

DO NOWEJ PODSTAWY  
PROGRAMOWEJ

Część 1

PODRĘCZNIK dla szkół ponadpodstawowych

# Informatyka

# Europejszka

Zakres podstawowy



Danuta Korman  
Grażyna Szabłowicz-Zawadzka

**Helion**  
EDUKACJA

Podręcznik dopuszczony do użytku szkolnego przez ministra właściwego do spraw oświaty i wychowania i wpisany do wykazu podręczników przeznaczonych do kształcenia ogólnego do nauczania informatyki na podstawie opinii rzeczoznawców:  
mgr. inż. Włodzimierza Kruszwickiego, prof. dr. hab. Macieja Sysło,  
mgr. Klemensa Stróżyńskiego.

Etap edukacyjny: III.

Typ szkoły: szkoła ponadpodstawowa (liceum ogólnokształcące i technikum).

Zakres kształcenia: podstawowy.

Rok dopuszczenia: 2020.

**Numer ewidencyjny w wykazie: 1087/1/2020**

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autorki oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autorki oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Joanna Zaręba

Ilustracja na okładce została wykorzystana za zgodą Shutterstock.

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto: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?ip1men>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-6261-1

Copyright © Helion 2020

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# ..... Spis treści

Wstęp .....	7
<b>Rozdział 1. Rozumienie i analizowanie problemów. Wprowadzenie do programowania w języku Python .....</b>	<b>9</b>
Temat 1. Etapy rozwiązywania zadań za pomocą komputera .....	10
Temat 2. Sposoby przedstawiania algorytmów .....	13
2.1. Lista kroków algorytmu .....	13
2.2. Schemat blokowy algorytmu .....	14
2.3. Program w języku programowania wysokiego poziomu .....	15
Temat 3. Klasyfikacja tekstowych języków programowania .....	16
Temat 4. Python od pierwszych kroków .....	20
4.1. Środowisko programowe Pythona .....	20
4.2. Operacje wejścia-wyjścia .....	21
4.3. Komentarze .....	22
4.4. Operatory arytmetyczne .....	22
Temat 5. Złożone operatory przypisania .....	23
Temat 6. Funkcje matematyczne w języku Python .....	25
Temat 7. Algorytmy z warunkami w języku Python .....	26
7.1. Instrukcje warunkowe .....	27
7.2. Operatory relacyjne .....	29
Temat 8. Operatory logiczne w języku Python .....	33
Temat 9. Instrukcje iteracyjne .....	35
9.1. Funkcja range() .....	35
9.2. Instrukcja iteracyjna for .....	36
9.3. Instrukcje iteracyjne while .....	37
Temat 10. Iteracja i n-wyrazowe ciągi liczbowe .....	41
10.1. Obliczanie wartości wyrazów ciągu liczbowego .....	41
10.2. Wykonywanie działań na wyrazach ciągu liczbowego .....	42

<b>Temat 11. Algorytmy rekurencyjne</b> .....	<b>45</b>
11.1. Funkcje w języku Python .....	46
11.2. Funkcje rekurencyjne i obliczanie wartości wyrazów ciągu liczbowego ..	48
<b>Temat 12. Wyznaczanie wyrazów ciągu Fibonacciego</b> .....	<b>50</b>
<b>Temat 13. Algorytm Euklidesa</b> .....	<b>54</b>
13.1. Algorytm Euklidesa .....	54
13.2. Obliczanie najmniejszej wspólnej wielokrotności .....	57
<b>Zadania do rozdziału 1.</b> .....	<b>59</b>
<b>Rozdział 2. Przestrzeganie prawa i zasad bezpieczeństwa</b> .....	<b>63</b>
<b>Temat 14. Przestrzeganie prawa</b> .....	<b>64</b>
14.1. Prawo autorskie .....	64
14.2. Ochrona danych osobowych .....	66
<b>Zadania do rozdziału 2.</b> .....	<b>70</b>
<b>Rozdział 3. Grafika komputerowa i prezentacje multimedialne</b> ...	<b>71</b>
<b>Temat 15. Grafika rastrowa</b> .....	<b>72</b>
15.1. Rozdzielczość a wielkość obrazu .....	73
15.2. Transformacje obrazu w grafice rastrowej .....	80
15.3. Formaty plików bitmapowych .....	86
<b>Temat 16. Grafika wektorowa</b> .....	<b>91</b>
16.1. Formaty grafiki wektorowej .....	91
16.2. Linia prosta i krzywa Béziera w grafice wektorowej .....	99
16.3. Grafika bitmapowa a grafika wektorowa .....	103
<b>Temat 17. Grafika trójwymiarowa — 3D</b> .....	<b>105</b>
17.1. Techniki tworzenia grafiki 3D .....	106
17.2. Geometria trójwymiarowa .....	107
17.3. Paint 3D .....	107
17.4. POV-Ray .....	113

