Leo Porter
Daniel Zingaro

Programuj ze sztuczną inteligencją

Twórz kod w Pythonie z wykorzystaniem GitHub Copilot i ChatGPT

Wydanie II

Helion

Tytuł oryginału: Learn AI-Assisted Python Programming: With GitHub Copilot and ChatGPT, 2nd Edition

Tłumaczenie: Radosław Meryk

ISBN: 978-83-289-3048-3

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/przes2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
https://ftp.helion.pl/przyklady/przes2.zip

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

1 Programowanie wspomagane sztuczną inteligencją z GitHub Copilot

W tym rozdziale:

- Jak asystenty AI zmieniają sposób nauki początkujących programistów
- Dlaczego programowanie już nigdy nie będzie takie samo?
- Jak działają asystenty AI, na przykładzie GitHub Copilot
- Potencjalne zagrożenia związane z programowaniem wspomaganym przez sztuczną inteligencję

Programowanie komputerów przez długi czas było domeną profesjonalistów posiadających specjalistyczne wykształcenie i zaawansowane umiejętności. W końcu chcemy, aby aplikacje obsługujące banki, telefony, samochody i inne urządzenia działały bezbłędnie za każdym razem! Jednak tak jak wielkie komputery zajmujące całe pomieszczenia, wykorzystujące stosy kart perforowanych i kilometry taśm magnetycznych zostały zastąpione przez nowoczesne urządzenia,

tak języki oraz narzędzia programistyczne stały się łatwiejsze w użyciu. Teraz, dzięki narzędziom sztucznej inteligencji takim jak ChatGPT, programowanie komputerów staje się dostępne dla prawie każdego. Chcemy pomóc Ci otworzyć te drzwi!

Naucz się programować, a będziesz w stanie podejmować nowe zadania, tworzyć własne gry komputerowe i wykorzystywać komputer do usprawnienia swojej pracy. W tej książce pokażemy Ci, jak pisać własne programy komputerowe z użyciem ChatGPT i GitHub Copilot. Po drodze poznasz podstawy Pythona, jednego z najpopularniejszych języków programowania. Zdobędziesz umiejętności, które pozwolą Ci tworzyć własne rozwiązania i wykorzystywać możliwości sztucznej inteligencji w codziennej pracy.

1.1. Doskonalenie komunikacji z komputerami

Zacznijmy od prostego zadania — poprosimy komputer o policzenie od 0 do 9. Jeszcze kilkadziesiąt lat temu podręcznik programowania wymagałby od Ciebie nauczenia się czytania i rozumienia następującego kodu:

```
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
```

Cieszymy się, że już nie programujemy w ten sposób. Ten potworek został napisany w języku asemblera, czyli w języku programowania niskiego poziomu. Jak widać, języki niskiego poziomu nie są łatwe do czytania i pisania dla ludzi. Zostały stworzone z myślą o komputerach, a nie o ludziach.

Nikt nie chce pisać programów w ten sposób, ale w przeszłości czasami było to konieczne. Programiści mogli używać tego podejścia, aby dokładnie określić, co komputer ma zrobić, aż do poziomu pojedynczych instrukcji. Taki poziom kontroli był potrzebny, aby wycisnąć maksimum wydajności z ówczesnych, mało wydajnych komputerów. Na przykład najbardziej kluczowe pod względem szybkości fragmenty gier komputerowych z lat 90., takich jak Doom i Quake, były napisane w języku asemblera, podobnym do wcześniejszego przykładu. Bez tego stworzenie tych gier nie byłoby możliwe.

1.1.1. Jak to nieco ułatwić?

Pójdźmy dalej. Oto bardziej współczesny program komputerowy, który również wyświetla liczby.

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

Ten kod jest napisany w języku Python, który jest obecnie bardzo popularny wśród programistów. W przeciwieństwie do języka asemblera, który jest językiem niskiego poziomu, Python jest uważany za język wysokiego poziomu, ponieważ jest znacznie bliższy językowi naturalnemu. Nawet jeśli nie znasz jeszcze kodu Pythona, możesz się domyślić, co ten program próbuje zrobić. Pierwszy wiersz wygląda, jakby robił coś z zakresem liczb od 0 do 9. Drugi wiersz coś wypisuje. Nietrudno dojść do przekonania, że ten program, podobnie jak poprzedni przykład w asemblerze, ma za zadanie wypisać liczby od 0 do 9. Niestety, coś poszło nie tak i program wypisuje liczby tylko od 0 do 8.

Chociaż ten kod jest bliższy językowi naturalnemu, to nie jest to język angielski. To język programowania, który podobnie jak język asemblera, ma swoje określone reguły. Tak jak w poprzednim przykładzie, niezrozumienie szczegółów tych reguł może prowadzić do błędów w programie. Jeśli jesteś ciekawy, to nieporozumienie dotyczyło funkcji `range` — która zatrzymuje się o jeden przed drugą z podanych liczb, więc nie uwzględnia liczby 9. Gdybyś chciał uzyskać zakres od 0 do 9, musiałbyś użyć `range(0,10)`.

