Krzysztof Łos

PYTHON DLA NASTOLATKÓW

Projekty graficzne z Python Turtle

Helion

Redaktor prowadzący: Szymon Szwajger
Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

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

ISBN: 978-83-283-7135-4

Printed in Poland.

SPIS TREŚCI

2
PODSTAWY JĘZYKA PYTHON

Niniejszy rozdział wprowadzi Cię w podstawy programowania w języku Python. Uprzedzając Twoje pytanie, od razu powiem, że wcale nie musisz go znać. Proszę Cię jedynie, byś otworzył umysł i uważnie czytał poniższą treść.

Dokładne i staranne czytanie połączone z analizą kodów źródłowych poskutkuje zrozumieniem zagadnień, które tutaj omówię. Jeśli dołożymy do tego praktykę, czyli pisanie kodu, to bardzo szybko osiągniesz sukces.

Zanim zaczniemy, jeszcze raz podkreślę, że nie da się nauczyć dobrze mówić w języku obcym, jeśli nie będziemy tego praktykować. To samo dotyczy programowania.

Czym właściwie jest komputer?

Na pewno słowo „komputer" jest Ci dobrze znane. Jednak czy zastanawiałeś się, czym właściwie jest urządzenie, którego używasz na co dzień? Pisząc „komputer", mam na myśli nie tylko Twój laptop czy komputer klasy PC (ang. *Personal Computer*), ale również smartfon, który także jest urządzeniem tego typu. Ma procesor, pamięć RAM oraz pamięć nieulotną, w której zapisywane są Twoje dane i programy. Żeby lepiej zrozumieć, czym są współczesne komputery, w tym wspomniane smartfony, musimy przenieść się do początków XX wieku. Wtedy to różne firmy tworzyły maszyny liczące charakteryzujące się skonkretyzowanym przez ich twórcę przeznaczeniem. Niektóre z nich umiały tylko dodawać, dzielić lub wykonywać jeszcze inne operacje matematyczne. Taka sytuacja miała miejsce aż do roku 1937, kiedy to angielski matematyk Alan Turing w swojej pracy opisał maszynę, która nie miała ściśle określonego przeznaczenia. Jej możliwości w kwestii rozwiązywania problemów można było zmieniać za pomocą specjalnych taśm, na których istniała możliwość zapisu algorytmów (zadań, jakie komputer ma wykonać).

Turing zaproponował, aby taśmy były nieskończenie długie, a koncepcję całej maszyny określił mianem maszyny Turinga.

Idąc tym tropem, znane nam dzisiaj komputery to nic innego jak maszyny Turinga. Można je tak nazwać ze względu na różnorakie funkcje, jakie pełnią, jak chociażby to, że możesz użyć swojego telefonu jako dyktafonu za pomocą dedykowanych do tego aplikacji. Przykłady funkcji można by mnożyć prawie bez końca.

Producenci z roku na rok prześcigają się w dostarczaniu nam coraz to nowszych, bezpieczniejszych, prostszych w obsłudze wersji systemów operacyjnych.

Sprzęt jest ważny. Jednak bez odpowiedniego oprogramowania nawet najlepszy komputer będzie tylko zlepkiem tranzystorów. Można śmiało powiedzieć, że rynek maszyn jako takich nie jest wart aż tyle, co rynek oprogramowania. Stwierdzenie to potwierdza sukces, jaki osiągnęła firma Microsoft, produkując system operacyjny MS-DOS, a później Windows.

Dla wyjaśnienia: system operacyjny tworzy środowisko pracy i jest pośrednikiem pomiędzy sprzętem a oprogramowaniem, które na nim zostaje uruchomione.

Współczesne komputery to bardzo skomplikowane twory, zarówno pod względem elektronicznym, jak i programowym. Jednak nie ma co się tym aż tak bardzo przejmować. Działanie komputera można wytłumaczyć w bardzo prosty sposób. Standardowo każdy komputer składa się z procesora, który możemy nazwać mózgiem komputera. Kolejny ważny element w budowie komputera to pamięć RAM (skrót pochodzi od angielskiego *Random Access Memory*). Skrót ten tłumaczymy na język polski jako pamięć swobodnego dostępu. Standardowo każdy komputer składa się z procesora, czyli mózgu, oraz wspomnianej pamięci operacyjnej zwanej RAM. Programy zapisywane są na dysku twardym. Program to ciąg poleceń, jakie ma wykonać za nas komputer. Uruchomiony program ładowany jest z dysku do pamięci RAM. Procesor operuje właśnie na tej pamięci, wykonując nasz program instrukcja po instrukcji. Zawartość pamięci RAM może zostać zapisana na dysku jako wynik działania naszego programu, jednak nie jest to regułą. Program może na przykład wypisać coś na ekranie albo wysłać jakąś informację do urządzenia podpiętego do komputera, takiego jak drukarka.