<b>Temat 18. Prezentacje multimedialne</b> .....	<b>128</b>
18.1. Jak tworzyć rozbudowane prezentacje? .....	128
18.2. Jak ustalać parametry pokazu prezentacji multimedialnej? .....	134
<b>Zadania do rozdziału 3.</b> .....	<b>140</b>
<b>Rozdział 4. Dokumenty tekstowe o złożonej strukturze</b> .....	<b>143</b>
<b>Temat 19. Odwołania — spisy</b> .....	<b>144</b>
19.1. Spis treści .....	144
19.2. Spis ilustracji .....	147
19.3. Spis tabel .....	149
<b>Temat 20. Kolumny i sekcje</b> .....	<b>151</b>
20.1. Podział tekstu na kolumny .....	151
20.2. Sekcje w dokumencie .....	153
<b>Temat 21. Własne style i szablony</b> .....	<b>154</b>
21.1. Tworzenie nowego stylu na podstawie formatowania tekstu .....	155
21.2. Szablony .....	157
<b>Temat 22. Praca w trybie recenzji</b> .....	<b>158</b>
22.1. Śledzenie zmian .....	158
22.2. Wersja końcowa po korekcie .....	159
22.3. Komentarze .....	160
22.4. Akceptacja lub odrzucenie zmian .....	160
22.5. Porównanie wersji .....	161
<b>Zadania do rozdziału 4.</b> .....	<b>163</b>
<b>Projekt zespołowy</b> .....	<b>165</b>
<b>Bibliografia</b> .....	<b>167</b>
<b>Skorowidz</b> .....	<b>168</b>



Technologia informacyjna i informatyka to dziedziny, których wykorzystanie i dostępność stale wzrastają, a tempo zachodzących w nich zmian jest nieporównywalne z tempem rozwoju innych dyscyplin. Widać to również w dziedzinie edukacji informatycznej. **Obszary informatyki, które pojawiają się w edukacji informatycznej** na poziomie szkoły podstawowej i ponadpodstawowej, to:

- rozumienie, analizowanie i rozwiązywanie problemów;
- programowanie, aplikacje i robotyka;
- komputery, sieci i urządzenia cyfrowe;
- kompetencje społeczne;
- prawo i bezpieczeństwo.

W konsekwencji na wszystkich etapach realizowane są zagadnienia związane z algorytmiką i programowaniem, co wpływa na rozwój logicznego i algorytmicznego myślenia, a tym samym na umiejętność samodzielnego rozwiązywania problemów.

**Informatyka realizowana w szkole ponadpodstawowej na poziomie podstawowym:**

- jest przedmiotem obowiązkowym;
- nie jest przedmiotem maturalnym;
- stanowi poziom podstawowy dla przedmiotu informatyka na poziomie rozszerzonym, a więc wchodzi w zakres wymagań maturalnych tego przedmiotu.

**Informatyka Europejczyka to podręczniki do informatyki na poziomie podstawowym** (trzy części) i **na poziomie rozszerzonym** (dwie części). Podręczniki te są w pełni zgodne z **podstawą programową obowiązującą od 2019 roku**, z uwzględnieniem jej specyficznych cech, takich jak:

- **Spiralność** — na każdym etapie wymaga się umiejętności zdobytych wcześniej i rozszerza się je o umiejętności nowe; na początku każdego rozdziału w podręczniku podane są informacje na temat tego, co uczeń powinien pamiętać ze szkoły podstawowej, czego nauczy się z tego podręcznika oraz co zaplanowane jest jako kontynuacja na poziomie rozszerzonym.
- **Myślenie komputacyjne** — w podręczniku wykorzystywane są narzędzia informatyczne do rozwiązywania problemów wywodzących się z różnych dziedzin życia i przedstawiane są różnorodne zastosowania metod i technik algorytmicznych; zastosowano nauczanie poprzez świadome wykorzystanie metod i narzędzi informatycznych.
- **Rozwiązywanie problemów** — zastosowano nauczanie poprzez rozwiązywanie problemów z różnych dziedzin życia.
- **Stopniowe wprowadzanie trudnej problematyki** — abstrakcyjne myślenie algorytmiczne i programowanie kształtowane są stopniowo, od pierwszej klasy szkoły podstawowej, poprzez poziom podstawowy, a dla niektórych również rozszerzony, w szkole ponadpodstawowej, aż do egzaminu maturalnego z informatyki.
- **Praca zespołowa i metoda projektów** — preferuje się pracę w grupach, wspólne rozwiązywanie problemów i uzyskiwanie pozytywnych efektów.

- **Nowoczesność** — uwzględniane są najnowsze trendy w zastosowaniach informatyki i wyborze narzędzi.

Podręcznik powinien być realizowany **w układzie liniowym** — temat po temacie. Jednak jeden temat można wprowadzać na więcej niż jednej lekcji, co zależy od tempa pracy uczniów i konieczności wykonania dodatkowych ćwiczeń utrwalających. Podręcznik zawiera **wiele przykładów i ćwiczeń**, na podstawie których uczeń może przystąpić do **rozwiązywania zadań**. Każdy rozdział kończy się **zadaniami podsumowującymi** przerobiony materiał. Odpowiedzi nie należy umieszczać w podręczniku.

W części pierwszej podręcznika uczeń:

- Poznaje podstawy **algorytmiki i języka tekstowego Python**.
- Uczy się postępowania zgodnie z regulacjami prawnymi dotyczącymi **ochrony danych osobowych** oraz **prawa autorskiego**, poznaje konsekwencje łamania tych zasad.
- Projektuje modele dwuwymiarowe i **trójwymiarowe**, tworzy i edytuje projekty w **grafice rastrowej i grafice wektorowej**, wykorzystuje różne formaty obrazów, przekształca pliki graficzne, uwzględniając wielkość i jakość obrazów.
- Opracowuje dokumenty o różnorodnej tematyce, w tym informatycznej, i o rozbudowanej strukturze, posługując się przy tym **konspektem dokumentu**, dzieli tekst na **sekcje i kolumny**, tworzy **spisy treści, rysunków i tabel**, stosuje **własne style i szablony**, pracuje nad **dokumentem w trybie recenzji**.

