1 Знакомство с ИИ-программированием и Copilot

В этой главе

- ✓ Как ИИ-помощники меняют методы обучения новых программистов.
- ✓ Почему программирование никогда не будет прежним.
- ✓ Как работают ИИ-помощники, подобные Copilot.
- ✓ Как Copilot решает задачи по программированию.
- ✓ Каковы опасности программирования с помощью ИИ.

Поговорим о том, как люди общаются с компьютерами. Мы познакомим вас с *ИИ-помощником*, GitHub Copilot, удивительным инструментом, который использует искусственный интеллект для помощи людям в написании программ. Что еще более важно, мы покажем, как Copilot может помочь вам научиться программировать. Мы не ожидаем, что вы уже писали какие-либо программы, но если да, то, пожалуйста, не пропускайте эту главу, даже если уже что-то знаете о программировании. Всем нужно знать, почему процесс написания программ теперь изменился, ведь у нас есть ИИ-помощники, такие как ChatGPT и Copilot, поэтому и навыки, необходимые нам, чтобы быть эффективными программистами, тоже меняются. Как вы увидите ниже, нужно быть бдительными, поскольку иногда такие инструменты, как ChatGPT и Copilot, лгут.

1.1. КАК МЫ РАЗГОВАРИВАЕМ С КОМПЬЮТЕРАМИ

Вам понравится, если мы начнем с того, что попросим вас прочитать этот код и разобраться в нем?[1]

```
section .text
global _start
_start:
    mov ecx, 10
    mov eax, '0'
    l1:
    mov [num], eax
    mov eax, 4
    mov ebx, 1
    push ecx
    mov ecx, num
    mov edx, 1
    int 0x80
    mov eax, [num]
    inc eax
    pop ecx
    loop l1
    mov eax, 1
    int 0x80
section .bss
    num resb 1
```

Это чудовище выводит числа от 0 до 9. Программа написана на языке ассемблера, низкоуровневом языке программирования. Низкоуровневые языки, как вы понимаете, — это не те языки, которые человек может легко прочитать и записать. Они предназначены для компьютеров, а не для людей.

Сейчас никто не хочет писать такие программы, но раньше это иногда было необходимо. Программисты могли использовать этот язык, чтобы указать, что именно они хотят сделать, и компьютер получал все команды, вплоть до отдельных инструкций. Такой уровень контроля был необходим, чтобы выжать максимум производительности из маломощных компьютеров. Например, наиболее влияющие на скорость фрагменты компьютерных игр 1990-х, таких как *Doom* и *Quake*, были написаны на языке ассемблера, как в предыдущем примере. Иначе сделать эти игры было бы невозможно.

---

[1] Основано на коде с сайта https://draftsbook.com/part-7-conditions-and-loop-uses-in-assembly-language/.

1.1.1. Слегка упростим

Но хватит об этом, пора двигаться дальше. Будет ли вам приятнее читать такой код?

```
for num in range(0, 9):
     print(num)
```

Он написан на языке Python, который в наши дни используют многие программисты. В отличие от языка ассемблера, низкоуровневого, Python считается высокоуровневым, поскольку гораздо более близок к естественному языку. Даже не зная Python, вы можете догадаться, что пытается сделать эта программа. Первая строка выглядит так, будто программа что-то делает с диапазоном чисел от 0 до 9. Вторая строка что-то выводит на печать. Нетрудно поверить, что эта программа, как и чудовище на языке ассемблера, должна вывести числа от 0 до 9. К сожалению, в ней что-то не так, и на самом деле она выводит числа от 0 до 8.

Такой код ближе к английскому языку, но это не английский в чистом виде. Это язык программирования, в котором, как и в языке ассемблера, есть определенные правила. Непонимание нюансов этих правил может привести к сбою программы, что мы и наблюдали в предыдущем коде.

Труднодостижимая цель общения с компьютером — общение на естественном языке, таком как английский. Последние 70 лет мы общаемся с компьютерами с помощью различных языков программирования, но не потому, что хотим, а потому, что вынуждены. Компьютерам не хватало мощности, чтобы справиться со всеми причудами и особенностями английского языка. Наши языки программирования эволюционировали: от насыщенного символами языка ассемблера до Python, но все равно остаются компьютерными, а не естественными языками. Сейчас ситуация меняется.