Czym jest program komputerowy?

Jak pewnie wiesz z lekcji informatyki, program komputerowy to ciąg poleceń zrozumiałych dla komputera. Oznacza to, że programując komputer, wydajemy mu rozkazy, co ma zrobić.

Programy mogą być pisane w różnych językach, od kodu maszynowego (który jest ciągiem zer i jedynek bezpośrednio rozumianych przez procesor), po języki bliskie językowi angielskiemu, które są bardziej przyjazne ludziom.

Obecnie programowanie nie jest aż tak trudne, jak było to jeszcze 30 lat temu.

Dziś dysponujemy językami programowania, które są proste, pozwalają cieszyć się efektem naszej pracy bez zbytniego wchodzenia w to, co dzieje się w głębi naszej maszyny. Przykładem takiego języka jest język Python.

Dlaczego Python?

Wbrew opiniom niektórych ludzi Python to pełnoprawny język programowania. Nie jest to język będący uproszczeniem języka programowania nadającym się tylko i wyłącznie do edukacji. Nawet wielkie korporacje, takie jak Google, używają Pythona.

Język ten prężnie się rozwija i można nawet powiedzieć, że jest w sile wieku. Nie jest jednak też bardzo młody, gdyż powstał w latach 90. ubiegłego wieku. Początkowo nie wyglądał tak jak dziś, jednak pewne konwencje, jak używanie wcięć w kodzie czy maksymalna prostota, są do dziś zachowane. Cechą szczególną programów napisanych w Pythonie jest ich czytelność i przejrzystość. Estetyka ta została uzyskana przez zastosowanie wspomnianych wcześniej wcięć.

Python może służyć do nauki programowania przez sterowanie ruchami żółwia, ale nic nie stoi na przeszkodzie, by język ten posłużył także do zbudowania serwisu podobnego do YouTube czy tworzenia algorytmów sztucznej inteligencji. Jest wszechstronny i bardzo łatwy do nauki.

Ciekawostka

Twórcą języka Python jest Holender Guido von Rossum. Python powstał jako jego wakacyjny projekt. Guido nazwał go w ten sposób nie dlatego, że był fanem węży, lecz na cześć swojego ulubionego serialu komediowego „Latający cyrk Monty Pythona".

Dużym plusem przemawiającym za tym językiem jest ogromna społeczność ludzi znających go oraz dzielących się wiedzą na różnorakich forach tematycznych. W większych miastach organizowane są nawet spotkania miłośników tego języka. W Warszawie jest to cykl spotkań PyWaw, a w Krakowie PyLight i PyKonik.

Twój pierwszy program w Pythonie

Powyższy tytuł może być nieco mylący ze względu na to, że prawdopodobnie swój pierwszy program, napisany świadomie lub nie, masz już za sobą, o ile przeczytałeś pierwszy rozdział niniejszej książki i wykonałeś konfigurację swojego komputera tak, abyś mógł programować w języku Python.

Każdy kurs programowania zaczyna się od wypisania na ekranie wyrażenia „Hello world", co po polsku oznacza „Witaj, świecie". W przypadku tej książki nie będzie inaczej.

Ciekawostka

Pierwsze wykorzystanie wyrażenia „Hello world" w nauce programowania ma swój początek około roku 1967[1]. Program ten został napisany w języku BCLP.

Uruchom PyCharm i przejdź do utworzonego wcześniej projektu *turtle_nauka*. Jeśli nie masz jeszcze projektu, to najwyższy czas go utworzyć. Następnie stwórzmy plik naszego programu i nazwijmy go *hello_world*. Nie będę tego opisywał krok po kroku, gdyż zostało to już zaprezentowane w pierwszym rozdziale książki. Szczególnie polecam Ci ponowne przejrzenie podrozdziału „Pierwszy program".

W naszym pliku *hello_world.py* wpisujemy kod z listingu 2.1.

Listing 2.1. Kod programu wyświetlający napis „Hello world!"