Danuta Korman  
Grażyna Szabłowicz-Zawadzka



# Rozdział 1.

## ROZUMIENIE I ANALIZOWANIE PROBLEMÓW. WPROWADZENIE DO PROGRAMOWANIA W JĘZYKU PYTHON

### Co już potrafisz

Wiesz, czym jest **algorytm**. Potrafisz podawać przykłady algorytmów i analizować ich przebieg.

Określasz **specyfikację analizowanego problemu**, czyli dane i wyniki dotyczące rozwiązywanego zadania.

Potrafisz przedstawiać algorytmy w postaci **schematów blokowych** i **list kroków**. Określasz kroki realizacji algorytmów.

Konstruujesz proste **algorytmy liniowe** i **iteracyjne**.

Wiesz, czym jest **tekstowy język programowania**, i znasz podstawy takiego języka, na przykład Pythona, C++, Processingu, Javy czy Pascala.

Projektujesz, tworzysz i testujesz **oprogramowanie sterujące robotem** na ekranie lub w rzeczywistości.

### Czego się nauczysz

Poznasz wszystkie **etapy rozwiązywania problemów za pomocą komputera** — jednym z nich jest specyfikacja problemu, którą już określasz.

Nauczysz się **klasyfikować tekstowe języki programowania**.

Poznasz **środowisko programowania dla języka Python**.

Dowiesz się, czym jest **rekurencja**, i będziesz ją stosować w algorytmach.

Będziesz **sprawdzać poprawność algorytmów poprzez testowanie** dla przykładowych danych.

Rozpoczniesz **naukę języka Python** — poznasz strukturę programu, operacje wejścia-wyjścia, zmienne, wyrażenia i operatory, instrukcje warunkowe i iteracyjne oraz funkcje.

## Co pojawi się na poziomie rozszerzonym

Poznasz **tekstowy język programowania dopuszczony do egzaminu maturalnego** z informatyki rozszerzonej. Może to być również język Python, którego podstawy poznajesz na lekcjach informatyki na poziomie podstawowym.

Będziesz omawiać oraz stosować **zaawansowane algorytmy iteracyjne i rekurencyjne**.

Poznasz i będziesz wykorzystywać reprezentację algorytmów w postaci **pseudokodu**.

Zajmiesz się **analizą efektywności algorytmów**.

Poznasz **źródła błędów pojawiających się w obliczeniach komputerowych**.

## ..... Temat 1. Etapy rozwiązywania zadań za pomocą komputera

Wiesz już, że:

- **informatyka** to dziedzina, której głównym celem jest rozwiązywanie problemów z wykorzystaniem komputera;
- **algorytmika** to dział informatyki obejmujący algorytmy oraz ich własności;
- **algorytm** jest dokładnym przepisem na rozwiązanie problemu lub osiągnięcie jakiegoś celu, realizowanym w skończonej liczbie kroków; istnieje wiele sposobów rozwiązania danego zadania, dlatego każdemu problemowi odpowiada wiele metod prowadzących do prawidłowych wyników.

Aby znaleźć rozwiązanie określonego problemu, musisz posiadać informacje początkowe, które następnie są wykorzystywane przy realizacji algorytmu prowadzącego do otrzymania danych wyjściowych. Dlatego mówimy również, że algorytm to skończona liczba wykonywanych kolejno czynności, które prowadzą do przekształcenia danych wejściowych na wyniki będące rozwiązaniem danego zadania.

### Ciekawostka

Termin **algorytm** pochodzi od nazwiska arabskiego matematyka i astronoma Muhammada ibn Musa Al-Chorezmiiego, żyjącego na przełomie VIII i IX wieku. Zapoczątkował on stosowanie metod obliczeniowych w matematyce, ponadto upowszechnił wykorzystanie systemu dziesiętnego i określił znaczenie zera w obliczeniach.

### Przykład 1.1.

Przygotowywanie potraw odbywa się według dokładnej instrukcji. Wiemy, jakie składniki są nam potrzebne oraz co chcemy przygotować — to specyfikacja naszego zadania. Wiemy również, jakie czynności krok po kroku trzeba wykonać, aby uzyskać oczekiwany efekt — to lista kroków lub opis słowny algorytmu. Z algorytmami spotykasz się każdego

dnia, definiujesz ich specyfikację, podając dane i wyniki, ponadto zapisujesz je w sposób czytelny i jednoznaczny, najczęściej jako listę kroków. Każda instrukcja to zapis algorytmu.

### Ćwiczenie 1.1.

Podaj przykłady algorytmów związanych z życiem codziennym lub innymi — poza informatyką — przedmiotami szkolnymi. Określ specyfikację tych algorytmów, czyli opisz dane i wyniki. Podaj krok po kroku czynności, które trzeba wykonać, aby otrzymać oczekiwany wynik.

## Etapy rozwiązywania zadań

Realizacja zadań za pomocą komputera przebiega etapami. **Pierwszy etap — specyfikacja zadania — jest Ci już znany ze szkoły podstawowej.** Poniżej wyszczególnione są wszystkie etapy, które prowadzą do rozwiązania problemu z wykorzystaniem komputera.

### 1. Określenie problemu, czyli specyfikacji zadania:

- dane (dane wejściowe),
- wyniki (dane wyjściowe).

Określając dane wejściowe i wyniki, podajemy również ich typ (czyli określamy rodzaj danych, na przykład liczby całkowite, liczby rzeczywiste, znaki, teksty) oraz sposób prezentacji. Ważny jest ponadto związek między nimi, a więc wyszczególnienie warunków, jakie muszą spełniać.