1.1.2. Упростим еще больше

С помощью ИИ-помощника мы можем спросить о том, что нам нужно, и получить в ответ написанный для нас компьютерный код. Чтобы получить корректную программу на Python, которая действительно выводит числа от 0 до 9, мы можем дать нашему ИИ-помощнику (Copilot) запрос (подсказка или промт от англ. promt) на обычном языке следующим образом:

```
# Выведи числа от 0 до 9
```

Copilot может ответить, сгенерировав что-то вроде этого:

```
for i in range(10):
    print(i)
```

В отличие от примера, который мы показали вам в подразделе 1.1.1, этот фрагмент кода на Python действительно работает!

ИИ-помощники по программированию способны помогать писать код. В книге мы разберемся, как делать это с помощью Copilot. Мы будем запрашивать то, что нам нужно, на обычном языке и получим код на Python.

Более того, мы сможем использовать Copilot как неотъемлемую часть нашего рабочего процесса. Не имея таких инструментов, как Copilot, программисты обычно держат открытыми два окна: одно — для написания кода, а другое — для запросов Google, как писать код. Во втором окне отображаются результаты поиска, документация по Python или форумы, на которых обсуждается, как же написать код для решения этой конкретной проблемы. Программисты часто вставляют такой код в свои программы, подгоняют его под свой контекст, пробуют альтернативные варианты и т. д. Это стало образом жизни программистов, но вы можете представить себе, насколько такие действия неэффективны. По некоторым оценкам, программисты тратят на поиск кода до 35 % времени [1], и бóльшая часть найденного кода непригодна для использования. Copilot значительно улучшает этот показатель, помогая писать код.

1.2. О ТЕХНОЛОГИЯХ

В книге мы будем использовать две основные технологии: Python и GitHub Copilot.

Python — это язык программирования. Он служит средством общения с компьютером. Люди используют Python для написания всевозможных программ: игр, интерактивных сайтов, инструментов визуализации, приложений для организации файлов, автоматизации рутинных задач и т. д.

Существуют и иные языки программирования: Java, C++, Rust и многие другие. Copilot работает и с ними, однако на момент написания книги очень хорошо работает только с Python. Код на Python намного проще писать по сравнению со многими другими языками (особенно с ассемблерным). Что еще более важно, Python легко *читать*. В конце концов, мы не будем писать код на Python. Это сделает наш ИИ-помощник!

На самом деле компьютеры не знают, как читать и выполнять код на Python. Единственное, что они могут понять, — это нечто называемое *машинным кодом*, который выглядит еще более нелепо, чем ассемблерный код, поскольку является двоичным представлением ассемблерного (да-да, просто набор 0 и 1!). Ваш компьютер принимает любой предоставленный вами код Python и перед запуском преобразует его в машинный код (рис. 1.1).

