конкретной задачи, и нет возможности использования сторонних оптимизированных библиотек на Python, то другие языки программирования, такие как C, C++, Rust или Go, могут предоставлять более эффективные решения.
5. Ограничения на память: Python, как интерпретируемый язык, может иметь ограничения на использование памяти, особенно при работе с большими объемами данных. Если задача требует работы с огромными объемами данных и оптимизации использования памяти, то другие языки, такие как C или C++, могут быть более подходящими.

Это не означает, что Python не может быть использован для этих задач вовсе, но в таких случаях могут быть лучшие альтернативы с учетом требований производительности, ограничений памяти, требуемого уровня оптимизации и других факторов. Важно тщательно оценить требования задачи и выбрать наиболее подходящий язык программирования на основе этих требований, вместо того чтобы всегда использовать только один язык. Во многих случаях, комбинация различных языков программирования может быть оптимальным решением для сложных проектов, чтобы достичь оптимальной производительности, эффективности использования ресурсов и функциональности.

2. Какие особенности Python могут доставлять больше всего неудобств?

Хотя Python является мощным и универсальным языком программирования, он также имеет свои особенности, которые могут вызывать неудобства в определенных случаях. Некоторые из таких особенностей могут включать:

1. Производительность: Python, в сравнении с некоторыми другими языками программирования, такими как C++ или Go, может быть менее

производительным, особенно в случаях, когда требуется высокая скорость выполнения или работа с большими объемами данных. Из-за своей динамической природы и автоматического управления памятью, Python может иметь ограничения производительности в определенных сценариях, таких как научные вычисления или обработка больших данных.

2. Ограничения мобильной разработки: Python не всегда является идеальным выбором для мобильной разработки, поскольку приложения, написанные на Python, обычно работают на интерпретаторе, что может снижать производительность на мобильных устройствах. Однако существуют фреймворки, такие как Kivy и BeeWare, которые позволяют создавать кроссплатформенные мобильные приложения на Python.
3. Ограничения ресурсов: Python, как интерпретируемый язык, может потреблять больше ресурсов (таких как память и процессорное время) в сравнении с компилируемыми языками, особенно при выполнении высоконагруженных или высокопроизводительных задач.
4. Глобальная интерпретация и GIL: Python имеет глобальную интерпретацию, что означает, что одинаковый код выполняется на всех платформах, и это может вызывать некоторые несоответствия и неудобства в разработке многоплатформенных приложений. Кроме того, в CPython (наиболее распространенная реализация Python) существует Global Interpreter Lock (GIL), ограничивающий потенциал многопоточности в Python, что может вызывать ограничения при разработке многопоточных приложений.
5. Ограничения встраиваемости: Python может быть ограничен во встраиваемости в некоторые системы или аппаратное обеспечение. Например, в некоторых встроенных системах, таких как микроконтроллеры или встраиваемые системы реального времени, использование Python может быть ограничено из-за его высокого уровня абстракции и автоматического управления памятью, что может вызывать проблемы с производительностью, доступом к аппаратным ресурсам и размером исполняемых файлов.

3. Возможно ли писать программы на Python в функциональном стиле? Насколько он для этого подходит?

Как уже выше упоминалось, Python является является многопарадигменным языком программирования и поддерживает функциональное программирование в дополнение к объектно-ориентированному и императивному стилям. Хотя Python в основном является императивным языком, он также предоставляет множество функциональных возможностей, таких как функции первого класса, замыкания, анонимные функции (lambda-функции), списковые включения, и многие другие.

Python предлагает множество встроенных функций, таких как map(), filter(), reduce(), и других, которые могут использоваться для написания программ в функциональном стиле. Он также поддерживает библиотеки, такие как functools

и itertools, которые предоставляют дополнительные функциональные возможности.

Однако стоит отметить, что Python все же больше ориентирован на объектно-ориентированное и императивное программирование, и функциональный стиль может не всегда быть наиболее эффективным или прозрачным в Python коде. Некоторые функциональные концепции, такие как неизменяемость данных, могут быть сложнее реализовать в Python из-за его мутабельных структур данных, таких как списки и словари.

Таким образом, Python вполне подходит для функционального программирования и предоставляет множество соответствующих возможностей, но также имеет свои особенности и ограничения, которые стоит учитывать при написании программ в функциональном стиле.

В качестве примера выложена программа на моем репозитории. Вот ссылка:

https://github.com/kitchen-bear/funbox/blob/main/check_factorial.py

4. Какие инструменты и методы помогают экономить время и избегать ошибок при написании кода в Python?