### 2. Definicja modeli i pojęć oraz zbadanie, czy analizowany problem ma rozwiązanie.

**Na podstawie uzyskanych informacji wybieramy odpowiedni algorytm rozwiązujący problem.** Istnieje wiele sposobów wykonania danego zadania, z czego wynika, że można skonstruować wiele algorytmów rozwiązujących określony problem. Na tym etapie analizujemy problem i dokonujemy wyboru algorytmu, który ma prowadzić do rozwiązania zgodnego ze specyfikacją.

### 3. Zapisanie algorytmu w wybranej postaci:

- opisu słownego,
- listy kroków,
- schematu blokowego,
- pseudokodu,
- programu
- i innych.

Jeśli chcemy użyć komputera, musimy skonstruować algorytm w postaci programu w wybranym języku programowania. Pozostałe formy mogą, ale nie muszą powstać jako pomoc w procesie pisania programu.

W szkole podstawowej pojawiły się już: opis słowny algorytmu, lista kroków, schemat blokowy oraz zapis algorytmu w postaci programów w wybranych przez nauczyciela wizualnych i tekstowych językach programowania.

### 4. Testowanie rozwiązania poprzez wykonywanie obliczeń na komputerze, w tym

testowanie programu, który jest implementacją algorytmu, dla różnych danych. Algorytm powinien działać dla dowolnych, zgodnych ze specyfikacją, danych

wejściowych, dając poprawne wyniki. Jest to jeden ze sposobów na wykrywanie błędów. Nie służy do sprawdzania poprawności algorytmu. Algorytm jest poprawny, jeśli dla każdego danych wejściowych jest skończony, a uzyskane wyniki są poprawne, czyli zgodne ze specyfikacją zadania.

- 5. Analiza własności wybranego algorytmu**, w tym ocena efektywności przyjętego rozwiązania, złożoności obliczeniowej (czasowej — określającej czas działania algorytmu — i pamięciowej — określającej, ile pamięci potrzeba do realizacji danego algorytmu) oraz błędów zaokrągleń wynikających z obliczeń na liczbach przybliżonych. Wybierany jest algorytm, który nie tylko daje poprawne dane wyjściowe, ale również jest efektywniejszy od innych rozwiązań postawionego problemu. Idealnym wynikiem naszych działań jest znalezienie optymalnego algorytmu, jednak nie zawsze jest to możliwe. Algorytm optymalny rozwiązuje problem w najkrótszym czasie, czyli przez wykonanie najmniejszej liczby operacji.

### Przykład 1.2.

Wiemy, czym są lata przestępne. To wtedy rok ma 366 dni, a nie 365. Ale czy zdajemy sobie sprawę z tego, kiedy rok będzie przestępny? Spróbujmy skonstruować algorytm, który sprawdzi, czy podany rok, mający nastąpić w przyszłości, jest przestępny.

Zacniemy od **określenia specyfikacji**. Dany mamy rok, który dopiero nastąpi. Wynikiem będzie informacja, czy podany rok jest przestępny.

Kolejnym krokiem jest **analiza problemu**. Musimy się dowiedzieć, jakie warunki powinien spełniać taki rok. Wiemy, że ma dopiero nastąpić, mamy więc do czynienia z kalendarzem gregoriańskim, który dokładnie określa te warunki.

**Konstruujemy więc algorytm**. Jeśli rok jest podzielny przez 4 i jednocześnie nie jest podzielny przez 100 lub jeśli rok jest podzielny przez 400, mamy rok przestępny, w przeciwnym razie rok nie jest przestępny.

Przejdźmy do **testowania algorytmu**. Na przykład rok 2020 jest podzielny przez 4 i niepodzielny przez 100 — jest więc przestępny. Kolejnym przykładem jest rok 2100 — podzielny przez 4 i podzielny przez 100, ponadto niepodzielny przez 400. Czy to jest rok przestępny? Podaj więcej przykładów.

### Zadanie 1.1.

Pracownik pewnej firmy otrzymuje miesięczne wynagrodzenie zasadnicze w wysokości  $k$  złotych. Dodatkowo co dwa miesiące uzyskuje premię w wysokości 10% miesięcznej płacy podstawowej. Wynagrodzenie zasadnicze wzrasta co trzy miesiące o 2%. Pracownik otrzymuje wynagrodzenie wraz z premią zawsze ostatniego dnia danego miesiąca. Jaką płacę uzyska pracownik po sześciu miesiącach i po roku? Podaj specyfikację zadania, przeanalizuj problem i przedstaw sposób (czyli opis słowny algorytmu) obliczenia wynagrodzenia pracownika. Przetestuj skonstruowany algorytm dla przykładowych danych.

## ..... Temat 2. Sposoby przedstawiania algorytmów

Istnieje wiele sposobów przedstawiania algorytmów. Wybór odpowiedniej metody powinien zależeć od rozwiązywanego problemu. Do wykonywanego zadania dobrać taki sposób reprezentowania algorytmu, który najdokładniej i najczytelniej pokaże jego przebieg. Najważniejsze metody przedstawiania algorytmów wykorzystywane w tym podręczniku to:

- lista kroków,
- schemat blokowy,
- program w tekstowym języku programowania Python.

Do prezentowania algorytmów można stosować też inne metody, na przykład opis słowny, pseudokod, drzewo algorytmu, drzewo wyrażenia, inne języki programowania. **Metody zastosowane w podręczniku poznaliście już w szkole podstawowej.** Poniżej przypomnimy najważniejsze informacje na temat listy kroków i schematu blokowego. Język programowania Python zostanie wprowadzony w kolejnych rozdziałach od początku, ponieważ w szkole podstawowej mógł pojawić się inny tekstowy język programowania.