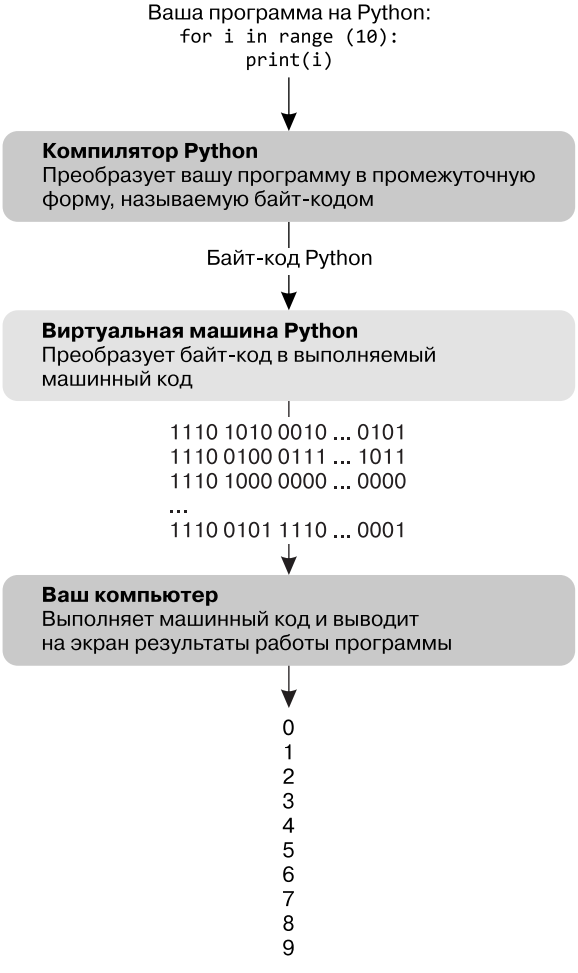

Рис. 1.1. Ваша программа на языке Python совершит несколько действий, прежде чем вы увидите на экране результат

1.2.1. Copilot, ваш помощник с искусственным интеллектом

Что такое ИИ-помощник? Это интеллектуальный агент, который помогает вам выполнять работу. Возможно, у вас дома есть умная колонка с «Алисой» или iPhone с Siri — все это разные ИИ-помощники. Они помогают заказывать продукты, узнавать погоду или определять, что да, женщина, сыгравшая Беллатрису в фильмах о Гарри Поттере, действительно снималась в «Бойцовском клубе».

ИИ-помощник — это просто компьютерная программа, которая реагирует на обычный человеческий запрос, например голосовой или текстовый, и отвечает на них так, как это сделал бы человек.

Copilot — это ИИ-помощник, имеющий особую способность: он преобразует человеческий язык в компьютерные программы (но может делать и многое другое, как вы скоро увидите). Есть и другие ИИ-помощники, подобные Copilot, например CodeWhisperer, Tabnine и Ghostwriter. Мы выбрали Copilot для данной книги благодаря сочетанию качества кода, который мы смогли создать с его помощью, стабильной работы (ни разу не завершилась аварийно) и нашим личным предпочтениям. Обязательно попробуйте и другие инструменты, если, конечно, вы почувствуете себя комфортно в этой области взаимодействия с ИИ.

1.2.2. Как работает Copilot: краткий обзор

Copilot можно рассматривать как прослойку между вами и компьютерной программой, которую вы пишете. Вместо того чтобы писать код Python напрямую, вы просто описываете нужную вам программу — создаете *запрос* — и Copilot генерирует ее.

«Мозг» Copilot — причудливая компьютерная программа, называемая *большой языковой моделью* (large language model, LLM). Она хранит информацию о связи слов, в том числе о том, какие слова имеют смысл в тех или иных контекстах, и с ее помощью предсказывает наилучшую последовательность слов для ответа на запрос.

Представьте, что мы спросили вас, каким должно быть следующее слово в этом предложении: «Человек открыл ___». Вы могли бы подобрать множество слов, например, «дверь», или «коробку», или «диалог», но есть и много слов, которые сюда не подойдут, например the, it или open. LLM учитывает текущий контекст, чтобы подобрать следующее слово, и продолжает делать это до тех пор, пока не выполнит задачу.

Обратите внимание: мы ничего не сказали о том, что Copilot должен понимать, что он делает. Он просто использует текущий контекст, чтобы продолжать писать код. Помните об этом на протяжении всей книги: только вы знаете, делает ли сгенерированный код то, что вы хотели сделать. Очень часто это именно так, но вы всегда должны проявлять здоровый скептицизм.

На рис. 1.2 показано, как Copilot переходит от запроса к программе.

Вы можете удивиться, почему Copilot пишет код на Python, а не сразу машинный код. Разве Python сейчас не является рядовым промежуточным этапом? Нет,

и причина в том, что иначе Copilot будет ошибаться. А если он будет допускать ошибки, которые нужно исправить, то сделать это с помощью Python гораздо проще, чем с помощью машинного кода.

