Temat C3. Tworzenie programów w języku Python

Warto powtórzyć

1. Jaką funkcję pełni pamięć operacyjna (RAM)?
2. Jakie informacje należy umieścić w specyfikacji zadania (problemu)?
3. Wyjaśnij pojęcia: *język programowania*, *program komputerowy*, *słowo kluczowe*.
4. Dlaczego o języku programowania mówimy, że jest językiem formalnym?
5. Czym jest kod źródłowy programu?
6. Z czego powinna wynikać kolejność instrukcji programu?
7. Na czym polega interpretacja programu?
8. Dlaczego komputer nie wykona błędnie napisanych instrukcji?
9. Czym jest typ zmiennej?

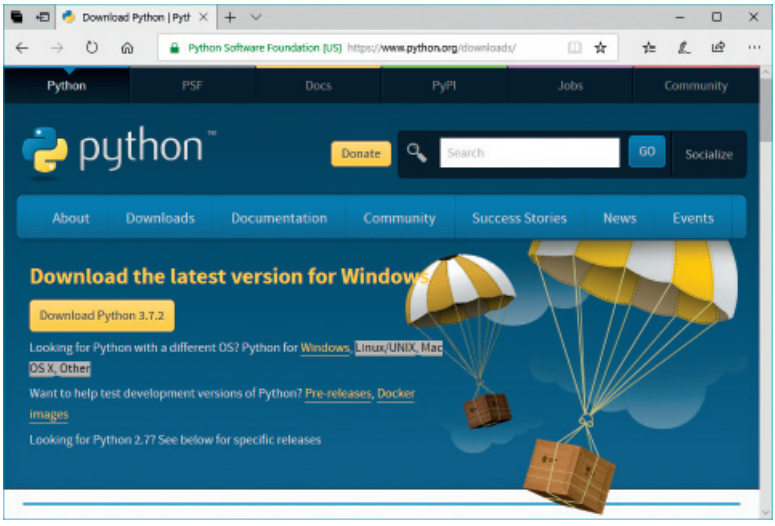
1. Środowisko programistyczne języka Python

Aby pisać programy w języku Python, należy zainstalować darmową aplikację Python, w skład której wchodzi **zintegrowane środowisko programistyczne IDLE**.

IDLE umożliwia tworzenie programów w języku Python, zapisywanie ich i uruchamianie. Zintegrowane środowisko programistyczne IDLE zawiera **powłokę Pythona** (**Python Shell**), **edytor kodu źródłowego** i **interpreter** oraz inne narzędzia wspomagające programowanie.

W tym podręczniku będziemy korzystać z aplikacji Python 3. Po zainstalowaniu aplikacji możemy ją uruchomić i rozpocząć programowanie.

Aby uruchomić IDLE, należy z menu **Start** wybrać program **Python 3.7/IDLE (Python 3.7 32-bit)**. Otworzy się okno programu powłoka Pythona (Python Shell).

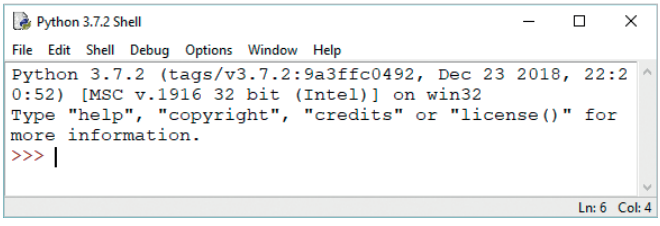


Rys. 1. *Oficjalna strona internetowa służąca m.in. do pobierania programu Python:* https://www.python.org/downloads/

1.1. Tryb interaktywny i skryptowy

Okno powłoki Pythona umożliwia pracę w dwóch trybach: **interaktywnym** i **skryptowym**.

Powłoka Pythona uruchamia się w trybie interaktywnym. Po znaku zachęty >>> widzimy migający kursor. W tym miejscu możemy napisać instrukcję lub wyrażenie. Po naciśnięciu klawisza **Enter** instrukcja zostanie wykonana, a wyrażenie obliczone. Tryb interaktywny widzieliśmy w poznanych w szkole podstawowej językach Logo i Scratch.



Rys. 2. *Okno powłoki Pythona*

Ćwiczenie 1. Sprawdzamy działanie trybu interaktywnego

1. Uruchom środowisko IDLE.
2. W oknie powłoki Pythona sprawdź działanie trybu interaktywnego. Wpisz po znaku zachęty:
   a. `2345 + 564` i naciśnij **Enter**,
   b. `print("Pracuję w trybie interaktywnym")` i naciśnij **Enter**.

W **trybie interaktywnym** możemy sprawdzić działanie jednego polecenia, np. gdy chcemy od razu zobaczyć wynik obliczeń czy sprawdzić działanie konkretnej instrukcji. Natomiast do tworzenia programów w języku Python służy **tryb skryptowy**.

W językach kompilowanych (takich jak C++) musimy napisać cały program, zapisać go w pliku, skompilować i dopiero po poprawnej kompilacji uruchomić. Jeśli kompilator znajdzie w programie błędy, wskaże je. Wtedy program należy poprawić, zapisać, ponownie skompilować, a następnie uruchomić.

W językach interpretowanych, takich jak Python, możemy napisać program składający się z wielu poleceń (tworząc tzw. skrypt), zapisać go w pliku tekstowym, a następnie uruchomić, używając interpretera. Mówimy wówczas, że pracujemy w trybie skryptowym.