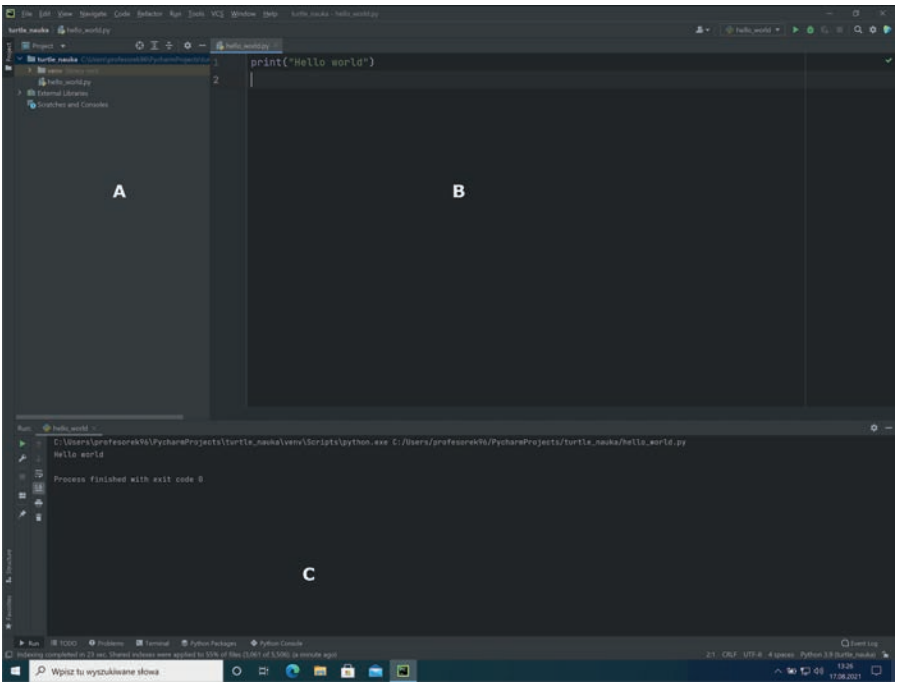
```
print("Hello world")
```

Następnie wciskamy prawy przycisk myszy, po czym naszym oczom powinno ukazać się menu. W nim wybieramy opcję `"Run 'hello_world.py'"`.

Jeśli na początku wszystko zostało poprawnie skonfigurowane, plik *hello_world.py* zostanie uruchomiony przez interpreter języka Python. Na dole okna programu PyCharm pojawi się okno. To właśnie w nim powinien wyświetlić się napis `Hello world`. Polecenie `print` to funkcja dostarczona przez twórców języka Python. Służy ona do wypisywania tego, co zostanie umieszczone w nawiasach okrągłych. Wynik jej działania pojawi się w oknie *Run*, czyli części C z rysunku 2.1.

Nasze okno PyCharm powinno przypominać to z rysunku 2.1.

[1] *https://pl.wikipedia.org/wiki/Hello_world*



Rysunek 2.1. A — widok projektu, B — aktualnie otwarty plik, C — okno Run

Ćwiczenie

Za pomocą polecenia `print` wypisz na ekranie swoje imię.

Zmienne

Program komputerowy operuje na danych i aby móc to robić, musi je w jakiś sposób przechowywać. Najprostszym sposobem jest użycie zmiennej, będącej jakby pudełkiem na dane, w którym możemy umieścić wartość będącą liczbą lub napisem. Spójrzmy na listing 2.2.

Listing 2.2. Kod programu tworzącego dwie zmienne

```
a = 5
b = 6
```

Kod ten tworzy dwie zmienne: `a` oraz `b`. Każda z nich ma inną wartość. Zmienne, jak nazwa wskazuje, mogą zmieniać swoją wartość, czyli to, co do nich przypiszemy, podczas działania programu. Nic nie stoi na przeszkodzie, abyśmy do zmiennej `a` dodali liczbę 3. Po tej operacji wartość zmiennej `a` będzie wynosić 8. Przykład ten został pokazany na listingu 2.3. Użyłem funkcji `print` do wpisania wartości zmiennej `a`.

Listing 2.3. Kod programu tworzącego dwie zmienne i następnie zwiększającego wartość jednej z nich o 3. Program wypisuje na ekranie wartość tej zmiennej

```
a = 5
b = 6
a = a + 3
print(a)
```

Zwróć uwagę na zapis `a = a + 3`. Zapis ten może być dla Ciebie dziwny, jeśli znasz już równania matematyczne. Zmienna jest równocześnie symbolem. W tej linii symbol `a` występuje zarówno po jednej stronie znaku równości, jak i po drugiej. Linię tę należy rozumieć tak, że do zmiennej `a` przypisujemy wartość `a` powiększoną o 3.