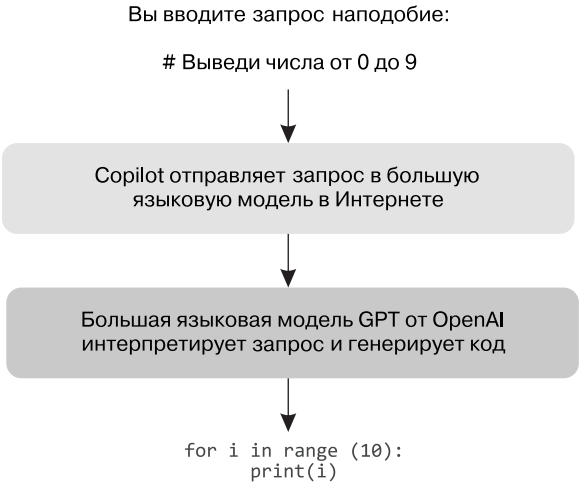

Рис. 1.2. Переход от запроса к программе с помощью Copilot

На самом деле практически никто не проверяет правильность машинного кода, созданного на Python. Отчасти это объясняется детерминизмом спецификации языка Python. Можно представить будущее, в котором выдача Copilot будет настолько точной, что проверка кода на Python не понадобится, но до этого еще далеко.

1.3. КАК COPILOT МЕНЯЕТ СИСТЕМУ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ

Чтобы проиллюстрировать, как меняется процесс обучения программированию при наличии ИИ-помощника, мы хотим показать, насколько эффективно Copilot решает задачу, которая обычно дается студентам в середине курса программирования или в книге о том, как надо программировать.

В прошлом при изучении курса учащиеся часто тратили бóльшую часть времени, работая с синтаксисом и базовой структурой программ. Написание программы с нуля (строка за строкой), подобно той, которую мы покажем вам ниже, считается кульминацией всего их обучения. Прежде чем мы погрузимся в задачу, необходимо прояснить: многие начинающие программисты не могут написать код для решения этой задачи, несмотря на то что большинство преподавателей программирования

и авторов книг считают, что новые программисты должны уметь это делать. Вот инструкции к одной из версий данной задачи [6].

> Напишите программу, которая будет считать количество осадков за каждый день. Отрицательное значение осадков должно быть отклонено, так как является недействительным и недопустимым. Программа должна выводить количество достоверно зарегистрированных дней, количество дождливых дней, количество осадков за период и максимальное количество осадков, выпавшее в один из дней. Используйте значение 9999 для завершения работы программы.

Если вы уже пытались научиться программировать, то такая постановка задачи может быть вам знакома и вы, возможно, даже пробовали решить подобную задачу. Но теперь, если мы дадим Copilot именно такой запрос, получим в результате следующий код на Python. (Напомним, что Copilot недетерминирован, поэтому если мы повторим этот запрос, то можем получить другой код!)

Листинг 1.1. Код Copilot, решающий классическую задачу с дождем

```
rainfall = 0
total_rainfall = 0
num_rainy_days = 0
num_valid_days = 0
max_rainfall = 0
rainfall = int(input(
    "Enter the amount of rainfall for the day: "))
while rainfall != 9999:
    if rainfall < 0:
        print("Invalid rainfall amount. Please try again.")
        rainfall = int(input(
            "Enter the amount of rainfall for the day: "))
    else:
        num_valid_days += 1
        total_rainfall += rainfall
        if rainfall > 0:
            num_rainy_days += 1
        if rainfall > max_rainfall:
            max_rainfall = rainfall
        rainfall = int(input(
            "Enter the amount of rainfall for the day: "))
print("Number of valid days:", num_valid_days)
print("Number of rainy days:", num_rainy_days)
print("Total rainfall over the period:", total_rainfall)
print("Maximum amount of rain that fell on any one day:", max_rainfall)
```

Мы не ожидаем, что вы прямо сейчас почувствуете, что этот код хорош. Но он действительно весьма неплох. Как преподаватели информатики, мы бы оценили его высоко.