1.2. Pisanie programu w trybie skryptowym

W programowaniu w języku Python, jak w każdym języku, bardzo istotna jest prawidłowa kolejność zapisu poleceń.

W języku Python bloki instrukcji wyodrębnia się inaczej niż w innych językach. Nie stosuje się nawiasów ani słów kluczowych, tylko wcięcia, składające się z przynajmniej jednej spacji (przyjęte jest wcięcie składające się z czterech spacji). Bloki instrukcji wykorzystamy w tematach C4 i C5, gdy będziemy omawiać instrukcje warunkowe i iteracyjne. Jeśli nasz program składa się z instrukcji, które nie wchodzą w skład bloku, piszemy je kolejno jedna pod drugą – bez wcięć (przykład takiego programu pokazano na rys. 5. w temacie C1 i na rys. 7a w tym temacie).

Komentarze w języku Python rozpoczyna się znakiem #. Tekst po znaku # do końca wiersza jest ignorowany przez interpreter.

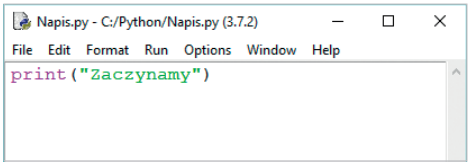
Do trybu skryptowego możemy przejść z trybu interaktywnego z okna powłoki Pythona. Program piszemy w edytorze kodu źródłowego, zapisujemy w pliku na dysku, a następnie uruchamiamy (poprzez wywołanie interpretera Pythona).

Etapy tworzenia programu w języku Python

1. W oknie powłoki Pythona (Python Shell) wybieramy opcję **File/New File** – otworzy się okno **edytora kodu źródłowego**.
2. W oknie edytora piszemy program (rys. 3a).
3. Zapisujemy program w pliku (w języku Python z rozszerzeniem *py*), wybierając w oknie edytora kodu źródłowego opcję **File/Save As**.
4. Uruchamiamy program, wybierając w oknie edytora opcję **Run/Run Module** (lub naciskając klawisz **F5**) – wynik działania programu pojawi się w oknie powłoki Pythona (rys. 3b).
5. Jeżeli wystąpiły błędy, należy je poprawić (interpreter wskazuje prawdopodobne miejsce popełnienia błędu – rys. 4a i 4b) i ponownie przejść do kroku 3.

Instrukcje języka Python, podobnie jak innych języków programowania, realizują takie czynności, jak: wprowadzanie danych, wyprowadzanie wyników, wykonywa-

nie obliczeń, sprawdzanie warunków czy powtarzanie operacji. Poznamy i zastosujemy te instrukcje w ćwiczeniach i zadaniach w tematach C3, C4 i C5.



Rys. 3a. *Okno edytora kodu źródłowego Pythona z programem źródłowym*

Dobra rada

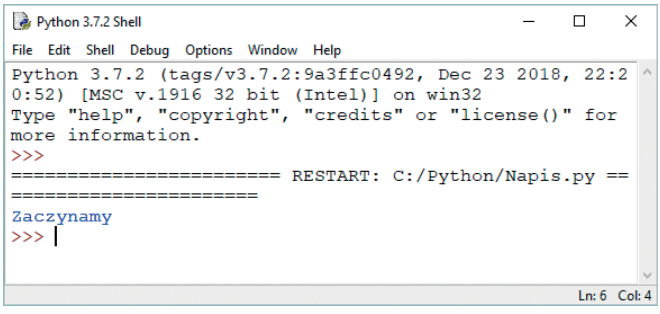
Staraj się samodzielnie poprawiać program do momentu, gdy da się go uruchomić. W nauce programowania umiejętność wyszukiwania i poprawiania błędów w programie jest bardzo ważna.

Ćwiczenie 2. Poznajemy etapy tworzenia programu w języku Python

1. Utwórz nowy plik źródłowy i napisz program wyświetlający na ekranie napis „Zaczynamy" – przepisz instrukcję pokazaną na rysunku 3a.
2. Zapisz program w pliku pod nazwą *Napis*.
3. Uruchom program.
4. Jeśli interpreter wykrył błędy, popraw je. Zapisz plik pod tą samą nazwą i ponownie uruchom.

Aby zmienić rozmiar czcionki, należy w oknie powłoki Pythona wybrać opcję **Options/Configure IDLE**.

W trakcie wykonywania programu interpreter rozpoznaje polecenia języka Python i zamienia je na instrukcje wewnętrznego języka procesora. Po wykonaniu programu wyniki pojawią się na wybranym urządzeniu zewnętrznym, np. na ekranie monitora (tu w oknie powłoki Pythona – rys. 3b).