Świętym Graalem komunikacji z komputerem jest możliwość porozumiewania się z nim w języku naturalnym, takim jak język angielski czy język polski. Przez ostatnie 80 lat rozmawialiśmy z komputerami za pomocą różnych języków programowania nie dlatego, że chcieliśmy, ale dlatego, że musieliśmy. Komputery po prostu nie były wystarczająco mocne, aby poradzić sobie z niuansami i osobliwościami języka naturalnego. Języki programowania ewoluowały — od najeżonego symbolami języka asemblera do, na przykład, Pythona — ale wciąż pozostawały językami komputerowymi, a nie naturalnymi. To się jednak zmienia.

1.1.2. Znaczne ułatwienie

Dzięki asystentowi AI możemy teraz poprosić o to, czego potrzebujemy, używając języka naturalnego, a komputer wygeneruje odpowiedni kod. Aby uzyskać poprawny program w Pythonie, który wypisze liczby od 0 do 9, możemy zwrócić się do naszego asystenta AI (Copilota) w zwykłym języku polskim, na przykład w ten sposób:

```
# Wypisz liczby od 0 do 9
```

Copilot może odpowiedzieć na ten prompt przez wygenerowanie, na przykład, takiego kodu:

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

W przeciwieństwie do przykładu, który pokazałem wcześniej, ten fragment kodu w Pythonie naprawdę działa!

Asystenty programistyczne wykorzystujące sztuczną inteligencję mogą pomóc w pisaniu kodu. W tej książce nauczysz się, jak używać narzędzia Copilot do tworzenia kodu. Będziesz formułować swoje wymagania w języku polskim, a w odpowiedzi otrzymasz gotowy kod w Pythonie.

Co więcej, będziesz mógł używać Copilota jako integralnej części swojego procesu pracy. Bez takich narzędzi jak Copilot programiści zazwyczaj mają otwarte dwa okna: jedno do pisania kodu, a drugie do szukania w Google, jak ten kod napisać. To drugie okno zawiera wyniki wyszukiwania Google, dokumentację Pythona lub fora, na których programiści dyskutują o tym, jak napisać kod rozwiązujący konkretny problem. Programiści często kopiują fragmenty kodu z różnych źródeł do swoich projektów, a następnie dostosowują je do specyfiki swojego zadania. Stało się to dla nich codziennością, ale jak łatwo sobie wyobrazić, takie podejście jest bardzo nieefektywne. Według niektórych szacunków nawet 35% czasu pracy programistów pochłania poszukiwanie kodu [1], a znaczna część znalezionych fragmentów nie nadaje się do bezpośredniego użycia. Copilot znacząco usprawnia ten proces.

1.2. O technologii

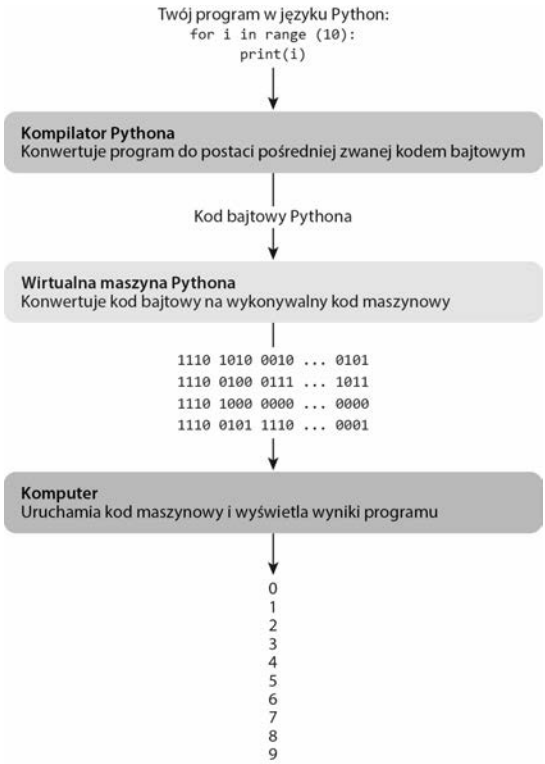
W tej książce będziemy korzystać z dwóch głównych technologii: Pythona i GitHub Copilot. Python to język programowania, którego użyjemy, a GitHub Copilot to asystent oparty na sztucznej inteligencji, który pomoże nam w pracy z kodem Pythona.

1.2.1. Python, Twój język programowania

Jak wspominaliśmy, Python jest językiem programowania, czyli sposobem komunikacji z komputerem. Używa się go do tworzenia różnorodnych programów, które wykonują przydatne zadania, takie jak analiza danych, gry, interaktywne strony internetowe, wizualizacje, aplikacje do organizacji plików, automatyzacja rutynowych czynności i wiele innych.

Istnieje wiele innych języków programowania, takich jak Java, C++ czy Rust. Copilot działa również z nimi, ale w momencie pisania tej książki szczególnie dobrze radzi sobie z Pythonem. Kod w Pythonie jest znacznie łatwiejszy do napisania w porównaniu z wieloma innymi językami (zwłaszcza kodem w asemblerze). Co ważniejsze, Python jest łatwy do *czytania*. Ostatecznie to nie my będziemy pisać kod w Pythonie — zrobi to za nas asystent AI!