Раньше люди, обучающиеся программированию, тратили недели или месяцы, чтобы дойти до того момента, когда они могли написать программы, подобные этой. Теперь

Copilot может предложить код сразу. Как вы увидите далее в книге, правильность этого кода все еще нужно проверять, поскольку Copilot может допускать ошибки. Однако нам больше не нужно писать код с нуля. Мы считаем, что это успешное взаимодействие с Copilot знаменует конец того способа преподавания и изучения программирования, который сложился исторически к настоящему моменту.

Теперь вам, как человеку, заинтересованному в изучении программирования, не нужно бороться с синтаксисом, потоками управления и множеством других понятий Python, необходимых для написания кода, как это было раньше. Конечно, в этой книге вы познакомитесь со всеми базовыми понятиями, но не для того, чтобы продемонстрировать свое понимание, написав код с нуля, который Copilot легко воспроизведет. Нет, мы будем объяснять эти концепции только потому, что они помогают решать сложные задачи и продуктивно взаимодействовать с Copilot. *Вы узнаете, как писать более объемные и значимые программы быстрее, поскольку ИИ-помощник кардинально меняет навыки, необходимые для изучения программирования.*

1.4. ЧТО ЕЩЕ МОЖЕТ ДЛЯ НАС СДЕЛАТЬ COPILOT

Как вы уже видели, с помощью Copilot можно написать код на Python, начиная с описания на английском того, что вам нужно. Программисты используют слово *«синтаксис»* для обозначения символов и слов, которые допустимы в данном языке. Таким образом, можно сказать, что Copilot принимает описание на английском и выдает код, соответствующий синтаксису Python. Это большая победа, поскольку изучение синтаксиса программирования было основным камнем преткновения начинающих программистов. Какую скобку — [, (или { — использовать? Нужен ли отступ в этом месте кода? В каком порядке писать: сначала x, затем y, или y, а потом x?

Таких вопросов множество, и будем честными: это неинтересно. Кого это волнует, когда все, чего мы хотим, — написать программу, чтобы что-то произошло. Copilot может избавить от утомительной рутины синтаксиса. Мы рассматриваем это как важный шаг, который поможет большему количеству людей успешно писать программы, и с нетерпением ждем дня, когда этот искусственный барьер будет устранен полностью. Пока же синтаксис Python нам по-прежнему нужен, но по крайней мере Copilot помогает нам с этим.

Но это далеко не все, на что способен Copilot. Вот несколько сопутствующих — и не менее важных — задач, которые Copilot может помочь решить.

- *Объяснение кода*. Когда Copilot генерирует код на Python, нам надо определить, делает ли этот код то, что нам нужно. Как мы уже говорили, Copilot будет делать ошибки. Мы не стремимся научить вас всем нюансам работы Python (это старая модель программирования), но научим вас читать код Python, чтобы вы могли

получить общее представление о том, что он делает. Кроме того, мы воспользуемся функцией Copilot, которая объясняет код человеческим языком. Когда вы закончите читать эту книгу и наши объяснения, у вас останется Copilot, который может помочь вам понять следующий фрагмент выданного им сложного кода.

- *Создание более понятного кода.* Есть разные способы написания кода, предназначенного для выполнения одной и той же задачи. Одни из них проще понять, чем другие. В Copilot есть инструмент, который может реорганизовать ваш код так, чтобы с ним было легче работать. Например, код, который легче читать, часто проще улучшить или исправить в случае необходимости.
- *Исправление багов.* Баг — это ошибка, допущенная при написании программы, которая может привести к тому, что программа сделает что-то не то. Иногда ваш код на Python почти работает или работает почти всегда, но не в одном конкретном случае. Если вы прислушивались к разговорам программистов, то, возможно, часто слышали историю, как программист потратил часы на то, чтобы в конце концов убрать один символ =, из-за которого программа не работала. Не самые приятные несколько часов! В таких случаях можно воспользоваться функцией Copilot, которая поможет автоматически найти и исправить ошибку в программе.