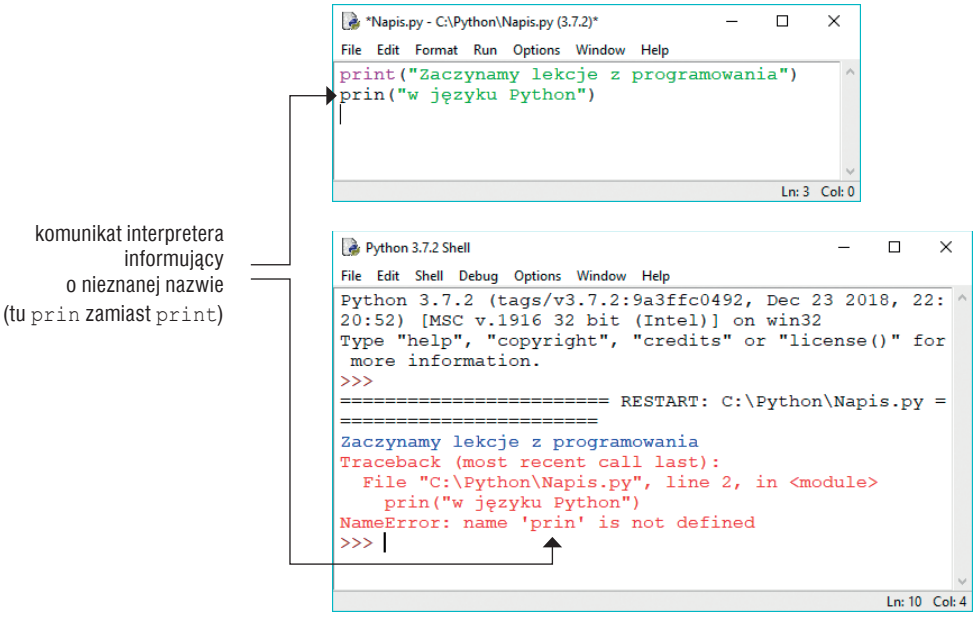


Rys. 3b. *Wynik działania programu – ćwiczenie 2.*

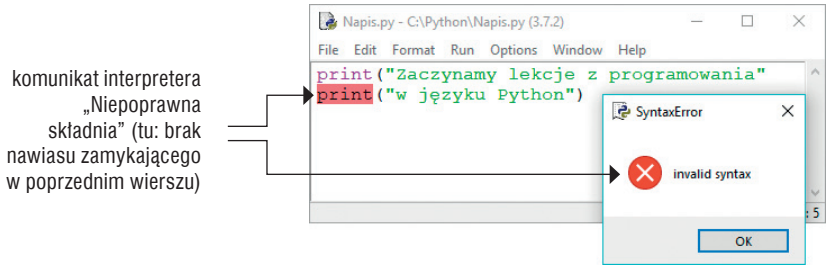
Uwaga

W nazwach plików, w których zapisujemy programy tworzone w języku Python, nie będziemy stosować polskich znaków diakrytycznych i spacji (w nazwach wieloczłonowych użyjemy znaku podkreślenia).

W odróżnieniu od kompilatora, **interpreter** próbuje wykonać program instrukcja po instrukcji. Jeżeli w programie wystąpi błąd, interpreter przerwie wykonanie programu i wyświetli komunikat o błędzie (rys. 4a i 4b). Należy poprawić wskazany błąd, ponownie zapisać program w pliku i uruchomić program.



Rys. 4a. *Okno powłoki Pythona z komunikatem interpretera*



Rys. 4b. *Okno powłoki Pythona z komunikatem interpretera*

Jeśli interpreter wykryje błąd, należy poprawić program, ponownie go zapisać i uruchomić.

Ćwiczenie 3. Modyfikujemy program

1. Zmodyfikuj program utworzony w ćwiczeniu 2., wyświetlając w tym samym wierszu dalszą część napisu: „lekcje z programowania", a w drugim wierszu – napis „w języku Python".
2. Jeśli interpreter wykrył błędy, popraw je. Zapisz plik pod tą samą nazwą. Ponownie uruchom program.

Wskazówki: Na rysunkach 4a i 4b pokazano przykładowe błędy wykryte przez interpreter. Oba błędy są w pierwszym wierszu programu, ale na rysunku 4b interpreter podświetlił drugi wiersz, ponieważ brakuje nawiasu zamykającego na końcu poprzedniego wiersza.

2. Stosowanie zmiennych

Chcemy obliczyć iloczyn dwóch liczb całkowitych. Jak napisać program w języku Python umożliwiający wprowadzenie dwóch liczb całkowitych z klawiatury, obliczający ich iloczyn i wyprowadzający wynik obliczeń na ekran?

Program, który utworzyliśmy w ćwiczeniach 2. i 3., wyświetlał na ekranie napis – nie używaliśmy w nim zmiennych i nie wykonywaliśmy obliczeń. Struktura kolejnych programów będzie bardziej rozbudowana, m.in. dodamy **zmienne**.

2.1. Nazywanie zmiennych

W wielu językach programowania (np. w języku C++) przed użyciem zmiennej należy ją zadeklarować, czyli określić jej nazwę i typ. W języku Python nie deklaruje się zmiennych. Nie musimy wcześniej określać typu zmiennej, czyli rodzaju przechowywanych w niej danych (np. liczby całkowite, rzeczywiste, znaki).

Interpreter określa typ zmiennej na podstawie rodzaju danych, jaki jej przypiszemy. Nie ustanawia jednak typu zmiennej na stałe, ponieważ przy kolejnym przypisaniu może się on zmienić. Na przykład, jeśli na początku programu przypiszemy zmiennej `liczba` wartość całkowitą (34), to jej typ jest całkowity. Jeśli jednak w dalszej części programu przypiszemy jej wartość rzeczywistą (np. 45,8, którą w języku Python zapisujemy z kropką: 45.8), to zmienna `liczba` będzie typu rzeczywistego.