Komputery nie potrafią bezpośrednio czytać i uruchamiać kodu napisanego w Pythonie. Jedyne, co komputery rozumieją, to tak zwany **kod maszynowy**, który wygląda jeszcze bardziej skomplikowanie niż kod asemblera, ponieważ jest binarną reprezentacją kodu asemblera (tak, to po prostu ciąg zer i jedynek!). Za kulisami, zanim Twój komputer uruchomi program napisany w Pythonie, musi przekształcić go na kod maszynowy, co zostało zobrazowane na rysunku 1.1.



Rysunek 1.1. *Zanim zobaczysz wynik na ekranie, Twój program w Pythonie przechodzi przez kilka etapów*

Obecnie nikt już nie pisze kodu od podstaw w języku maszynowym komputerów. Programiści wybierają język najwygodniejszy do konkretnego zadania i korzystają z oprogramowania wspomagającego pisanie, uruchamianie i debugowanie kodu, zwanego zintegrowanym środowiskiem programistycznym (IDE). W tej książce jako naszego IDE będziemy używać Visual Studio Code (VS Code), ponieważ świetnie współpracuje ono z GitHub Copilot.

1.2.2. GitHub Copilot, Twój asystent AI

Czym jest asystent AI? Asystent AI to program komputerowy, który pomaga w wykonywaniu różnych zadań. Być może masz w domu urządzenie Amazon Alexa lub iPhone'a z Siri — to właśnie są asystenty AI. Pomagają zamawiać zakupy, sprawdzać pogodę lub potwierdzić, że aktorka grająca Bellatrix w filmach o *Harrym Potterze* rzeczywiście wystąpiła również w *Podziemnym kręgu*. Asystent AI to po prostu program komputerowy, który reaguje na typowe dane wejściowe, takie jak mowa i tekst, i udziela odpowiedzi w sposób zbliżony do ludzkiego.

Copilot to asystent AI o konkretnym zadaniu: przekształca język naturalny w programy komputerowe (a także, jak się wkrótce przekonamy, wykonuje wiele innych funkcji). Istnieją także inne narzędzia podobne do Copilota, takie jak Amazon Q Developer, Tabnine czy Ghostwriter. W tej książce zdecydowaliśmy się na Copilota ze względu na jakość generowanego kodu, stabilność działania (nigdy nam się nie zawiesił!) oraz nasze osobiste preferencje. Zachęcamy jednak do wypróbowania innych narzędzi, gdy poczujesz się na to gotowy.

1.2.3. Jak działa Copilot za kulisami — w 30 sekund

Copilota można postrzegać jako warstwę pośrednią między Tobą a programem komputerowym, który tworzysz. Zamiast bezpośrednio pisać kod w Pythonie, po prostu opisujesz słowami program, który chcesz stworzyć — polecenie dla asystenta to tzw. **prompt**. W odpowiedzi Copilot generuje dla Ciebie gotowy kod.

Sercem Copilota jest zaawansowany program komputerowy zwany **dużym modelem językowym** (LLM). Model LLM przechowuje informacje o relacjach między słowami, w tym o tym, które słowa mają sens w określonych kontekstach, i wykorzystuje tę wiedzę do przewidywania najlepszej sekwencji słów w odpowiedzi na zadane polecenie.

Wyobraź sobie, że zapytalibyśmy Cię, jakie słowo powinno znaleźć się w następującym zdaniu: „Nieznajomy otworzył _____". Jest wiele słów, które mogłyby pasować, takich jak „drzwi", „pudełko" czy „książkę", ale są też słowa, które zupełnie tu nie pasują, jak „on", „zrobił" czy „zamknęła". Model LLM bierze pod uwagę aktualny kontekst słów, aby wygenerować kolejne, i powtarza ten proces, aż ukończy zadanie. Robi to w sposób *niedeterministyczny*, co oznacza,

że jego decyzje są do pewnego stopnia losowe. Innymi słowy, jeśli poprosisz go o uzupełnienie tego słowa, czasem może zaproponować „drzwi", a innym razem „list". To właśnie dlatego, gdy prosisz Copilota o wygenerowanie kodu, za każdym razem możesz otrzymać różne odpowiedzi.

Warto również zwrócić uwagę, że nie wspomnieliśmy nic o tym, że Copilot rozumie, co robi. Po prostu wykorzystuje bieżący kontekst do dalszego pisania kodu. Pamiętaj o tym podczas swojej pracy: tylko my wiemy, czy wygenerowany kod faktycznie realizuje nasze zamierzenia. Bardzo często tak jest, ale zawsze warto zachować zdrowy sceptycyzm. Sposób, w jaki Copilot przechodzi od polecenia do gotowego programu, pokazano na rysunku 1.2.

Rysunek 1.2. *Od promptu do programu utworzonego za pomocą Copilota*

Mógłbyś się zastanawiać, dlaczego Copilot pisze kod w Pythonie, a nie bezpośrednio w kodzie maszynowym. Czy Python nie jest teraz zbędnym pośrednim krokiem? Otóż nie, a powód jest taki, że Copilot popełnia błędy. A jeśli popełnia błędy, które musimy naprawiać, to o wiele łatwiej jest to zrobić w Pythonie niż w kodzie maszynowym.