### 2.1. Lista kroków algorytmu

Jednym z najczęściej stosowanych sposobów prezentacji algorytmów jest lista kroków. Forma ta umożliwi dokładne przedstawienie kolejnych operacji realizowanych przez algorytm, z uwzględnieniem powtarzania działań, zakończenia algorytmu po spełnieniu określonych warunków czy zmiany kolejności wykonywanych operacji. Lista kroków jest listą numerowaną, w której każdy krok algorytmu ma przypisany numer. Kroki algorytmu zawarte w liście realizuje się kolejno, należy jednak uwzględniać zmianę kolejności spowodowaną wykonaniem polecenia przejścia do kroku o podanym numerze. Umożliwia to na przykład realizację powtarzania operacji w algorytmie.

W podręczniku w listach kroków zastosowano matematyczne symbole operacji i warunków.

#### Przykład 2.1.

Skonstruujemy algorytm w postaci listy kroków rozwiązujący **równanie liniowe**  $ax + b = 0$ , które rozwiązywaliście na lekcjach matematyki w szkole podstawowej. Zaczniemy od określenia **specyfikacji zadania**.

#### Specyfikacja:

**Dane:** dowolne liczby rzeczywiste:  $a$ ,  $b$ .

**Wynik:** wartość rzeczywista pierwiastka równania liniowego:  $x$  lub komunikat informujący o braku rozwiązania bądź uzyskaniu nieskończenie wielu rozwiązań.

Kolejnym etapem będzie **przeprowadzenie analizy rozwiązywanego problemu**. Tworząc algorytm, należy rozpatrzyć wszystkie możliwe sytuacje. W przypadku równania liniowego rozwiązanie zadania zależy od wartości współczynników  $a$  i  $b$ , co przedstawia się następująco:

$$a \neq 0: \quad x = \frac{-b}{a},$$

$a = 0$  i  $b = 0$ : rozwiązaniem jest każda liczba rzeczywista,

$a = 0$  i  $b \neq 0$ : równanie sprzeczne.

Po wykonaniu analizy zadania możemy przejść do konstruowania algorytmu.

**Algorytm w postaci listy kroków:**

**Krok 1.** Jeśli  $a \neq 0$ , oblicz  $x = \frac{-b}{a}$ , wypisz wynik  $x$  i zakończ algorytm.

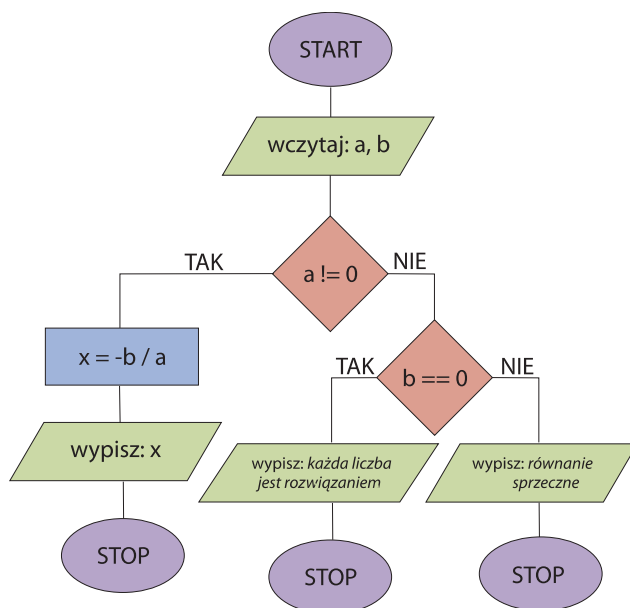
**Krok 2.** (W tym przypadku  $a = 0$ ). Jeśli  $b = 0$ , wypisz komunikat „każda liczba jest rozwiązaniem”, w przeciwnym razie wypisz komunikat „równanie sprzeczne”. Zakończ algorytm.

## 2.2. Schemat blokowy algorytmu

Schemat blokowy to graficzny sposób prezentacji algorytmu. Składa się z elementów zwanych blokami. Poszczególne części schematu należy łączyć, wykorzystując strzałki, które określają porządek wykonywania operacji.

### Przykład 2.2.

Na rysunku 2.1 przedstawiono przykładowy schemat blokowy algorytmu rozwiązującego równanie liniowe  $ax + b = 0$ , zapisanego w przykładzie 2.1 w postaci listy kroków.



**Rysunek 2.1.** Schemat blokowy algorytmu rozwiązującego równanie liniowe  $ax + b = 0$ . Zastosowano operatory relacyjne języka Python:  $!$  = oznacza różne,  $==$  oznacza równe

Przeanalizuj zaproponowane rozwiązanie widoczne na rysunku 2.1. W schemacie zastosowano operatory relacyjne języka Python (tabela 7.5). Czy warunki są zgodne z przeprowadzoną wcześniej analizą? Czy potrafisz określić znaczenie poszczególnych bloków schematu?

W schemacie blokowym można stosować matematyczne symbole operacji i warunków, operatory danego języka programowania lub opisy słowne wykonywanych operacji. Zapis jednak musi być jednoznaczny i dokładnie określać wykonywane działania. Dla ułatwienia w podręczniku w schematach blokowych wykorzystywane są operatory języka Python, w którym będziesz zapisywać programy realizujące omawiane algorytmy.

## 2.3. Program w języku programowania wysokiego poziomu

Jednym z najważniejszych sposobów przedstawiania algorytmów jest program napisany w języku programowania wysokiego poziomu. Praktyczna implementacja algorytmu, jaką jest napisanie programu, ułatwia jego testowanie dla różnych danych wejściowych. Ponadto jest to sposób zrozumiały dla komputera.