Ćwiczenie

Przepisz kod z listingu 2.3. Następnie zamień operację dodawania `+` na operację odejmowania `-`, mnożenia `*` oraz dzielenia `/`. Po każdej zmianie uruchom program i sprawdź, co zostało wypisane na ekranie.

Typy zmiennych

Jak już wspominałem w poprzednim rozdziale, zmienna to pudełko na dane. Jak wiemy z codziennego życia, nie każde pudełko nadaje się do przechowywania każdej substancji. Na przykład, jeśli chciałbyś przechować litr wody, potrzebujesz szczelnego pojemnika, zaś do przechowywania monet taki pojemnik nie jest Ci potrzebny. Tak samo jest ze zmiennymi, które mają swoje typy. W przypadku Pythona to sam język dobiera odpowiedni typ, pasujący do danych, jakie zostały przypisane do zmiennej.

Spójrz na kod z listingu 2.4. Kod ten przedstawia cztery zmienne: `a`, `b`, `c` oraz `d`. Zmienne te są różnych typów. Przypisując zmiennej `c` wartość napisu `"Ala ma kota"`, interpreter języka Python ustawił zmiennej `c` typ `str`, który pozwala na przechowywanie oraz operacje na napisach.

Ciekawostka

Python jest przykładem języka dynamicznie typowanego. Istnieją również języki statycznie typowane, takie jak C++, Java, Haskell. W językach tych programista tworzy zmienną o konkretnym typie, którego nie jesteśmy w stanie później zmienić.

Listing 2.4. Kod programu tworzącego cztery zmienne o typach int, float, str, bool

```
a = 5
b = 6.8
c = "Ala ma kota"
d = True
```

Oczywiście nie musisz mi wierzyć na słowo, w Pythonie bardzo łatwo można sprawdzić typ zmiennej. Służy do tego funkcja `type`. Kod z listingu 2.5 pozwala udowodnić, że zmienna `a` jest typu `int`, `b` typu `float`, `c` typu `str`, zaś `d` typu `bool`. Zwróć uwagę na funkcję `print`, możemy za jej pomocą wypisać na ekranie więcej niż jeden element. Wystarczy, że wewnątrz nawiasów okrągłych wymienimy elementy, które chcemy wypisać, rozdzielając je przecinkami.

Listing 2.5. Kod programu tworzącego cztery zmienne o typach int, float, str, bool oraz wypisujący wartości i ich typy na ekranie

```
a = 5
b = 6.8
c = "Ala ma kota"
d = True
print("a:", type(a))
print("b:", type(b))
print("c:", type(c))
print("d:", type(d))
```

Linia `print("a:", type(a))` powoduje wypisanie na ekranie `a: <class 'int'>`.

Dla wyjaśnienia dodam jeszcze, że typ `int` (z ang. *Integer*) to typ zmiennej przechowującej liczby całkowite. Typ `float` pozwala na przechowywanie liczb zapisywanych z przecinkiem, zwanych liczbami rzeczywistymi. Zmienne przechowujące tekst mają typ `str`, co jest skrótem od *string* (łańcuch, napis). Zmienna `d` w przykładzie z listingu 2.5 ma typ `bool`. Jest to specjalny typ pozwalający na przechowywanie dwóch wartości: prawdy i fałszu. W języku Python wartość prawda oznaczamy jako `True`, zaś fałsz jako `False`.

Ciekawostka

Język Python, podobnie jak większość języków programowania, używa jako separatora dziesiętnego znaku kropki. Liczba dwa i pół zapisywana jest jako 2.5. Znak przecinka służy do czegoś innego. Z kolei na przykład w Excelu do zapisu separatora dziesiętnego w liczbach rzeczywistych służy właśnie przecinek.

Oczywiście język Python posiada w swoim arsenale znacznie więcej typów niż te wymienione. Jest to jedynie wstęp do bardziej obszernego tematu.

Ciekawostka

Typ bool wziął się od nazwiska angielskiego matematyka George'a Boole'a. Współczesne komputery korzystają z tak zwanej algebry Boole'a.

Dobrą praktyką jest tworzenie zmiennych o nazwach sugerujących, co może robić dana zmienna czy funkcja. Tworząc dłuższe programy, staraj się nie stosować jednoliterowych oznaczeń zmiennych.

Ćwiczenie

Przepisz kod z listingu 2.5. Zmień wartości zmiennych `a`, `b`, `c`, `d`. Następnie uruchom program. Sprawdź, jakie typy mają poszczególne zmienne.