W gruncie rzeczy prawie nikt nie sprawdza, czy kod maszynowy wygenerowany z Pythona jest poprawny. Wynika to częściowo z determinizmu specyfikacji języka Python. Można sobie wyobrazić przyszłość, w której rozmowy z asystentami AI będą tak precyzyjne, że przeglądanie kodu w Pythonie stanie się zbędne, ale do tego jeszcze daleka droga.

1.3. Jak Copilot zmienia sposób nauki programowania?

W przeszłości osoby uczące się programowania poświęcały większość czasu na opanowanie składni i podstawowej struktury programów. Składnia odnosi się do *symboli* i słów, które są poprawne w danym języku programowania. Programiści

musieli pisać całą składnię programu od podstaw, znak po znaku, wiersz po wierszu. Nauka programowania do poziomu pozwalającego na napisanie nawet najprostszych programów wymagała tygodni lub miesięcy. Obecnie Copilot potrafi natychmiast tworzyć te same podstawowe programy i generuje kod, który jest niemal zawsze poprawny pod względem składni i struktury. Jak dowiesz się z dalszej części książki, nadal trzeba weryfikować poprawność tego kodu, ponieważ Copilot może popełniać błędy. Jednak nie musimy już pisać go od zera. Uważamy, że Copilot i podobne narzędzia oznaczają koniec dotychczasowego sposobu nauki programowania.

Jeśli interesujesz się nauką programowania, nie musisz już zmagać się z zawiłościami składni, dokładnym zrozumieniem, jak wywoływać konkretne funkcje Pythona, ani z innymi pojęciami, które kiedyś były niezbędne do pisania kodu. Oczywiście w tej książce omówimy te zagadnienia, ale nie po to, abyś musiał udowadniać swoją wiedzę, pisząc od podstaw kod, który łatwo może wygenerować Copilot. Poznamy te pojęcia, ponieważ pomagają w rozwiązywaniu istotnych problemów i skutecznej współpracy z Copilotem. Dzięki fundamentalnym zmianom, jakie wprowadza asystent AI w nauce programowania, będziesz mógł szybciej tworzyć większe i doskonalsze oprogramowanie.

1.4. Co jeszcze może dla nas zrobić Copilot?

Jak już się przekonałeś, możemy użyć Copilota do pisania kodu w Pythonie na podstawie opisu w języku naturalnym. Można więc powiedzieć, że Copilot przyjmuje opis w języku naturalnym i zwraca kod zgodny ze składnią Pythona. To duża zaleta, ponieważ nauka składni programowania była w przeszłości jedną z głównych przeszkód dla początkujących programistów. Jakiego rodzaju nawiasu — kwadratowego, okrągłego czy klamrowego — powinienem użyć w tym miejscu? Czy potrzebuję wcięcia? W jakiej kolejności należy zapisać te elementy: najpierw x, a potem y, czy odwrotnie?

Takie pytania są powszechne i, bądźmy szczerzy, niezbyt interesujące. Kogo to obchodzi, skoro chcemy po prostu napisać program, który coś zrobi? Copilot uwalnia nas od konieczności wykonywania żmudnej pracy związanej ze składnią. Uważamy to za ważny krok w kierunku ułatwienia większej liczbie osób skutecznego pisania programów i z niecierpliwością czekamy na dzień, gdy ta sztuczna bariera zostanie całkowicie usunięta. Na razie wciąż potrzebujemy składni Pythona, ale przynajmniej mamy do pomocy Copilota.

Ale to nie wszystko, co potrafi Copilot. Oto kilka powiązanych — i nie mniej ważnych — zadań, w których Copilot może nam pomóc:

- *Wyjaśnianie kodu*. Gdy Copilot generuje kod w Pythonie, musisz ocenić, czy ten kod faktycznie robi to, czego od niego oczekujesz. Jak wspominaliśmy wcześniej, Copilot może popełniać błędy. Chociaż nie zamierzamy uczyć Cię wszystkich niuansów działania Pythona (to stary model programowania), pokażemy Ci, jak czytać kod Pythona, aby zrozumieć jego ogólne działanie. Skorzystamy również z funkcji Copilota, która wyjaśnia kod w języku naturalnym. Po przeczytaniu tej książki i naszych objaśnień nadal będziesz potrzebował Copilota, aby zrozumieć kolejny fragment skomplikowanego kodu, który otrzymasz.
- *Zwiększanie czytelności kodu*. Jest wiele sposobów na napisanie kodu realizującego to samo zadanie. Niektóre z nich są łatwiejsze do zrozumienia niż inne. Możesz poprosić Copilota o przeorganizowanie Twojego kodu, aby praca z nim była łatwiejsza. Kod, który jest bardziej czytelny, często jest również łatwiejszy do rozbudowy lub naprawy, gdy zajdzie taka potrzeba.
- *Naprawianie błędów*. **Błąd** (ang. *bug*) to pomyłka popełniona podczas pisania programu, która może spowodować jego nieprawidłowe działanie. Czasami Twój kod w Pythonie prawie działa lub działa prawie zawsze, ale zawodzi w jednym konkretnym przypadku. Jeśli słuchałeś rozmów programistów, mogłeś usłyszeć typową historię, w której programista spędził godziny, by w końcu usunąć jeden znak =, który powodował awarię programu. To nie są przyjemnie spędzone godziny! W takich sytuacjach możesz skorzystać z funkcji Copilot, która pomaga automatycznie znaleźć i naprawić błąd w programie.
- *Wyjaśnianie błędów*. Gdy Twój kod nie działa poprawnie, często otrzymujesz raport o błędzie ze środowiska uruchomieniowego Pythona. Te komunikaty czasami są dość niejasne, ale Copilot może pomóc Ci je zinterpretować i zasugerować, jak naprawić problem. Asystent wyjaśnia znaczenie komunikatów o błędach i proponuje możliwe rozwiązania, co znacznie ułatwia debugowanie i poprawianie kodu.
- *Odkrywanie bibliotek Pythona*. Python to dojrzały język programowania z wieloma modułami (bibliotekami), które ułatwiają realizację konkretnych zadań, takich jak analiza danych, tworzenie gier czy obsługa różnych formatów plików graficznych. Krótka rozmowa z Copilotem często pozwala znaleźć moduły, które usprawnią pracę i dostarczą przykładów, dzięki którym szybko rozpoczniesz korzystanie z nowej biblioteki.

1.5. Zagrożenia i wyzwania związane z korzystaniem z Copilota

Skoro już wszyscy jesteśmy podekscytowani możliwościami wykorzystania Copilota do pisania kodu, musimy porozmawiać o zagrożeniach związanych z korzystaniem z asystentów opartych na sztucznej inteligencji (więcej informacji na ten temat można znaleźć w pozycjach [2] i [3] bibliografii):

- *Prawa autorskie* — Copilot nauczył się programować na podstawie kodu napisanego przez ludzi (w kontekście narzędzi AI takich jak Copilot często usłyszysz określenie „szkolenie"; w tym przypadku szkolenie oznacza po prostu uczenie się). Dokładniej mówiąc, został on przeszkolony na milionach repozytoriów GitHub zawierających kod open source. Zachodzi obawa, że Copilot może „kraść" kod i przekazywać go nam. Z naszego doświadczenia wynika, że Copilot rzadko proponuje obszerne fragmenty cudzego kodu, ale taka możliwość istnieje. Nawet jeśli kod generowany przez Copilota jest wynikiem połączenia i przetworzenia różnych fragmentów kodu innych osób, mogą pojawić się problemy licencyjne. Na przykład: kto jest właścicielem kodu wytworzonego przez Copilota? Obecnie brakuje jednoznacznej odpowiedzi na to pytanie. Zespół Copilota dodaje funkcje, które mają pomóc w tej kwestii; na przykład Copilot może informować, czy wygenerowany kod jest podobny do już istniejącego i jaka licencja obowiązuje dla tego kodu [4]. Samodzielna nauka i eksperymentowanie są wartościowe i do tego zachęcamy, jednak należy zachować ostrożność, jeśli zamierzasz użyć tego kodu do celów wykraczających poza własne projekty. Celowo wyrażamy się tutaj dość ogólnie, ponieważ może minąć trochę czasu, zanim prawo dostosuje się do tej nowej technologii. Najbezpieczniej jest zachować ostrożność, dopóki w społeczeństwie trwa debata.
- *Edukacja*. Jako wykładowcy prowadzący kursy wprowadzające do programowania sami przekonaliśmy się, jak dobrze Copilot radzi sobie z zadaniami, które dotychczas dawaliśmy naszym studentom. W jednym z badań Copilot miał rozwiązać 166 typowych zadań z podstaw programowania [5]. Jak sobie poradził? Za pierwszym podejściem rozwiązał prawie 50% tych problemów. Po dostarczeniu dodatkowych informacji odsetek ten wzrósł do 80%. W obliczu narzędzi takich jak Copilot edukacja musi się zmienić, a wykładowcy obecnie dyskutują nad tym, jak te zmiany mogą wyglądać. Na niektórych uczelniach studenci mogą korzystać z Copilota w nauce i przy wykonywaniu zadań.

Na innych uczelniach Copilot jest niedozwolony w pewnych sytuacjach (np. podczas egzaminów) lub dla niektórych studentów (np. kierunków informatycznych). W wielu szkołach modele LLM pełnią funkcję wirtualnych korepetytorów. W niektórych przypadkach są to standardowe modele LLM, takie jak Copilot czy ChatGPT, a w innych interfejs modelu LLM został zmodyfikowany, aby ograniczyć rodzaj odpowiedzi, jakie otrzymują studenci. Jest jeszcze za wcześnie, by ocenić, jaki wpływ modele LLM będą miały na edukację w dziedzinie informatyki, ale już teraz można zaobserwować pewne trendy.