### Przykład 2.3.

Poniżej przedstawiono programy w różnych tekstowych językach programowania realizujące algorytm opisany w przykładach 2.1 i 2.2, rozwiązujący równanie liniowe  $ax + b = 0$ .

#### **Program w języku Python:**

```
def rownanie_linowe(a, b):
    if a != 0:
        x = -b / a
        print("x = ", x)
    elif b == 0:
        print("każda liczba jest rozwiązaniem")
    else: print("równanie sprzeczne")

rownanie_linowe(-2, 5)
```

#### **Program w języku C++:**

```
#include <iostream>
using namespace std;
int main()
{
    double a, b, x;
    cout << "podaj a, b: " << endl;
    cin >> a >> b;
    if (a != 0) { x = -b / a; cout << "x = " << x << endl; }
    else if (b == 0) cout << "każda liczba jest rozwiązaniem" << endl;
        else cout << "równanie sprzeczne" << endl;
    return 0;
}
```

### Program w języku Pascal:

```
program rownanie liniowe;
var a, b, x: real;
begin
  writeln('podaj a, b:');
  readln(a, b);
  if a <> 0 then begin x := -b / a; writeln('x = ', x) end
  else if b = 0 then writeln('każda liczba jest rozwiązaniem')
  else writeln('równanie sprzeczne')
end.
```

Przeanalizuj realizację algorytmu w różnych językach programowania.

#### Ćwiczenie 2.1.

W wybranym przez Ciebie języku przetestuj działanie algorytmu opisanego w tym temacie.

#### Zadanie 2.1.

Algorytm Euklidesa wyznacza największy wspólny dzielnik dwóch liczb naturalnych. Można go realizować z wykorzystaniem reszty z dzielenia lub za pomocą odejmowania. Algorytm ten pojawił się w szkole podstawowej, więc jest Ci już znany. Określ specyfikację tego algorytmu, wybierz metodę i przedstaw go za pomocą dowolnej notacji omówionej w tym temacie. Następnie przetestuj go dla przykładowych danych.

## ..... Temat 3. Klasyfikacja tekstowych języków programowania

### Definicja

**Języki programowania** to specjalne języki przeznaczone do formułowania algorytmów w taki sposób, aby były zrozumiałe dla komputera. Możemy je podzielić na dwie grupy:

- **języki wewnętrzne** — każde słowo zapisywane jest w postaci ciągu 0 i 1 o ustalonej długości; słowo może być interpretowane jako liczba lub rozkaz;
- **języki zewnętrzne** — języki stworzone z myślą o zapisywaniu algorytmów przez człowieka w sposób zrozumiały dla komputera; dzielimy je na języki niskiego i wysokiego poziomu.

Konieczność stworzenia języków zewnętrznych została podyktowana tym, że zapisywanie algorytmów bezpośrednio w języku wewnętrznym z wykorzystaniem systemu binarnego jest uciążliwe. Dlatego programista zapisuje algorytm w postaci programu w wybranym języku zewnętrznym, który następnie jest tłumaczony na język wewnętrzny komputera.

Do **języków zewnętrznych niskiego poziomu** zaliczamy języki zorientowane maszynowo, czyli ściśle związane z określonym typem procesora, a więc również językiem wewnętrznym.



nym komputera. Należą do nich asemblery, których używa się zwykle do konstruowania „wstawek” w programach pisanych w językach wysokiego poziomu.

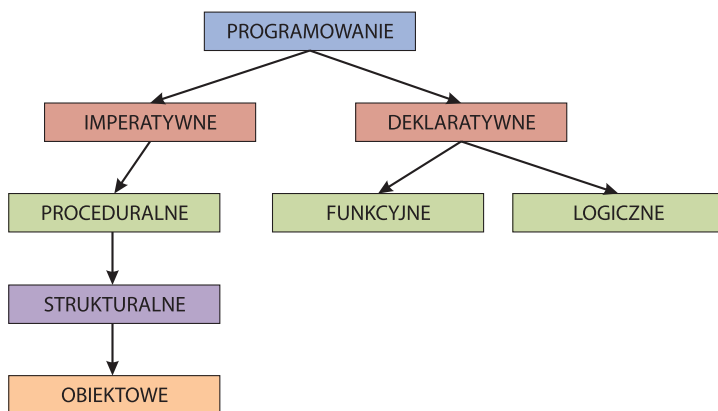
**Języki zewnętrzne wysokiego poziomu** służą do pisania programów przez programistów. Ich struktura jest czytelna dla człowieka i pozwala w prosty sposób zapisywać konstrukcje algorytmiczne pojawiające się w rozwiązywanym problemie.

**Translacja** to proces tłumaczenia programu z języka źródłowego (na przykład języka zewnętrznego wysokiego poziomu) na język wynikowy (na przykład język wewnętrzny) zrozumiały dla procesora.

Istnieją dwa typy programów realizujących tłumaczenie:

- **kompilatory** — najpierw cały program napisany w języku programowania jest tłumaczony na język wewnętrzny, a następnie wykonywany;
- **interpretery** — każde polecenie jest kolejno tłumaczone na język wewnętrzny i wykonywane przez komputer.

Na rysunku 3.1 przedstawiono **podstawowy podział języków programowania** na dwie grupy: języki imperatywne i deklaratywne.



**Rysunek 3.1.** Podstawowy podział języków programowania

**Języki imperatywne** charakteryzują się tym, że:

- używają rozkazów określających czynności, które komputer ma wykonywać;
- program jest listą rozkazów do wykonania, stąd nazwa „imperatywne”, czyli rozkazowe;
- mówią komputerowi, jak ma osiągnąć wynik, choć nie określają, jaki to ma być wynik.