1.5. РИСКИ И ПРОБЛЕМЫ, СВЯЗАННЫЕ С ИСПОЛЬЗОВАНИЕМ COPILOT

Теперь, когда мы воодушевились идеей, что Copilot может писать код за нас, необходимо поговорить об опасностях, связанных с использованием ИИ-помощников. В источниках [2] и [3] подробно описаны некоторые из этих моментов.

- *Авторское право.* Copilot научился программировать, используя написанный человеком код. (Возможно, вы слышали, что люди используют слово «тренировка», когда говорят об ИИ-инструментах, подобных Copilot. Это еще один специальный термин для *обучения*.) Если говорить более конкретно, то Copilot был обучен с использованием миллионов репозиториев GitHub, содержащих открытый код. Одно из опасений — что Copilot «украдет» этот код и отдаст его нам в свободное пользование. По нашему опыту, Copilot нечасто предлагает большой кусок чужого кода, но такая возможность существует. Даже если код, который выдает Copilot, создан на основе слияния и преобразования различных фрагментов чужого кода, проблемы с лицензированием все равно могут возникнуть. Например, кому принадлежит код, созданный Copilot? В настоящее время единого мнения по этому поводу нет. Команда Copilot добавляет новые возможности, чтобы помочь; например, Copilot может сообщить вам, похож ли созданный им код на уже существующий и есть ли лицензия на этот код [4]. Самостоятельное обучение и экспериментирование — это отличное занятие, и мы его поощряем. Но будьте внимательны, если собираетесь использовать

такой код для целей, выходящих за пределы вашего дома. Это немного расплывчатое объяснение, и такая формулировка намеренна: может потребоваться некоторое время на то, чтобы законодательная база пришла в соответствие с этой новой технологией. Лучше перестраховаться, пока в обществе идут дебаты по этому поводу.

- *Обучение*. Будучи преподавателями курсов по программированию для начинающих, мы на собственном опыте убедились, насколько хорошо Copilot справляется с теми видами заданий, которые мы традиционно давали нашим студентам. В одном из исследований [5] Copilot попросили решить 166 обычных задач курса программирования. Насколько хорошо он справился? С первой попытки он решил почти 50 % этих задач. Если дать ему немного больше информации, то это число увеличится до 80 %. Вы уже видели, как Copilot решает стандартную задачу по программированию для начинающих. Благодаря таким инструментам, как Copilot, образование должно измениться, и в настоящее время преподаватели обсуждают, как могут выглядеть эти перемены. Будет ли студентам разрешено использовать Copilot и если да, то в каких целях? Как Copilot может помочь студентам при обучении? И как теперь будут выглядеть задания по программированию?
- *Качество кода*. Мы должны быть осторожны и не доверять Copilot, особенно если речь идет о конфиденциальном ПО или коде, который должен быть защищен. Например, код, написанный для медицинских устройств, или код, обрабатывающий конфиденциальные данные пользователей, всегда должен быть тщательно изучен. Есть соблазн попросить Copilot предоставить код, восхититься результатом выдачи и принять его без проверки. Но этот код может быть совершенно неправильным. В книге мы будем работать над кодом, который не будет использоваться в промышленном масштабе, поэтому мы не станем беспокоиться о последствиях более широкого использования этого кода, а сосредоточимся на получении примеров корректного кода. Мы начнем строить фундамент, который будет помогать вам самостоятельно определять корректность кода.
- *Безопасность кода*. Как и в случае с качеством кода, при получении кода от Copilot абсолютно не гарантируется и безопасность кода. Например, если мы работаем с пользовательскими данными, то получить код от Copilot недостаточно. Нам потребуется провести аудит безопасности и иметь опыт, позволяющий гарантировать, что код действительно безвреден. Хотя опять же мы не будем использовать код из Copilot в реальных сценариях. Поэтому в книге не будем слишком заострять внимание на проблемах безопасности.
- *Не эксперт*. Одним из признаков эксперта является осознание того, что он знает и, что не менее важно, чего он не знает. Эксперты часто могут сказать, насколько они уверены в своем ответе; и если они недостаточно уверены, то будут учиться дальше, пока не убедятся, что знают. Copilot и в более общем смысле LLM не делают этого. Вы задаете им вопрос, и они отвечают, просто