Zasady dotyczące nazw zmiennych w języku Python

1. Wielkie i małe litery w nazwach traktowane są odmiennie (np. `suma` i `Suma` oznaczać będą różne zmienne). Pisząc program, należy zwracać uwagę na poprawne używanie małych i wielkich liter.
2. W nazwach zmiennych powinno się używać liter, znaku podkreślenia i cyfr. Nazwa nie może zaczynać się od cyfry. Przyjęte jest stosowanie małych liter i niestosowanie polskich liter.
3. W nazwach zmiennych nie wolno stosować spacji. W przypadku nazw kilkuczłonowych zamiast spacji stosujemy znak podkreślenia.
4. Należy nadawać nazwy, które określają znaczenie danej zmiennej, np. `suma`, `liczba_elementow`.

W kilku kolejnych ćwiczeniach napiszemy program obliczający iloczyn dwóch liczb całkowitych wprowadzanych z klawiatury. Zaczniemy od uzupełnienia specyfikacji zadania.

Ćwiczenie 4. Zapisujemy dane i wyniki do zadania

Wzorując się na opisie podanym w przykładzie 6. z tematu C1, zapisz w zeszycie przedmiotowym dane i wyniki do zadania: *Napisz w wybranym języku programowania program umożliwiający wprowadzenie z klawiatury dwóch liczb całkowitych, obliczający ich iloczyn i wyprowadzający wynik obliczeń na ekran. Przed wprowadzeniem pierwszej danej wyświetlaj napis „Podaj pierwszą liczbę”, przed wprowadzaniem drugiej – „Podaj drugą liczbę”, a przed wyprowadzeniem wyniku – „Iloczyn wynosi:”.*

2.2. Nadawanie zmiennym wartości

Zmiennej stosowanej w programie możemy nadać konkretną wartość za pomocą **instrukcji przypisania**. W instrukcji przypisania zmiennej podanej po lewej stronie instrukcji zostanie przypisana obliczona przez komputer wartość wyrażenia znajdującego się po prawej stronie instrukcji.

Instrukcja przypisania	`zmienna = wyrażenie`	**Python**

Wyrażeniem mogą być m.in. konkretna wartość, zmienna, wyrażenie algebraiczne.

Przykład 1. Stosowanie instrukcji przypisania

`rok = 2019`
– zmiennej `rok` przypisujemy wartość, która jest liczbą całkowitą, czyli zmienna `rok` będzie typu całkowitego (`int`),
`a = 7.58`
– zmiennej `a` przypisujemy wartość, która jest liczbą rzeczywistą, czyli zmienna `a` będzie typu rzeczywistego (`float`),
`x = y`
– zmiennej `x` przypisujemy wartość zmiennej `y`.
`obwod = 2 * a + 2 * b`
– zmiennej `obwod` przypisujemy wartość wyrażenia `2 * a + 2 * b`; jeśli wynik obliczeń będzie liczbą całkowitą, interpreter określi dla zmiennej `obwod` typ całkowity; jeśli wynik będzie liczbą dziesiętną, interpreter określi typ zmiennej jako rzeczywisty.

Zmiennej stosowanej w programie możemy również nadać wartość za pomocą instrukcji przypisania, wprowadzając wartość z klawiatury w trakcie działania programu. W tym celu stosujemy **instrukcję wejścia** – funkcję `input()`.

Argumentem funkcji `input()` (podanym w nawiasach) jest tekst (ciąg znaków, łańcuch). Tekst ten pojawi się na ekranie jako swego rodzaju „zaproszenie" do wpisania ciągu znaków z klawiatury. Jeśli użyjemy funkcji `input()` w instrukcji przypisania, wpisany przez użytkownika ciąg znaków zostanie zapamiętany w zmiennej podanej po lewej stronie.

Wprowadzanie danych z klawiatury	`zmienna = input(tekst_zachęty)`	**Python**

Aby dana wprowadzona za pomocą instrukcji `input()` została zapamiętana jako liczba, musimy dodać instrukcję, która zamieni ciąg znaków na liczbę: `int()` – w przypadku liczb całkowitych, `float()` – w przypadku liczb rzeczywistych.

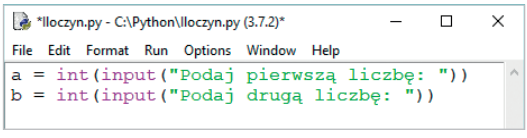
Przykład 2. Wprowadzanie danych z klawiatury – stosowanie funkcji `input()`

```
a = input("Wprowadź liczbę: ")
```
– jeśli wpiszemy z klawiatury liczbę 346, w zmiennej `a` zostanie zapamiętany ciąg znaków „346”, a nie liczba 346,
```
a = int(input("Wprowadź liczbę: "))
```
– jeśli wpiszemy z klawiatury liczbę 346, w zmiennej `a` zostanie zapamiętana liczba całkowita 346,
```
srednia = float(input("podaj średnią ocen: "))
```
– jeśli wpiszemy z klawiatury liczbę 4.3, w zmiennej `srednia` zostanie zapamiętana liczba rzeczywista 4,3,
```
miasto = input("Wprowadź nazwę miasta: ")
```
– jeśli wpiszemy z klawiatury nazwę miasta „Wrocław”, w zmiennej `miasto` zostanie zapamiętany ciąg znaków „Wrocław”.

W ćwiczeniach 5–7 napiszemy program obliczający iloczyn dwóch liczb na podstawie specyfikacji zadania zapisanego w ćwiczeniu 4.