Konwersje typów

W poprzednim podrozdziale dowiedziałeś się, czym są typy zmiennych, poznałeś też ich podstawowe rodzaje. Nic we wszechświecie nie jest stałe i tak samo jest z typami zmiennych. Python pozwala nam na zmianę tych typów. Przeanalizujmy listing 2.6.

Listing 2.6. Kod programu tworzącego jedną zmienną o typie int (Python sam nadał taki typ). Następnie typ tej zmiennej konwertowany jest na int. Kolejna linia wypisuje wartość oraz typ zmiennej

```
a = 5
a = int(a)
print(a, type(a))
```

Funkcja `int` pozwala na konwertowanie zmiennej podanej w nawiasach okrągłych na zmienną o typie `int`. W przypadku kodu z listingu 2.6 zmienna `a` na samym początku jest typu `int`, konwersja ani trochę jej nie zmienia, jednak nie zawsze musi tak być. Uruchommy kod z listingu 2.7.

Listing 2.7. Kod programu tworzącego jedną zmienną o typie `float`. Następnie typ tej zmiennej konwertowany jest na `int`. Kolejna linia wypisuje wartość oraz typ zmiennej

```
a = 5.9
a = int(a)
print(a, type(a))
```

Na ekranie powinniśmy zobaczyć następujący wynik: `5 <class 'int'>`. Oznacza on, że zmiana typu się powiodła. Przed wykonaniem linii `a = int(a)` zmienna `a` miała typ `float`. Dzieje się tak, ponieważ w pierwszej linii wpisaliśmy do zmiennej `a` wartość `5.9`, wartość ta jest liczbą rzeczywistą. Na ekranie została wypisana wartość nie `5.9`, ale `5`, ponieważ zmiana typu niesie za sobą często również zmianę wartości. Zmienne typu `int` nie mogą przechowywać liczb rzeczywistych. Zamiana liczby typu `int` na liczbę typu `float` nie spowoduje zmiany wartości, a więc nie tracimy żadnych liczb po przecinku. Ilustruje to listing 2.8.

Listing 2.8. Kod programu tworzącego jedną zmienną o typie int. Następnie typ tej zmiennej konwertowany jest na float. Kolejna linia wypisuje wartość oraz typ zmiennej po konwersji

```
a = 5
a = float(a)
print(a, type(a))
```

Warto pamiętać, że niektóre operatory nie są dostępne dla każdego typu zmiennej. Wszelkie konwersje typów należy przeprowadzać z rozwagą. Spróbujmy uruchomić teraz kod z listingu 2.9.

Listing 2.9. Kod programu tworzącego jedną zmienną o typie str. Następnie typ tej zmiennej konwertowany jest na int. Kolejna linia wypisuje wartość oraz typ zmiennej

```
a = "Ala ma kota"
a = int(a)
print(a, type(a))
```

Niestety kod ten się nie uruchomi. Na ekranie zobaczymy komunikat o błędzie. Dzieje się tak, ponieważ Python nie umie skonwertować typu tekstowego na liczbę, co stanowi namacalny dowód, że nie każda konwersja typów jest możliwa.

Jeśli natomiast odwrócilibyśmy sytuację i do zmiennej `a` przypisalibyśmy liczbę zapisaną jako napis (w podwójnych lub pojedynczych znakach cudzysłowu), to okaże się, że konwersja jest możliwa. Sytuacja ta została pokazana na listingu 2.10.

Listing 2.10. Kod programu tworzącego jedną zmienną o typie str. Następnie typ tej zmiennej konwertowany jest na int. Kolejna linia wypisuje wartość oraz typ zmiennej

```
a = "5"
a = int(a)
print(a, type(a))
```

Ćwiczenie

Przepisz kod z listingu 2.10. Następnie spraw, aby zmienna `a` była typu `bool`. Sprawdź, czy możliwa jest konwersja typu `bool` na `int`.

Komentarze

Pisząc program, często chcielibyśmy w jakiś sposób opisać swoimi słowami, za co jaka część programu jest odpowiedzialna. Język Python oferuje taką funkcję i nazywa się ona komentarzem. W języku Python wyróżniamy dwa rodzaje komentarzy: jednoliniowe, obejmujące swoim zasięgiem tylko jedną linię, oraz wieloliniowe. Zacznijmy od omówienia tych pierwszych. W omawianym przez nas języku wszystko, co znajdzie się po znaku hash (#), stanie się komentarzem. Innymi słowy, interpreter języka Python nie bierze tego pod uwagę. Komentarz taki kończy się wraz ze znakiem nowej linii. Kolejna linia już nie jest komentarzem. Spójrz na listing 2.11.