- *Jakość kodu*. Musimy być ostrożni i nie wolno nam ślepo ufać takim narzędziom jak Copilot, szczególnie w przypadku wrażliwego kodu lub kodu wymagającego wysokiego poziomu bezpieczeństwa. Kod pisany dla urządzeń medycznych czy obsługujący poufne dane użytkowników musi być zawsze dokładnie zrozumiany i przeanalizowany. W pracy z Copilotem łatwo jest ulec pokusie, by przyjąć wygenerowany kod bez głębszej analizy — zwłaszcza jeśli na pierwszy rzut oka wygląda poprawnie. Taki kod może jednak zawierać błędy. W tej książce będziemy pracować z kodem, który nie trafi do produkcji ani nie będzie wykorzystywany na szeroką skalę. Dzięki temu możemy skupić się na poprawności rozwiązań, nie analizując ich wpływu w szerszym kontekście. Zbudujemy też podstawy, które pozwolą Ci samodzielnie oceniać, czy dany kod rzeczywiście działa zgodnie z założeniami.
- *Bezpieczeństwo kodu*. Tak jak nie możemy oczekiwać gwarancji jakości kodu generowanego przez Copilota, tak samo nie mamy pewności co do bezpieczeństwa kodu. Przy pracy z danymi użytkowników samo użycie takiego kodu nie wystarczy — konieczny jest audyt bezpieczeństwa i odpowiednia wiedza, by ocenić, czy kod rzeczywiście spełnia wymagania. Ponieważ w tej książce nie korzystamy z tego kodu w rzeczywistych zastosowaniach, pomijamy kwestie bezpieczeństwa.
- *Brak eksperckiej wiedzy*. Jedną z cech eksperta jest świadomość tego, co wie, a co równie ważne, czego nie wie. Eksperci często potrafią też określić, na ile są pewni swojej odpowiedzi, a jeśli nie są wystarczająco pewni, dalej się uczą do czasu zyskania pewności, że mają odpowiednią wiedzę. Copilot i ogólnie modele językowe tego nie robią. Zadajesz im pytanie, a one po prostu odpowiadają. Czasami też ulegają halucynacjom: łączą elementy prawdy z nonsensem w odpowiedź, która brzmi wiarygodnie, ale ogólnie jest bezsensowna. Na przykład widzieliśmy, jak modele językowe wymyślały nekrologi dla żyjących osób, co jest zupełnie nielogiczne, choć te „nekrologi” zawierały prawdziwe elementy z życia tych ludzi. Zdarza się też, że model udziela odpowiedzi z pełnym przekonaniem, nawet jeśli są one niedorzeczne — jak wtedy, gdy

zapytany, dlaczego liczydło działa szybciej niż komputer, tłumaczył to rzekomo mechaniczną naturą liczydła, która ma czynić je „z definicji" najszybszym narzędziem. Trwają prace nad tym, by modele te potrafiły przyznać: „Przepraszam, nie znam odpowiedzi na to pytanie", ale jeszcze nie osiągnęliśmy tego etapu. Modele nie wiedzą, czego nie wiedzą, co oznacza, że wymagają nadzoru.

- *Stronniczość*. Modele językowe powielają trendy występujące w danych, na których zostały przeszkolone. Jeśli poprosisz Copilota o wygenerowanie listy imion, prawdopodobnie otrzymasz głównie imiona angielskie. Jeśli poprosisz o wykres, może nie uwzględnić różnic percepcyjnych między użytkownikami. A jeśli poprosisz o kod, Copilot może wygenerować go w stylu typowym dla osób, które przez lata dominowały w środowisku programistycznym — bo właśnie z takich przykładów się uczył. Informatyka i inżynieria oprogramowania od lat cierpią na brak różnorodności. Nie możemy sobie pozwolić, by ten problem się pogłębiał — trzeba odwrócić ten trend. Trzeba otworzyć branżę na nowe osoby i dać im przestrzeń do wyrażania się na własnych zasadach. To, w jaki sposób narzędzia takie jak Copilot poradzą sobie z tym wyzwaniem, wciąż pozostaje kwestią otwartą — i będzie miało kluczowe znaczenie dla przyszłości programowania. Wierzymy jednak, że Copilot ma potencjał, by wspierać różnorodność, ponieważ obniża bariery wejścia do zawodu.

1.6. Potrzebne umiejętności

Jeśli Copilot potrafi pisać kod, wyjaśniać go i naprawiać w nim błędy, to czy programiści są już niepotrzebni? Czy wystarczy powiedzieć Copilotowi, co ma zrobić, po czym zacząć świętować swoją genialność?

Nie. Po pierwsze, Copilot może popełniać błędy. Kod, który generuje, może być poprawny składniowo, ale czasami nie spełnia naszych oczekiwań. Musimy być czujni, aby wychwycić te pomyłki. Po drugie, choć niektóre umiejętności programistyczne, jak pisanie poprawnej składni, mogą stracić na znaczeniu, inne pozostaną kluczowe. Na przykład nie można powierzyć Copilotowi ogromnego zadania typu „Stwórz grę wideo i spraw, by była interesująca". Copilot sobie z tym nie poradzi. Zamiast tego musimy podzielić taki duży problem na mniejsze zadania, w których Copilot może nam pomóc. Jak to zrobić? Okazuje się, że nie jest to łatwe. Osoby muszą rozwinąć tę kluczową umiejętność podczas pracy z narzędziami takimi jak Copilot, i właśnie tej umiejętności uczymy w tej książce.