Ćwiczenie 5. Wprowadzamy dane z klawiatury

1. Otwórz nowy plik źródłowy. Umieść polecenia wprowadzania danych z klawiatury, zgodnie z programem pokazanym na rysunku 5.
2. Zapisz program w pliku pod nazwą *Iloczyn* i uruchom program. Jeśli interpreter wykrył błędy, popraw je, ponownie zapisz plik i uruchom.



Rys. 5. *Program, w którym wprowadza się dane z klawiatury – ćwiczenie 5.*

2.3. Wykonywanie obliczeń

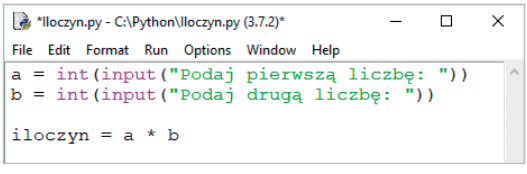
Aby zapisać w programie obliczenia, możemy skorzystać z instrukcji przypisania.

W języku Python stosujemy podstawowe operatory arytmetyczne: `+` (dodawania), `-` (odejmowania), `*` (mnożenia), `//` (dzielenia całkowitego), `/` (dzielenia zmiennoprzecinkowego) oraz `%` (reszty z dzielenia dwóch liczb całkowitych).

Ćwiczenie 6. Stosujemy instrukcję przypisania do zapisania obliczeń

1. Otwórz plik *Iloczyn* zapisany w ćwiczeniu 5.
2. Dodaj obliczanie iloczynu tak jak pokazano na rysunku 6.
3. Zapisz plik pod tą samą nazwą i uruchom program. Dlaczego nie widzisz wyniku obliczeń na ekranie? Uzasadnij odpowiedź.

Wskazówka: Aby otworzyć ponownie plik (jeśli zamkniemy okno edytora z programem), należy w oknie powłoki wybrać opcję **File/Open**.



Rys. 6. *Program obliczający iloczyn dwóch liczb wprowadzanych z klawiatury – ćwiczenie 6.*

Operator	Działanie	Przykład instrukcji przypisania	Wynik działania dla danych: a = 11 i b = 4
+	dodawanie	`suma = a + b`	11 + 4 = 15
–	odejmowanie	`roznica = a - b`	11 – 4 = 7
*	mnożenie	`iloczyn = a * b`	11 * 4 = 44
//	dzielenie całkowite (z zaokrągleniem części ułamkowej „w dół” – do największej liczby całkowitej mniejszej od wyniku dzielenia)	`iloraz = a // b`	11 // 4 = 2
/	dzielenie zmiennoprzecinkowe (z zachowaniem części ułamkowej)	`iloraz = a / b`	11 / 4 = 2.75
%	obliczenie reszty z dzielenia dwóch liczb całkowitych	`reszta = a % b`	11 % 4 = 3

Tabela 1. *Podstawowe operatory arytmetyczne w języku Python*

3. Wyprowadzanie komunikatów i wyników na ekran monitora

W ćwiczeniach 2. i 3. wyprowadzaliśmy komunikaty na ekran monitora, używając polecenia `print()`. Za pomocą tego polecenia można wyprowadzać również wyniki obliczeń.

Wyprowadzanie wyników i komunikatów na ekran monitora	`print(`*wartość*`)`	**Python**

W języku Python `print()` jest **funkcją**, która wyświetla na ekranie wartość zapisaną w nawiasach i umieszcza kursor w nowym wierszu.

Dobra rada

Mimo że interpreter pomija dodatkowe spacje, kod programu powinien być napisany czytelnie, przejrzyście i jednolicie (rys. 7a). Innej osobie łatwiej będzie zrozumieć program napisany w dobrym stylu.

Wartością umieszczoną wewnątrz nawiasów może być zmienna, wyrażenie, a także napis. Napis umieszczany w programie, który ma zostać wyświetlony na ekranie monitora, musi być ujęty w górne cudzysłowy podwójne `""` lub pojedyncze `''`. Funkcja `print()` pozwala łączyć wyświetlanie napisów, wyrażeń i zmiennych. Poszczególne wartości, występujące jako parametry funkcji `print()`, należy rozdzielić przecinkami.

Przykład 3. Wyprowadzanie komunikatów i wyników

```
print("Zaczynamy lekcje z programowania")
print(p)
print(a + b)
print(23 + 89)
print("Obwód =", 2 * a + 2 * b)
print("Obwód wynosi:", obwod)
print("Dla boku o długości", a, "pole kwadratu wynosi:", a * a)
```

Umieszczenie sekwencji znaków `\n` w tekście wyprowadzanym za pomocą instrukcji `print()` wymusza przejście do nowego wiersza.
Instrukcja:

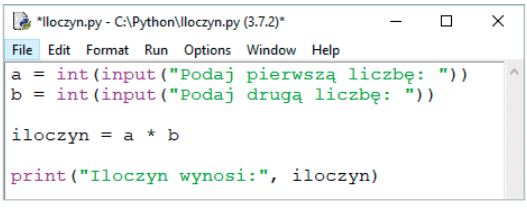
```
print("Pole prostokąta wynosi:", "\nbok1 * bok2 = ", pole)
```

i instrukcje:

```
print("Pole prostokąta wynosi:")
print("bok1 * bok2 =", pole)
```