Существует несколько инструментов и методов, которые могут помочь экономить время и избегать ошибок при написании кода на Python:

1. Интегрированные среды разработки (IDE): Использование популярных IDE, таких как PyCharm, VSCode, или PyDev, может значительно упростить разработку Python-кода. Они предлагают автодополнение, отладку, проверку синтаксиса, и другие функции, которые помогают избегать ошибок и повышают производительность.
2. Статические анализаторы кода: Использование инструментов статического анализа, таких как Pyright, Flake8, или Pyre, может помочь выявлять потенциальные ошибки в коде до его выполнения, такие как ошибки синтаксиса, типизации или стиля кодирования.
3. Отладчики: Python предоставляет встроенные отладочные инструменты, такие как pdb (Python Debugger), которые позволяют искать и исправлять ошибки в коде, а также отслеживать выполнение программы на шаги.
4. Тестирование: Написание автоматических тестов с использованием библиотек таких, как unittest, pytest, или nose, позволяет обнаруживать и исправлять ошибки раньше, а также обеспечивает надежность кода при его изменениях.
5. Использование библиотек и фреймворков: Использование сторонних библиотек и фреймворков, таких как NumPy, pandas, Django, или Flask,

может значительно сократить время разработки, предоставив готовые решения для распространенных задач.

6. Документация и сообщество: Хорошая документация и активное сообщество Python-разработчиков могут помочь экономить время и избегать ошибок, предоставляя решения для распространенных проблем, а также советы и рекомендации по написанию эффективного и безопасного кода.
7. Стиль кодирования: Соблюдение рекомендаций по стилю кодирования, таким как PEP 8 (стандарт стиля кодирования Python), может сделать код более читабельным и понятным, упрощая его поддержку и снижая вероятность ошибок.
8. Рефакторинг: Периодический рефакторинг кода, то есть его оптимизация, улучшение структуры и удаление дублирующегося кода, может уменьшить вероятность ошибок, повысить читабельность и поддерживаемость кода, что в итоге экономит время и помогает избегать ошибок при написании кода на Python. Рефакторинг - это процесс преобразования кода без изменения его функциональности, с целью улучшения его качества.

Кроме отмеченного существует множество других инструментов, практик и методик, которые также могут быть полезны в повседневной работе программиста, в зависимости от конкретной задачи, стека технологий и личных предпочтений разработчика. Некоторые из таких инструментов и методов могут включать использование интегрированных сред разработки (IDE) с автодополнением кода, автоматическим форматированием, статическим анализом кода и отладочными возможностями, использование систем контроля версий, написание модульных тестов, использование сторонних библиотек и фреймворков, использование документации и сообществ разработчиков, а также поддержание чистоты и организации кода с использованием принципов SOLID, DRY, KISS и других. Выбор оптимальных инструментов и методов может зависеть от конкретных условий и предпочтений программиста, и их использование может помочь сэкономить время и избегать ошибок при написании кода на Python.

5. На каких ресурсах можно получить информацию и новости по используемым технологиям?

Есть множество ресурсов, где можно получить информацию и новости о технологиях, связанных с Python. Некоторые из них включают:

1. Официальная документация Python: Официальный сайт Python (https://docs.python.org) предлагает подробную документацию, включая руководства, справочники и PEP (Python Enhancement Proposals) - документы, описывающие новые возможности и изменения в Python.