Inne umiejętności mogą zyskać na znaczeniu wraz z pojawieniem się Copilota, choć może to brzmieć zaskakująco. Testowanie kodu zawsze było kluczowym zadaniem w tworzeniu wysokiej jakości oprogramowania. Mamy szeroką wiedzę na temat testowania kodu napisanego przez programistów, ponieważ

znamy typowe miejsca, w których pojawiają się problemy. Wiemy, że błędy programistyczne często występują na granicznych wartościach. Na przykład, pisząc program do mnożenia dwóch liczb, prawdopodobnie uzyskamy poprawne wyniki dla większości przypadków, lecz możemy napotkać problem, gdy jedna z liczb wynosi 0. Jak jednak podejść do kodu stworzonego przez sztuczną inteligencję, gdzie 20 wierszy bezbłędnego kodu może skrywać jeden absurdalny, którego byśmy się tam nie spodziewali? Nie mamy w tym zakresie doświadczenia. Musimy testować jeszcze bardziej skrupulatnie niż dotychczas.

Musimy również wiedzieć, jak naprawiać błędy, gdy kod nie działa poprawnie. Ten proces nazywa się **debugowaniem** i nadal jest niezbędny, szczególnie gdy Copilot generuje kod, który jest bliski poprawności. Umiejętność ta jest kluczowa, zwłaszcza gdy automatycznie wygenerowany kod wymaga dopracowania.

Niektóre z wymaganych umiejętności są zupełnie nowe. Najważniejszą z nich jest tzw. **inżynieria promptów** (ang. *prompt engineering*), czyli sztuka formułowania poleceń dla narzędzi AI, takich jak Copilot. Jak już wspomniano, gdy prosimy Copilota o wygenerowanie kodu, robimy to za pomocą polecenia, które stanowi formę naszego żądania. Chociaż możemy użyć języka naturalnego do sformułowania tego polecenia, nie jest to wystarczające. Musimy być bardzo precyzyjni, aby Copilot miał szansę wykonać zadanie poprawnie. Nawet przy precyzyjnych instrukcjach Copilot może popełnić błąd. W takim przypadku konieczne jest najpierw zidentyfikowanie błędu, a następnie doprecyzowanie opisu, aby skierować działanie Copilota we właściwym kierunku. Z naszego doświadczenia wynika, że pozornie drobne zmiany w poleceniu mogą znacząco wpłynąć na wynik generowany przez Copilota. W tej książce nauczymy Cię wszystkich tych umiejętności.

1.7. Obawy społeczne dotyczące asystentów programistycznych AI takich jak Copilot

W społeczeństwie panuje obecnie niepewność co do asystentów programistycznych opartych na sztucznej inteligencji, takich jak Copilot. Postanowiliśmy zakończyć ten rozdział kilkoma pytaniami i naszymi obecnymi odpowiedziami. Być może sami zastanawialiście się nad niektórymi z tych kwestii! Nasze odpowiedzi mogą okazać się z czasem zabawnie nietrafione, ale odzwierciedlają aktualne przemyślenia dwóch profesorów i badaczy, którzy poświęcili swoje kariery nauczaniu programowania:

P: Czy teraz, gdy mamy Copilota, będzie mniej miejsc pracy w branży IT i programowaniu?

O: Prawdopodobnie nie. To, czego możemy się spodziewać, to zmiana charakteru tych stanowisk. Na przykład: uważamy, że Copilot może pomóc w wielu zadaniach typowo związanych z pracą początkujących programistów. Nie oznacza to, że stanowiska dla juniorów znikną, ale raczej że się zmienią, ponieważ programiści, dzięki coraz bardziej zaawansowanym narzędziom, będą w stanie zrobić więcej. Wraz z rozwojem technologii charakter pracy programistów ewoluuje. Dzięki nowym technologiom mogą oni skupić się na bardziej złożonych i kreatywnych aspektach programowania.

P: Czy Copilot ograniczy ludzką kreatywność? Czy będzie jedynie obracać się wokół tego samego kodu, który już został napisany, przez co będzie coraz mniej nowych pomysłów? Czy sztuczna inteligencja zamknie nas w pętli powielania istniejących rozwiązań i zahamuje innowacje w programowaniu?

O: Podejrzewamy, że nie. Copilot pomaga pracować na wyższym poziomie, bardziej oddalonym od niskopoziomowego kodu maszynowego, asemblera czy nawet Pythona. W informatyce do określenia stopnia, w jakim możemy oderwać się od niskopoziomowych szczegółów komputerów, używamy terminu „abstrakcja". Abstrakcja towarzyszy nam od zarania informatyki i nie wydaje się, aby komukolwiek zaszkodziła. Wręcz przeciwnie — pozwoliła oderwać się od problemów, które już zostały rozwiązane, i skupić na rozwiązywaniu problemów coraz bardziej kompleksowych. W gruncie rzeczy to właśnie pojawienie się lepszych języków programowania umożliwiło tworzenie lepszego oprogramowania. Programy, na których jest oparta wyszukiwarka Google, koszyki zakupowe Amazona czy system macOS nie zostały napisane (i prawdopodobnie nie mogłyby powstać) w czasach, gdy mieliśmy do dyspozycji tylko asembler!