Listing 2.11. Kod programu prezentującego użycie komentarza jednoliniowego

```
a = 5  # zmienna o nazwie a
# Python to mój ulubiony język programowania
print(a)
```

Program ten wypisze na ekranie wartość zmiennej `a`, czyli 5. Nie ma znaczenia, w jakim języku będziemy pisać komentarz, interpreter języka Python i tak go nie widzi.

Ciekawostka

Większość środowisk programistycznych i edytorów kodu pozwala na bardzo szybkie oznaczanie kodu komentarzami i usuwanie ich. Wystarczy, że zaznaczymy obszar, który chcemy objąć komentarzem, a następnie użyjemy kombinacji klawiszy *Ctrl+/* dla Windows oraz *cmd+/* dla systemów MacOS. Ten sam skrót klawiszowy pozwala zarówno na dodanie komentarza, jak i usunięcie go.

Zajmijmy się teraz komentarzem wieloliniowym. Został on pokazany na listingu 2.12.

Listing 2.12. Kod programu prezentującego użycie komentarza jednoliniowego i wieloliniowego

```
"""
Prosty program prezentujący
komentarz
wieloliniowy
"""
a = 5 # zmienna o nazwie a
# Python to mój ulubiony język programowania
print(a)
```

Wystarczy, że umieścimy tekst, który ma być komentarzem, między znakami cudzysłowu, tj. " — trzema na początku oraz trzema na końcu. Tak naprawdę zapis ten tworzy napis wielowierszowy. Jednak napisami zajmiemy się w późniejszych rozdziałach tej książki.

Pamiętaj jednak, by nie nadużywać narzędzia, jakim jest komentarz, i komentuj tylko trudniejsze kawałki kodu. Nie chciałbyś mieć więcej tekstu opisującego kod niż samego kodu.

Istnieje jeszcze inne zastosowanie dla komentarzy. Często komentarz, szczególnie ten jednoliniowy, stosuje się do wyszukiwania błędów w naszym programie. Komentujemy jak największą część kodu, tak aby program uruchomił się bez błędów. Następnie linia po linii sukcesywnie usuwamy z kodu znaki komentarza. W efekcie natrafimy na linię, która sprawia problem.

Ćwiczenie

Przepisz kod z listingu 2.12. Następnie zmień komentarze na zaproponowane przez siebie. Sprawdź, czy po tej zmianie program nadal się uruchamia. Zmiana treści komentarzy nie powinna prowadzić do błędów.

Operatory arytmetyczne

Tytuł tego podrozdziału mógł wprawić Cię w przerażenie, jednak nie jest to nic specjalnie nowego ani trudnego. Tak naprawdę już wcześniej w tej książce używaliśmy operatorów. Przykładem operatora jest np. operator dodawania, czyli znak +. Pozwala on na dodanie wartości dwóch zmiennych.

Patrząc na listing 2.13, pewnie jesteś zdziwiony. Operatorów arytmetycznych w Pythonie jest znacznie więcej niż cztery znane Ci ze szkoły podstawowej (dodawanie, odejmowanie, mnożenie oraz dzielenie). Zwróć uwagę na dwa podobne na pierwszy rzut oka operatory / oraz //. Pierwszy z nich oznacza zwykłe dzielenie, a wynikiem będzie liczba rzeczywista. Drugi z nich powoduje dzielenie całkowite, a wynikiem jego działania jest liczba całkowita bez reszty. Nie jest to zaokrąglenie. Jeśli wynikiem dla operatora / jest liczba 2,9, to dla tych samych danych operator // zwróci 2. Innymi słowy, wartość po przecinku zostanie pominięta w przypadku tego operatora.

Operator `%` (zwykle oznaczający procent) nie ma nic wspólnego z obliczaniem procentu z danej liczby. Wynikiem jego działania jest reszta z dzielenia liczby `a` przez liczbę `b`. W przykładzie z listingu 2.13 dzielimy 18 przez 4, wartość 4 mieści się w liczbie 18 dokładnie 4 razy. Po pomnożeniu 4 razy 4 otrzymujemy liczbę 16, my dzielimy liczbę 18 przez 4. Czyli zostaje nam 2 do podziału. Właśnie ta liczba to tak zwana reszta z dzielenia. Sprawa troszkę się komplikuje, kiedy jedna z liczb ma wartość ujemną, jednak nie będę tego opisywał w tej książce. Ostatnim z operatorów jest operator potęgowania, oznaczony jako **. Dzięki niemu jesteśmy w stanie szybko podnieść liczbę do potęgi. Zapis `a ** b` oznacza, że do potęgi równej liczbie `b` chcemy podnieść liczbę `a`. Dodam, że dla kodu `2 ** 4` wynik to `16`.