W programowaniu imperatywnym program jest listą instrukcji, a instrukcje wykonywane są kolejno z możliwością wielokrotnego powtarzania pewnych czynności zapisanych raz, czyli wykonywania pętli (iteracji).

**Programowanie proceduralne** to najstarszy typ programowania imperatywnego. W kodzie programu wydzielone są fragmenty zwane podprogramami (procedury, funkcje, metody,

operacje), które mogą być wielokrotnie uruchamiane. Ten typ programowania przyczynił się do powstania techniki programowania *bottom-up*, polegającej na pisaniu kodu od małych fragmentów do coraz większych i w końcu do stworzenia całego programu. Przykładami języków proceduralnych są asemblery, Basic, Fortran, Pascal, C, C++, Object Pascal, Lisp, Logo, Python, Ruby. Te języki należą do grupy języków imperatywnych. Zauważ, że prawie wszystkie wymieniane języki można zaliczyć do różnych grup. Są to języki ogólnego przeznaczenia, które mają szerokie zastosowanie, przez co rozwijają się w różnych kierunkach.

### Przykład 3.1.

Przeanalizuj przykładowy program napisany w języku proceduralnym Pascal.

#### **Program w języku Pascal:**

```
program suma;
var i, n, s: integer;
begin
  readln(n);
  s := 0;
  for i := 1 to n do
    s := s + i;
  writeln(s)
end.
```

Powyższy program oblicza sumę kolejnych liczb naturalnych 1, 2, ...,  $n$ , dla  $n$  podanego z klawiatury, a następnie wypisuje obliczony wynik.

**Programowanie strukturalne** to rozwinięcie programowania proceduralnego. Korzysta z możliwości zapisywania każdego algorytmu za pomocą podstawowych struktur, takich jak sekwencja, instrukcje warunkowe, pętle, funkcje. Ten typ programowania umożliwia stosowanie techniki *top-down*, która jest odwróceniem techniki *bottom-up*. Polega ona na dzieleniu całego zadania na mniejsze części zgodnie z przewidywaną strukturą pisanego programu oraz na wypełnianiu tej struktury podstawowymi poleceniami. Przykładami języków strukturalnych są: Pascal, C, C++, Object Pascal, Python, Ruby.

**Programowanie obiektowe** to rozwinięcie programowania strukturalnego. Programy definiuje się tutaj za pomocą obiektów — elementów łączących stan (czyli dane nazywane polami) i zachowanie (czyli metody). Program napisany obiektowo wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań. Uważa się, że ten typ programowania dobrze odzwierciedla sposób, w jaki ludzie myślą o świecie — widzimy obiekty i ich cechy (czyli pola), oraz operacje, które na nich możemy wykonywać. Jeśli obiektem jest szkoła, to polami są uczniowie i nauczyciele, a wszystko, co robimy w szkole, to metody. Do języków obiektowych zaliczyć można języki: C++, Object Pascal, C#, Java, Python, Ruby.

### Przykład 3.2.

Przykładowy program w języku obiektowym Java.

#### **Program w języku Java:**

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Witaj");
    }
}
```

Ten program wypisuje na ekranie komunikat „Witaj”.

**Języki deklaratywne** to języki, w których programista podaje komputerowi pewne zależności oraz cele, jakie program ma osiągnąć. Opisują one, co ma zostać osiągnięte, ale nie podają, w jaki sposób.

**Programowanie funkcyjne** to odmiana programowania deklaratywnego. Program jest tutaj złożoną funkcją, która na podstawie danych wejściowych wylicza pewien wynik. To programowanie różni się od tych wcześniej omówionych przede wszystkim brakiem zmiennych i instrukcji pętli. Konstruowanie programów to składanie funkcji, zazwyczaj z wykorzystaniem rekurencji, czyli uruchamiania funkcji wewnątrz niej samej. Przykładami języków funkcyjnych są: Lisp, Logo, Python, Ruby.

**Programowanie logiczne** to kolejny rodzaj programowania deklaratywnego. Program jest tutaj zbiorem zależności (przesłanki) i pewnych stwierdzeń (cel), a wykonanie programu to próba udowodnienia celu na podstawie podanych przesłanek. Nie piszemy poleceń, a jedynie opisujemy, co wiemy i co chcemy uzyskać. Przykładem języka logicznego jest Prolog. Języki deklaratywne wykorzystywane są przede wszystkim w zakresie sztucznej inteligencji, systemów eksperckich i innych pokrewnych dziedzin.

### Przykład 3.3.

Poniżej podano przykładowy program w języku logicznym Prolog.

#### **Program w języku Prolog:**