P: Ciągle słyszę o ChatGPT. Co to takiego? Czy to to samo co Copilot?

O: ChatGPT to nie to samo co Copilot, ale opiera się na tej samej technologii. Zamiast jednak koncentrować się na kodzie, skupia się na wiedzy ogólnej. W rezultacie może wspomagać wykonywanie szerszej gamy zadań niż Copilot. Na przykład potrafi odpowiadać na pytania, pisać eseje, a nawet dobrze radzić sobie na egzaminie MBA na Wharton [6]. W związku z tym edukacja będzie musiała się zmienić — nie możemy pozwolić, by ludzie zdobywali tytuły MBA za pomocą ChatGPT! Może to wpłynąć na sposób, w jaki spędzamy czas. Czy ludzie nadal będą pisać książki, a jeśli tak, to w jaki sposób? Czy będą chcieli je czytać, wiedząc, że zostały częściowo lub w całości napisane przez sztuczną inteligencję? Zmiany te wpłyną na różne branże, w tym finanse, opiekę zdrowotną i wydawnictwa [7]. Jednocześnie obecnie mamy do czynienia z niepohamowanym szumem medialnym,

co utrudnia oddzielenie prawdy od fikcji. Problem ten pogłębia fakt, że nikt nie wie, co się wydarzy w dłuższej perspektywie. Istnieje stare powiedzenie Roya Amary, znane jako prawo Amary, które mówi, że mamy tendencję do przeceniania efektu technologii w krótkim okresie i niedoceniania go w długim. Dlatego trzeba uważnie śledzić tę dyskusję, aby móc odpowiednio się dostosować.

W następnym rozdziale nauczysz się korzystać z Copilota na swoim komputerze, co pozwoli Ci szybko rozpocząć tworzenie oprogramowania.

Podsumowanie

- Copilot to asystent wykorzystujący sztuczną inteligencję, który pomaga w wykonywaniu różnych zadań.
- Copilot zmienia sposób, w jaki ludzie wchodzą w interakcję z komputerami, a także sposób, w jaki tworzymy programy.
- Copilot zmienia priorytety umiejętności, które należy rozwijać (mniejszy nacisk na składnię, większy na dekompozycję problemów i testowanie).
- Copilot nie działa w sposób niedeterministyczny. Czasem generuje poprawny kod, a czasem nie. Podczas korzystania z niego trzeba zachować czujność i ostrożność.
- Kwestie praw autorskich do kodu, edukacji, szkolenia zawodowego oraz stronniczości w wynikach Copilota nadal wymagają rozwiązania.

Nie jesteś w stanie konkurować ze sztuczną inteligencją. Zamiast tego sprawdź, jak może wspomagać Twoją pracę.

— **Srihari Sridharan**, Thoughtworks

Wyobraź sobie, że zamiast mozolnego pisania kodu linijka po linijce opisujesz w języku naturalnym, jak ma działać gotowy program. I po chwili Twój ulubiony asystent AI, taki jak GitHub Copilot, generuje dobry, działający kod!

Książka w mistrzowski sposób łączy podstawy programowania ze skutecznym zastosowaniem narzędzi sztucznej inteligencji do tworzenia kodu!

— **Mehran Sahami**, Uniwersytet Stanforda

Dzięki tej książce przekonasz się, że tak wygląda praca programisty przyszłości. W trakcie nauki opanujesz Pythona na tyle, by rozumieć i ulepszać kod, który tworzy Copilot. Dowiesz się, jak z jego użyciem tworzyć gry, narzędzia i inne proste aplikacje. Zdziwisz się, jak szybko możesz przejść od pomysłu do gotowego programu! Znajdziesz tu wszystkie potrzebne instrukcje zapisane przejrzyście krok po kroku: od prostych projektów, takich jak program sprawdzający siłę hasła, po narzędzia automatyzujące żmudne zadania.

W książce:

- **skuteczne prompty, które generują działający kod Pythona**
- **zasady dekompozycji złożonych pomysłów na mniejsze, zrozumiałe dla AI części**
- **zastosowanie AI do testowania i ulepszania kodu**
- **podstawy projektowania programów komputerowych**

Ta książka niewiarygodnie przyspiesza naukę programowania z Copilotem!

— **Austin Z. Henley**, Microsoft

Wartościowa lektura dla każdego, kto szuka niekonwencjonalnego sposobu nauki programowania.

— **Wei Luo**, Deakin University

Leo Porter jest profesorem informatyki na Uniwersytecie Kalifornijskim w San Diego. Był wielokrotnie wyróżniany prestiżowymi nagrodami za działalność badawczą w zakresie dydaktyki w informatyce.

Daniel Zingaro jest adiunktem dydaktycznym na informatyce na Uniwersytecie w Toronto i laureatem nagród za osiągnięcia w nauczaniu. Jego głównym obszarem badawczym jest edukacja informatyczna.

Helion

helion.pl

HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-3048-3

Cena: 89,00 zł