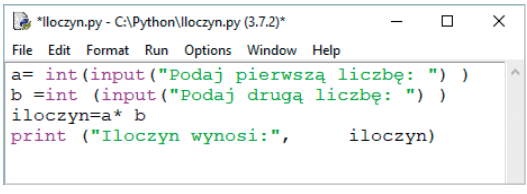
wyświetlą w taki sam sposób te same wyniki

```
Pole prostokąta wynosi:
bok1 * bok2 = 96
```



LEPIEJ

Rys. 7a. *Program napisany w uporządkowany sposób – ćwiczenie 7.*



GORZEJ

Rys. 7b. *Program napisany mniej przejrzyście – ćwiczenie 7.*

Ćwiczenie 7. Wyprowadzamy napisy i wynik na ekran

1. Otwórz plik *Iloczyn* zapisany w ćwiczeniu 6.
2. Dodaj wyświetlanie komunikatu i wyniku podobnie jak pokazano na rysunku 7a.
3. Zapisz plik pod tą samą nazwą i uruchom program.
4. Co sądzisz o programach pokazanych na rysunkach 7a i 7b? Wskaż kilka zasadniczych różnic.

Ćwiczenie 8. Testujemy działanie programu obliczającego średnią dla danych różnego typu

Uwaga

W większości języków programowania, w tym w języku Python, część dziesiętną liczby zapisujemy po kropce, a nie po przecinku, jak na matematyce. W ten sam sposób liczby dziesiętne są również wyświetlane na ekranie.

1. Do programu zapisanego w ćwiczeniu 7. dodaj obliczanie średniej arytmetycznej liczb *a* i *b*. Wynik wyprowadź na ekran. Dodaj odpowiednie komunikaty.
2. Zapisz program w pliku pod nazwą *Srednia_c*. Uruchom program. Przetestuj program dla par liczb (wartości zmiennych *a* i *b*): (2; 3), (14; 8); (7; 6). Dodatkowo przetestuj program dla pary liczb dziesiętnych (23,3; 12,9). Dlaczego program nie zadziałał?
3. Zmień w instrukcji przypisania typ wprowadzanych danych *a* i *b* na `float`. Zapisz program w pliku pod nazwą *Srednia_r*. Uruchom program. Przetestuj program dla tych samych danych. Czy teraz program zadziałał za każdym razem?
4. Przetestuj program *Srednia_r* dla liczb: (342,3; 25,7) (3763,82; 109,87).

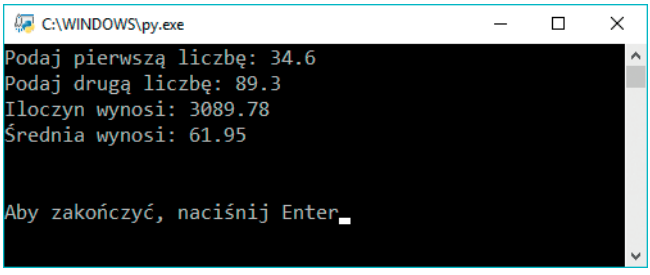
Wskazówka: Liczby dziesiętne wprowadzaj z kropką, a nie z przecinkiem.

Program możemy również uruchomić, klikając dwukrotnie nazwę pliku z zapisanym programem w Eksploratorze plików – otworzy się okno, w którym zobaczymy wynik działania programu (rys. 8.). Aby program został wykonany, niezbędny jest interpreter Pythona.

Aby okno pokazane na rysunku 8. nie zamknęło się automatycznie po wyświetleniu napisu, możemy napisać na końcu programu instrukcję oczekiwania na naciśnięcie klawisza **Enter**:

```
input("\n\nAby zakończyć, naciśnij Enter").
```

W tym przypadku wartość zwracana przez funkcję `input()` jest pomijana (nie jest przypisywana żadnej zmiennej). Umieszczenie zapisu `\n\n` oznacza wstawienie dwóch pustych wierszy przed komunikatem.

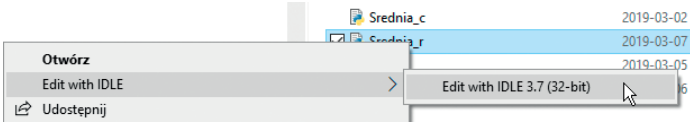


Rys. 8. *Wynik działania programu dla zmiennych typu rzeczywistego – ćwiczenia 8. i 9.*

Ćwiczenie 9. Uruchamiamy program, klikając jego nazwę

1. Otwórz plik *Srednia_r* zapisany w ćwiczeniu 8.
2. Dodaj na końcu programu polecenie oczekiwania na naciśnięcie klawisza **Enter**. Zapisz plik pod tą samą nazwą.
3. Zamknij okno edytora i uruchom program, klikając dwukrotnie jego nazwę lub ikonę w oknie Eksploratora plików. Wynik działania programu pokazano na rysunku 8.

Jeśli zamknęliśmy okno powłoki Pythona, a chcemy edytować program, można z menu kontekstowego nazwy pliku w oknie Eksploratora plików wybrać polecenie **Edit with IDLE/ Edit with IDLE 3.7 (32-bit)** – rys. 9.