и понятно. При необходимости они будут заниматься болтовней: смешивать частички правды с кусочками лжи в правдоподобно звучащий, но в целом бессмысленный ответ. Например, мы видели, как LLM фабриковали некрологи для живых людей, что в целом не имеет никакого смысла, но при этом некрологи всегда содержат элементы правды о жизни этих людей. На вопрос о том, почему абак может выполнять математические вычисления быстрее компьютера, LLM придумывали разные ответы, в том числе о том, что абак — механический прибор и поэтому обязательно самый быстрый. В настоящее время работа в этой области продолжается для того, чтобы LLM могли просто сказать: «Извините, нет, я этого не знаю», но мы еще не достигли такого уровня знания. ИИ-системы не знают того, чего не знают, а это значит, что за ними нужен надзор.

- *Предвзятость*. LLM будут воспроизводить искажения, которые присутствуют в данных, на которых они были обучены. Если вы попросите Copilot создать список имен, то он будет генерировать преимущественно английские имена. Если вы попросите создать график, то Copilot может выдать график, который не учитывает различия в восприятии людей. А если вы попросите создать код, то он выдаст код в стиле, в котором пишут подавляющее большинство программистов. (В конце концов, эти люди написали бóльшую часть всего существующего в мире кода, и Copilot всего лишь обучается на нем.) Компьютерные науки и программная инженерия уже давно испытывают недостаток разнообразия. Мы не можем позволить себе и дальше подавлять разнообразие, и нам необходимо обратить эту тенденцию вспять. Мы должны впустить в систему больше людей и позволить им выражать себя по-своему. Вопрос о том, как все это будет осуществляться с помощью инструментов наподобие Copilot, в настоящее время находится в стадии разработки и имеет решающее значение для будущего программирования. Тем не менее мы считаем, что Copilot способен улучшить разнообразие кода, если снизить барьеры входа в эту область.

1.6. НЕОБХОДИМЫЕ НАВЫКИ

Если Copilot может написать код, объяснить его и исправить в нем ошибки, то, может, на этом и все? Можем ли мы просто говорить Copilot, что делать, и наслаждаться собственной крутизной?

Нет. Одни навыки, на которые полагаются программисты (например, написание правильного синтаксиса), перестанут иметь решающее значение. Но другие по-прежнему останутся очень важными. Например, вы не можете поставить перед Copilot огромную задачу типа: «Сделай видеоигру. О, и сделай ее веселой». Copilot потерпит неудачу. Вместо этого вам нужно разбить такую большую задачу на более мелкие, с которыми Copilot может помочь справиться. Но как разбить задачу, как именно это сделать правильно? Оказывается, это не так просто. Людям необходимо развивать этот ключевой навык, и мы будем учить ему на протяжении всей книги.

Верите ли вы в это или нет, но другие навыки могут стать еще более значимыми при работе с Copilot. Тестирование всегда было важнейшей задачей при написании работающего кода. Мы многое знаем о тестировании кода, написанного людьми, поскольку знаем, где искать типичные проблемы. Мы знаем, что люди часто допускают ошибки в программировании на границах значений. Например, если бы мы написали программу для умножения двух чисел, то, скорее всего, получили бы большинство верных значений, но, возможно, не получили бы, если бы один из множителей был равен 0. А как насчет кода, написанного искусственным интеллектом, где среди 20 строк безупречного кода может скрываться одна, настолько абсурдная, что мы, скорее всего, никак не ожидаем ее увидеть? У нас пока нет такого опыта, и нам нужно тестировать еще более тщательно, чем раньше.

Наконец, некоторые необходимые навыки являются совершенно новыми. Главный из них называется *инженерией подсказок* — навык, благодаря которому мы знаем, как указать Copilot, что делать. Прося Copilot написать код, мы используем *подсказки*, чтобы сделать запрос. Это правда, что мы можем использовать человеческий язык, чтобы написать эту подсказку и попросить то, чего мы хотим. Но этого недостаточно. Нам нужно быть очень точными, если мы хотим, чтобы у Copilot был хоть какой-то шанс совершить верные действия. И даже если мы будем точны, он все равно может поступить неправильно. В этом случае нам нужно сначала определить, что Copilot действительно совершил ошибку, а затем исправить наше описание, чтобы подтолкнуть его в правильном направлении. По нашему опыту, незначительные на первый взгляд изменения в подсказке могут оказать огромное влияние на результаты работы Copilot.