2. Python.org: Официальный сайт Python (https://www.python.org) также предлагает раздел "News" со свежими новостями и анонсами, связанными с Python, а также блоги и подкасты.
3. Python Weekly (https://www.pythonweekly.com/): Еженедельный бесплатный бюллетень, который содержит последние новости, статьи, руководства, библиотеки и события, связанные с Python.
4. PyCon (https://www.pycon.org/): Международная конференция, посвященная Python, проводимая ежегодно. Во время PyCon представляются новые технологии, проводятся доклады и мастер-классы, и есть возможность взаимодействия с сообществом Python.
5. GitHub (https://github.com/): Онлайн-платформа для хранения и совместной разработки программного обеспечения. Множество проектов, связанных с Python, хранятся на GitHub, и это может быть отличным источником информации о новых библиотеках, фреймворках, инструментах и техниках, используемых в сообществе Python.
6. Python Reddit (https://www.reddit.com/r/Python/): Реддит - популярная платформа социальных новостей и дискуссий. Сообщество Python на Reddit предлагает множество дискуссий, новостей, советов и ресурсов, связанных с Python.
7. Официальные блоги и сайты разработчиков библиотек и фреймворков Python: Многие библиотеки и фреймворки, используемые в Python, имеют официальные блоги или веб-сайты, где разработчики публикуют новости, обновления, документацию и руководства.
8. Курсы и онлайн-обучение: Множество платформ для онлайн-обучения, таких как Coursera, Udemy, Pluralsight и другие, предлагают курсы и обучение, связанные с Python.
9. Stack Overflow (https://stackoverflow.com/questions/tagged/python): Популярный вопросно-ответный ресурс, где программисты могут задавать вопросы и получать ответы на темы, связанные с Python. Множество вопросов и ответов содержит актуальную информацию о различных аспектах Python, включая новые технологии и методы.
10. Twitter: Социальная сеть Twitter является популярным местом для обмена новостями, статьями, обновлениями и мнениями о Python. Множество разработчиков, библиотек и фреймворков Python, а также известных экспертов по Python активно общаются и делятся информацией на Twitter.
11. Официальные социальные медиа-аккаунты Python: Многие организации, связанные с Python, имеют официальные аккаунты в социальных сетях, таких как Twitter, Facebook, LinkedIn и др., где они регулярно публикуют новости, обновления и полезную информацию о Python и связанных технологиях.
12. Книги и журналы: Множество книг и журналов по Python регулярно публикует новости, статьи и обновления о последних тенденциях и разработках в мире Python.

6. Рассказать о себе и последних работах (в рамках допустимого).

В свете обсуждаемой вакансии, думаю, вас интересует опыт python-разработчика… Для этого открыл один из моих репозиториев на github, чтобы можно было на это посмотреть своими глазами:
https://github.com/kitchen-bear/td-bot
Где:
https://github.com/kitchen-bear/td-bot/blob/master/tradeogre.py - библиотека wrapper api-функций для работы с криптовалютной биржей…
https://github.com/kitchen-bear/td-bot/blob/master/tr_trading_bot.py - торговый бот, использующий данную библиотеку…
В README все честно указано, что-куда и откуда…
Формально никогда коммерческой разработкой на питоне не занимался. Тем не менее, с 2018 года у меня несколько закрытых научных проектов, где использовался питон – в основном это работа с данными по климату, обработка данных в формате netcdf, визуализация и анализ…в качестве пруфлинка можно привести некоторые из статей за моим соавторством, основанные на обработке упомянутых данных:

• Kaplunenko D.D., Lobanov V.B., Ostrovsky A.G., Lazaryuk A.Yu., Gulenko T.A., Chang K.I., Nam S.H., Yoon S.T Short-term variability of thermohaline characteristics in the northwestern Japan/East Sea from moored buoy data in 2013-2016 // Abstracts of the 26th International Conference of Pacific Congress on Marine Science and Technology (PACON-2019), July 16–19, 2019, Владивосток, ISBN: 978-5-6043211-0-2, С. 330
• Сорокин М.А., Петров П.С., Каплуненко Д.Д., Степанов Д.В., Моргунов Ю.Н. Оценка влияния синоптических вихрей на точность решения задач акустической дальнометрии // Подводные исследования и робототехника. 2020, Т. 34, № 4. С. 66-69, Владивосток: Дальнаука, ISSN-ISBN: 1992-4429 e-2409-4609
• Sorokin M.A., Petrov P.S., Kaplunenko D.D., Golov A.A., Morgunov Yu.N. Predicting effective propagation velocities of acoustic signals using an ocean circulation model // Acoustical Physics 2021 Т.102. № 97. С. 501-511. American Institute of Physics. DOI: 10.1134/S1063771021050080
• Каплуненко Д.Д., Дубина В.А., Моргунов Ю.Н., Голов А.А ВОССТАНОВЛЕНИЕ ГОРИЗОНТАЛЬНЫХ ПОЛЕЙ СКОРОСТИ ЗВУКА В ЯПОНСКОМ МОРЕ НА ОСНОВЕ СПУТНИКОВЫХ И МОДЕЛЬНЫХ ДАННЫХ .// Подводные исследования и робототехника, 2021, Владивосток: Дальнаука , Т. 36. № 2. С. 28-40 DOI: 10.37102/1992-4429_2021_36_02_03
• Каплуненко Д. Д., Зотов С. С., Суботэ А. Е., Фищенко В. К. Применение нейронных сетей для классификации биологических объектов по подводным камерам МЭС острова Попова // Подводные исследования и робототехника. – 2022. – № 1(39). – С. 72-79. – DOI 10.37102/1992-4429_2022_39_01_07.

7. Выполнение практических заданий. Выложено депозитарием на github. Можно посмотреть по ссылке:

https://github.com/kitchen-bear/funbox