Rys. 9. *Otwieranie okna edytora kodu źródłowego z programem zapisanym w pliku*

4. Zapisywanie rozwiązania problemu w języku Python

Chcesz przygotować tradycyjną sałatkę jarzynową na imprezę urodzinową. W Internecie są przepisy, ale w proporcjach dla dwóch, pięciu czy dziesięciu osób. Nie możesz znaleźć przepisu dla trzynastu osób – uczestników spotkania. Jak napisać program w języku Python, który policzy, ile potrzeba danego składnika sałatki w zależności od liczby gości, i wyświetli wynik na ekranie?

Przykład 4. Sformułowanie zadania, opis specyfikacji i projekt rozwiązania

Zadanie: Napisz program obliczający, ile potrzeba danego składnika sałatki dla *o2* osób, gdy znasz ilość składnika dla *o1* osób.

Dane: liczba całkowita $o1 > 0$ oznaczająca liczbę osób uwzględnioną w przepisie, liczba rzeczywista $s1 > 0$, określająca ilość składnika dla *o1* osób, liczba całkowita $o2 > 0$, oznaczająca liczbę uczestników spotkania.

Wynik: liczba rzeczywista $s2 > 0$ oznaczająca ilość składnika dla *o2* osób.

W rozwiązaniu skorzystamy z zależności:
jeśli

$s1 \longrightarrow$ dla *o1* osób
$s2 \longrightarrow$ dla *o2* osób

to

$$s2 = \frac{s1 \cdot o2}{o1}$$

Aby zapisać te obliczenia w języku Python, zastosujemy instrukcję przypisania:

```
s2 = (s1 * o2) / o1
```

Uwaga: Z matematycznego punktu widzenia nawiasy są zbędne – umieściliśmy je w programie dla zwiększenia czytelności.

Ćwiczenie 10. Realizujemy algorytm obliczania ilości składnika sałatki w języku Python

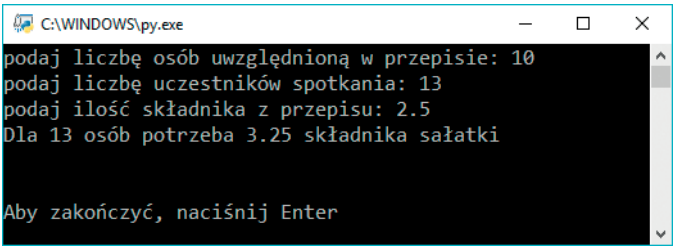
Uwaga

Na razie w programach nie sprawdzamy poprawności danych wprowadzanych przez użytkownika. Zakładamy, że użytkownik podaje poprawne dane. Problem poprawności danych omówimy w temacie C4.

1. Utwórz program realizujący zadanie sformułowane w przykładzie 4. Przed wprowadzaniem zmiennych dodaj komunikaty. Dodaj również komunikat przed wyprowadzeniem wyniku. Komunikaty i wynik wyprowadź na ekran monitora tak jak pokazano na rysunku 10.
2. Zapisz program w pliku pod nazwą *Przepis*. Uruchom program i przetestuj dla następujących trójek liczb (wartości zmiennych *o1*, *o2*, *s1*): (10; 13; 2,5), (3; 12; 0,45), (15; 4; 3,20).
3. Przetestuj program dla trzech innych trójek liczb.

Wskazówka: Aby wyświetlić wyniki, jak pokazano na rysunku 10., należy wpisać instrukcję:

```
print("Dla ", o2, " osób potrzeba ", s2, "składnika sałatki")
```



Rys. 10. *Wynik działania programu – ćwiczenie 10.*

Zauważmy, że program utworzony w ćwiczeniu 10. oblicza ilość jednego składnika sałatki. Jeśli chcemy obliczyć ilość kolejnego składnika, musimy ponownie uruchomić program. W temacie C5 zmodyfikujemy ten program, aby obliczał i wyświetlał na ekranie kolejno ilości wszystkich składników sałatki.

Warto zapamiętać

- Środowisko programistyczne Pythona (IDLE) udostępnia dwa tryby pracy:
  - interaktywny, w którym możemy napisać jedno polecenie i zatwierdzić je klawiszem **Enter**; jeśli polecenie będzie poprawne, zostanie od razu wykonane,
  - skryptowy, w którym możemy napisać program, zapisać go w pliku, a następnie uruchomić; jeśli program będzie poprawny, zostanie wykonany.
- Aby utworzyć program w języku Python, piszemy kod źródłowy w edytorze tekstu, zapisujemy program w pliku i uruchamiamy za pomocą interpretera Pythona.
- W języku Python typ wartości, która jest przypisana zmiennej, określa typ tej zmiennej.
- Zmiennej możemy nadać wartość za pomocą instrukcji przypisania. Po lewej stronie znaku = podajemy nazwę zmiennej, a po prawej:
  - konkretną wartość lub
  - funkcję `input()`, dzięki której użytkownik wprowadzi tę wartość z klawiatury po uruchomieniu programu lub
  - wyrażenie, którego wartość zostanie obliczona w czasie działania programu.
- Aby wyświetlić komunikaty i wyniki na ekranie, stosujemy instrukcję wyjścia – funkcję `print()`.