Listing 2.13. Kod programu prezentującego użycie operatorów arytmetycznych

```
a = 18
b = 4
# dodawanie
print("a+b:", a + b)
# odejmowanie
print("a-b:", a - b)
# mnożenie
print("a*b:", a * b)
# dzielenie
print("a/b:", a / b)
# dzielenie całkowite
print("a//b:", a // b)
# reszta z dzielenia
print("a%b:", a % b)
# potęgowanie
print("a**b:", a ** b)
```

Zanim przejdziemy dalej, spójrz na przykład przedstawiony na listingu 2.14.

Listing 2.14. Kod programu prezentujący działanie operatorów arytmetycznych

```
a = 1 + 4 * 9 - 5
print(a)
```

Uruchamiając ten przykład, powinniśmy zobaczyć liczbę 32. Jest to poprawny wynik; Python zachowuje kolejność wykonywania działań znaną z lekcji matematyki.

Ćwiczenie

Przepisz kod z listingu 2.13. Następnie sprawdź, jakie wyniki uzyskujesz dla różnych wartości zmiennych `a` oraz `b`.

Operatory porównania

Python pozwala na porównywanie zmiennych tego samego typu. Mając dwie zmienne, możemy w łatwy sposób sprawdzić, czy jedna z nich jest równa drugiej; a może jest większa? Służą do tego operatory porównania. Zobacz listing 2.15.

Listing 2.15. Kod programu prezentujący działanie operatorów porównania

```
a = 18
b = 4
# czy a jest równe b
print("a==b:", a == b) (1)
# czy a jest mniejsze od b
print("a<b:", a < b)
# czy a jest większe od b
print("a>b:", a > b) (2)
# czy a jest mniejsze od b bądź równe
print("a<=b:", a <= b)
# czy a jest większe od b bądź równe
print("a>=b:", a >= b)
# czy a jest różne od b
print("a!=b:", a != b)
```

Po uruchomieniu kodu z listingu 2.15 możesz zobaczyć na ekranie wartości `True` oraz `False`. Operatory porównania zwracają wartość `True`, jeśli sprawdzana wartość jest prawdziwa, a `False` w przeciwnym wypadku. W celu głębszego zrozumienia spójrz na pierwszą linię, w której występuje operator porównania, czyli **print**(`"a==b:", a == b`). Linia ta zawiera operator `==`, który

służy do sprawdzania, czy jedna zmienna jest równa drugiej. Nie należy go mylić z operatorem = zwanym operatorem przypisania. W tym przykładzie wartość zmiennej `a` jest równa 18, a wartością zmiennej `b` jest 4. Wartości tych zmiennych nie są sobie równe, dlatego linia ❶ wypisze na ekranie `False`.

Spójrz na linię oznaczoną ❷. Sprawdza ona, czy wartość zmiennej `a` jest większa od wartości zmiennej `b`. Jest to prawda, więc na ekranie zostanie wypisana wartość `True`.

Więcej przykładów użycia operatorów porównania znajdziesz w podrozdziałach poświęconych wyrażeniom warunkowym oraz pętlom.

Ćwiczenie

Przepisz kod z listingu 2.15. Następnie sprawdź, jakie wyniki uzyskujesz dla różnych wartości zmiennych `a` oraz `b`.

Operatory przypisania

Język Python pozwala na operowanie na wartościach zmiennych. Służą do tego operatory przypisania. Pierwszym ich przedstawicielem jest operator, który poznaliśmy przy okazji omawiania zmiennych. Jest to operator przypisania, czyli znak równa się (=). Używamy go, kiedy chcemy nadać wartość zmiennej. Zapraszam Cię do analizy oraz uruchomienia listingu 2.16.