В книге мы научим вас всем этим навыкам.

1.7. ОПАСЕНИЯ ОБЩЕСТВА ПО ПОВОДУ ИИ-ПОМОЩНИКОВ ПО ПРОГРАММИРОВАНИЮ

В обществе сейчас царит неопределенность по поводу ИИ-помощников, подобных Copilot. Мы решили завершить главу несколькими вопросами и нашими текущими ответами. Возможно, вы и сами задумывались над некоторыми из этих вопросов. Может быть, наши ответы окажутся до смешного неправильными, но они отражают сегодняшние наши мысли — нас, как двух профессоров и исследователей, посвятивших свою карьеру преподаванию программирования.

Вопрос. Уменьшится ли количество рабочих мест в сфере технологий и программирования теперь, когда у нас есть Copilot?

Ответ. Скорее всего, нет. Чего мы действительно ожидаем, так это изменения характера этих профессий. Например, мы видим, что Copilot может помочь в решении

многих задач, с которыми обычно сталкиваются программисты начального уровня. Это не означает, что такие программисты окажутся невостребованными. Речь идет лишь о том, что эта профессия изменится, поскольку программисты смогут делать больше, используя все более сложные инструменты.

В. Подавит ли Copilot человеческое творчество? Будет ли он просто продолжать бегать по кругу и перерабатывать код, который уже написали люди, ограничивая внедрение новых идей?

О. Мы считаем, что нет. Copilot помогает нам работать на более высоком уровне, дальше от машинного кода, ассемблера или кода Python. Ученые-компьютерщики используют термин *«абстракция»* для обозначения степени, в которой мы можем отвлечься от низкоуровневых деталей компьютеров. Абстрагирование происходит с момента зарождения компьютерной науки. И похоже, мы не страдаем от этого. Напротив, абстракция позволяет игнорировать проблемы, которые уже решены, и сосредоточиться на решении новых и все более широких задач. Действительно, именно появление более совершенных языков программирования способствовало появлению более качественного программного обеспечения: поиск Google, корзины покупателя Amazon и macOS не были написаны (и, скорее всего, не могли быть написаны), когда у нас был только ассемблер!

В. Я постоянно слышу о ChatGPT. Что это такое? Это то же самое, что и Copilot?

О. Это не то же самое, что Copilot, но построено на той же технологии. Вместо того чтобы сосредотачиваться на коде, ChatGPT фокусируется на знаниях в целом. И в результате он стал выполнять более широкий спектр задач, чем Copilot. Например, он может отвечать на вопросы, писать эссе и даже успешно сдавать экзамен на степень MBA в Уортоне [7]. В результате образование должно измениться: мы не можем допустить, чтобы люди с помощью ChatGPT получали MBA! Могут измениться и способы времяпрепровождения. Будут ли люди продолжать писать книги и если да, то как? Захотят ли люди читать книги, зная, что они частично или полностью написаны искусственным интеллектом? Это отразится на всех отраслях, в том числе финансах, здравоохранении и издательском деле [8]. В то же время сейчас царит шумиха, поэтому бывает трудно отделить правду от вымысла. Эта проблема усугубляется тем, что никто не знает, что произойдет в долгосрочной перспективе. Есть старая поговорка, придуманная Роем Амарой (известная как «закон Амары»), которая гласит: «Мы склонны переоценивать эффект технологии в краткосрочной перспективе и недооценивать в долгосрочной». Поэтому мы должны делать все возможное, чтобы быть в курсе происходящих дискуссий и успевать адаптироваться.

В следующей главе мы дадим вам возможность использовать Copilot на вашем компьютере, чтобы вы могли приступить к работе с ПО для написания текстов.