Pytania i polecenia

1. Wyjaśnij, czym jest tryb interaktywny, a czym tryb skryptowy w środowisku Python.
2. Przedstaw i omów na konkretnym przykładzie etapy tworzenia programu komputerowego w języku Python.
3. Przedstaw na przykładzie sposób wprowadzania danych z klawiatury.
4. Co w języku Python określa typ zmiennej?
5. W jaki sposób możemy nadać wartość zmiennej?
6. Do czego służy funkcja `input()`?
7. Omów na przykładzie sposób wyświetlania napisów i wyników na ekranie monitora.
8. Jaka jest rola zapisu `\n` umieszczonego w programie?
9. Wyjaśnij zapisy:
   a. `bok1 = int(input("podaj długość boku: "))`
   b. `liczba1 = float(input("podaj liczbę: "))`
   c. `iloraz = liczba1 / liczba2`
   d. `print("Iloraz wynosi:", iloraz)`
10. Wyjaśnij, jaka dana zostanie zapamiętana w zmiennej `x` po wykonaniu instrukcji przypisania, gdy z klawiatury wprowadzimy liczbę 258:
    a. `x = int(input("podaj długość"))`
    b. `x = input("podaj długość"))`
11. Wskaż nieprawidłowo zapisane instrukcje. Wyjaśnij, na czym polegają błędy:
    a. `liczba1 = input(int("Podaj liczbę: "))`
    b. `y = 29871`
    c. `print("Suma wynosi: ", a + b)`
    d. `iloraz = liczba1 : liczba2`

Zadania

Uwagi:

- W edytorze języka Python można kopiować (**Edycja**/**Kopiuj**) i wklejać (**Edycja**/**Wklej**) fragmenty kodu programu – w tym samym dokumencie i pomiędzy dokumentami (tak jak w edytorze tekstu). Ta możliwość jest przydatna, gdy ponownie wykorzystujemy te same lub podobne fragmenty kodu.
- Staraj się pisać przejrzyste programy oraz dodawać w odpowiednich miejscach komentarze.
- Każdy program należy uruchomić i przetestować dla różnych danych, nawet jeśli w zadaniu nie ma takiego polecenia.

1. Napisz specyfikację zadania i program obliczający pole powierzchni prostokąta dla danych długości boków *a* i *b* wprowadzanych z klawiatury. Po uruchomieniu programu na ekranie powinny pojawiać się w kolejnych wierszach komunikaty: „Podaj długość boku a: ”, „Podaj długość boku b: ”. Po podaniu danych w trzecim wierszu powinny wyświetlić się: napis „Pole prostokąta wynosi” oraz wartość pola. Zapisz program w pliku pod nazwą *Pole.*
2. Napisz specyfikację zadania i program realizujący algorytm obliczania kwadratu i sześcianu liczby rzeczywistej wprowadzanej z klawiatury. Zapisz program w pliku pod nazwą *Potegi* i uruchom. Przetestuj program dla różnych danych.
3. Napisz specyfikację i program do zadania: *Oblicz drogę* S *przebytą w czasie* t *przez pojazd poruszający się ze średnią prędkością* v. Zapisz program w pliku pod nazwą *Droga*.
   Wskazówka: Zadanie wykonaj na podstawie specyfikacji zapisanej w ćwiczeniu 2., punkt 1. z tematu C1.

4. Przepisz program pokazany na rysunku 11. Zapisz go w pliku pod nazwą *Test* i uruchom. Po wprowadzeniu z klawiatury liczby program powinien wyświetlić na ekranie liczbę o 2 większą. Jakie błędy pojawiły się w zapisie programu? Zmodyfikuj program, aby działał poprawnie. Zapisz plik i uruchom ponownie program.

```
liczba1 = int(input("podaj liczbę: "))
print(liczba2)
liczba2 = liczba1 + 2
```

Rys. 11. *Błędny program – zadanie 4.*

5. Napisz specyfikację i program do zadania: *Dane są: suma długości przekątnych rombu i długość boku. Oblicz długości przekątnych rombu*. Zapisz program w pliku pod nazwą *Romb*.
6. Napisz specyfikację i program do zadania: *Jaki procent liczby* a *stanowi liczba* b? Zapisz program w pliku pod nazwą *Procenty*.
7. Napisz program obliczający wysokość trójkąta, gdy podane są jego pole i podstawa. Zapisz program w pliku pod nazwą *Trojkat*.
8. Napisz program obliczający, ile litrów wody spadło na plac o powierzchni P m^2, jeśli pokryła go warstwa wody o grubości d milimetrów. Zapisz program w pliku pod nazwą *Deszcz*.

Dla zainteresowanych

9. Napisz specyfikację zadania i opisz rozwiązanie: *Dany jest średni wiek trzech osób:* S. *Najmłodsza osoba ma* x *lat, najstarsza:* y. *Ile lat ma trzecia osoba?* S, x, y *to liczby naturalne różne od zera.* Napisz program na podstawie zapisanej specyfikacji. Przetestuj program dla następujących wartości zmiennych (S, x, y): (15; 10; 20), (27; 20; 40), (24; 14; 33). Dodaj kilka swoich propozycji danych. Jakie powinny być wartości zmiennych: S oraz x i y, aby otrzymany wynik był sensowny? Uwzględnij te warunki w opisie algorytmu.
10. Dane są suma i iloczyn dwóch liczb całkowitych. Napisz specyfikację zadania i program znajdujący te liczby. Zapisz program w pliku pod nazwą *Liczby*.
11. Opisz wymyślony samodzielnie problem podobnie jak w punkcie 4. tego tematu. Ułóż zadanie, zapisz specyfikację i przygotuj rozwiązanie podobnie jak w przykładzie 4. Na koniec napisz program realizujący to zadanie. Zapisz program w pliku pod nazwą *Problem*.