Listing 2.16. Kod programu prezentujący działanie operatorów przypisania

```
a = 18
b = 4

# przypisujemy do zmiennej c
# sumę wartości zmiennej a oraz b
c = a + b ❶
print("c =:", c)

# zwiększamy wartość zmiennej c
# o wartość zmiennej a
# zapis ten jest tożsamy z zapisem
# c = c + a
c += a ❷
print("c +=:", c)

# zmniejszamy wartość zmiennej c
# o wartość zmiennej a
# zapis ten jest tożsamy z zapisem
# c = c – a
c -= a
print("c -=:", c)
```

```
# mnożymy wartość zmiennej c
# przez wartość zmiennej a
# zapis ten jest tożsamy z zapisem
# c = c * a
c *= a
print("c *=:", c)

# dzielimy wartość zmiennej c
# przez wartość zmiennej a
# zapis ten jest tożsamy z zapisem
# c = c / a
c /= a
print("c /=:", c)

# obliczamy resztę z dzielenia zmiennej c
# przez wartość zmiennej a
# wynik zapisujemy do zmiennej c
# zapis ten jest tożsamy z zapisem
# c = c % a
c %= a
print("c %=:", c)

# podnosimy wartość zmiennej c
# do potęgi równej wartości
# zmiennej a
# zapis ten jest tożsamy z zapisem
# c = c ** a
c **= a
print("c **=:", c)

# dzielimy całkowicie wartość zmiennej c
# przez wartość zmiennej a
# zapis ten jest tożsamy z zapisem
# c = c // a
c //= a
print("c //=:", c)
```

Operatorów przypisania i omawianych wcześniej operatorów arytmetycznych oraz porównania można używać wspólnie. W linii ❶ najpierw wykonuje się operacja dodawania wartości zmiennych `a` oraz `b`. Następnie otrzymaną wartość przypisujemy do nowej zmiennej `c`. W tym przypadku używamy zarówno operatora arytmetycznego, jak i operatora przypisania.

Czasem zachodzi potrzeba zwiększenia obecnej wartości zmiennej o jeden lub inną wartość. Nic nie stoi na przeszkodzie, by zrobić to za pomocą operatora arytmetycznego, a następnie wynik ten przypisać do zmiennej. Jednak twórcy języka Python stworzyli operator `+=` pozwalający na skrócenie zapisu. Spójrz na linię oznaczoną jako ❷. Przedstawia ona praktyczne użycie tego operatora. Pozostałe operatory działają analogicznie do operatora `+=`.

PYTHON DLA NASTOLATKÓW

Każdy może zostać programistą!

Czy wiesz, czym się zajmuje programista? To ktoś, kto, używając swojego umysłu i odpowiedniego języka programowania, rozwiązuje rozmaite problemy. Programista to taki współczesny superbohater. Przychodzi, siada do komputera, szybko przebiega palcami po klawiaturze i proszę — działa. Oczywiście, to pewne uproszczenie, ale… Brzmi ciekawie? Słusznie. Bo praca programisty, kodera, developera jest ciekawa. I fajna. I daje dużo satysfakcji. A najlepsze jest to: podstaw programowania można się szybko nauczyć, po prostu się bawiąc. We własny, ulubiony sposób.

Choć Twoim przewodnikiem po świecie programowania w Pythonie będzie żółw, obiecujemy, że praca pójdzie Ci w mig. Na początek nauczysz się konfigurować środowisko pracy, czyli uruchomisz na komputerze wszystko, co przyda się Tobie i żółwiowi. Potem zapoznasz się z językiem Python, z jego zmiennymi, funkcjami i klasami. Następnie zajrzysz do biblioteki turtle i dowiesz się, jak sterować swoim żółwiem. Wreszcie najlepsze: algorytmy. Przekonasz się między innymi, jak za pomocą kodu języka Python i elementów biblioteki turtle wygenerować niesamowite figury geometryczne.

UWAGA!

Książka jest polecana osobom biorącym udział w konkursie Logia. Informacje o konkursie można znaleźć pod adresem: **www.logia.oeiizk.waw.pl**.

Krzysztof Łos — pasjonat programowania. Jego ulubionym zajęciem, obok czytania książek, jest nauka. Od pięciu lat prowadzi zajęcia edukacyjne z młodzieżą, co jest dla niego okazją do samorozwoju i wypróbowywania nowych technologii. Stara się przedstawiać najtrudniejsze zagadnienia z dziedziny informatyki i matematyki w sposób jak najbardziej przystępny i dopasowany do możliwości uczniów. Jego ulubionym językami są: Python, który ceni za czytelność i niesamowitą prostotę, a także C++, który uwielbia za wydajność. W sieci występuje pod pseudonimem „profesorek96".

Helion

helion.pl

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

ISBN 978-83-283-7135-4

Cena: 39,90 zł