```
ojciec(Jan, Adam).
ojciec(Adam, Krzysztof).
ojciec(Adam, Piotr).
dziadek(X, Z) :- ojciec(X, Y), ojciec(Y, Z).
?- dziadek(X, Krzysztof).
```

Przeanalizujmy kod tego programu:

- Jan jest ojcem Adama.
- Adam jest ojcem Krzysztofa.
- Adam jest ojcem Piotra.
- Jeśli X jest dziadkiem Z, to X jest ojcem Y i Y jest ojcem Z.
- X jest dziadkiem Krzysztofa. Kim jest X?

Spróbuj na podstawie podanych informacji odpowiedzieć na to pytanie i podać, kim jest X.

### Zadanie 3.1.

Odpowiedz na podane pytania:

- Jaka jest różnica między kompilacją a interpretacją kodu programu?
- Czym różni się programowanie imperatywne od deklaratywnego?
- Na czym polega różnica między technikami programowania *bottom-up* i *top-down*?

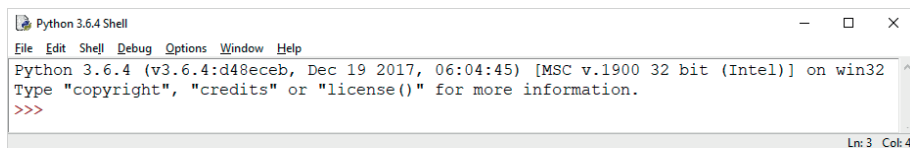
## ..... Temat 4. Python od pierwszych kroków

**Python** to język programowania wysokiego poziomu. Ma on przejrzystą i zwięzłą składnię. Programy pisane są w plikach tekstowych z rozszerzeniem *.py*. W języku Python istotna jest wielkość liter.

W kodach programów **wykorzystuje się wcięcia do wydzielenia bloków kodu**. Przyjęło się stosowanie czterech spacji, ale nie jest to konieczne. Jednak musimy być konsekwentni, co oznacza, że jeśli pierwsze wcięcie w funkcji miało trzy spacje, to kolejne wcięcia także muszą mieć trzy spacje. Wiersz bez wcięcia będzie się znajdował poza blokiem.

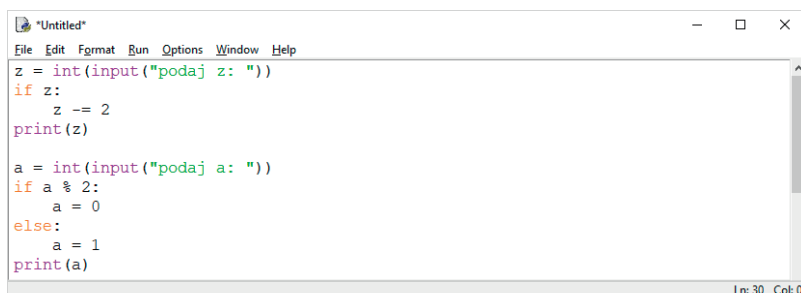
### 4.1. Środowisko programowe Pythona

**IDLE** to zintegrowane środowisko programistyczne tego języka dołączone do interpretera Pythona, które można znaleźć na stronie <https://www.python.org/>. Oprogramowanie jest darmowe, a interpretery są dostępne na wiele systemów operacyjnych. Zintegrowane środowisko programowe *IDLE* to zestaw narzędzi, które ułatwiają pisanie programów w Pythonie. Po uruchomieniu *IDLE* zgłasza się **tryb interaktywny**, przedstawiony na rysunku 4.1.



Rysunek 4.1. Tryb interaktywny środowiska programowania IDLE

Aby przejść do edycji nowego programu, należy z menu *File* wybrać polecenie *New File*. Otworzy się okno edytora przeznaczone do pisania programów — **tryb skryptowy** — co widać na rysunku 4.2.



Rysunek 4.2. Tryb skryptowy środowiska programowania IDLE

Pole ksi k

Istnieją też inne środowiska, w których można pisać programy w tym języku, na przykład *PyCharm*, edytor *Geany*, edytor *Mu*. Można również korzystać ze środowisk udostępnianych online. Być może na lekcjach informatyki w szkole podstawowej korzystaliście z różnych środowisk programowania w języku Python.

Aby rozpocząć pisanie programu w określonym języku programowania, powinno się poznać jego podstawowe polecenia. Bez znajomości „słów” nie jesteśmy w stanie porozumiewać się w żadnym języku. Zapoznaj się z podanymi poniżej informacjami i przeanalizuj przykłady. Rozpocznij pisanie swoich pierwszych programów w języku Python w środowisku wybranym przez nauczyciela.

## 4.2. Operacje wejścia-wyjścia

W tabeli 4.1 podano **podstawowe operacje wejścia-wyjścia języka Python**.

**Tabela 4.1.** Podstawowe operacje wejścia-wyjścia w języku Python

Polecenie	Opis polecenia
<code>print("tekst", zmienna)</code>	Wypisanie komunikatu na ekranie
<code>a = float(input("podaj liczbę rzeczywistą: "))</code> <code>b = int(input("podaj liczbę całkowitą: "))</code> <code>c = input("podaj tekst: ")</code>	Wprowadzanie wartości zmiennych z klawiatury (zmienne są tworzone przez przypisanie wartości)

### Przykład 4.1.

Przykład pokazuje zastosowanie operacji wejścia-wyjścia języka Python. Po wprowadzeniu daty urodzenia, kolejno dnia, miesiąca i roku wypisywana jest podana data w odpowiednim formacie.

```
d = int(input("podaj dzień: "))
m = input("podaj miesiąc: ")
r = int(input("podaj rok: "))
print(d, m, r, "r.")
```

Dla przykładowych danych program powinien działać następująco:

*podaj dzień: 23*

*podaj miesiąc: marca*

*podaj rok: 2008*

*23 marca 2008 r.*

### Ćwiczenie 4.1.

Napisz program w języku Python, w którym wypiszesz na ekranie komunikat:

*MÓJ PIERWSZY PROGRAM W JĘZYKU PYTHON.*

## Ćwiczenie 4.2.

Napisz program w języku Python, w którym utworzysz zmienne różnego typu poprzez przypisanie im wartości z klawiatury, a następnie wypiszesz wartości tych zmiennych na ekranie. Przykładowy wynik działania programu powinien być taki:

*podaj wartość zmiennej rzeczywistej: 25.14*

*podaj wartość zmiennej całkowitej: 378*

*podaj wartość zmiennej tekstowej: Python*

*liczba rzeczywista = 25.14*

*liczba całkowita = 378*

*tekst = Python*

Zwróć uwagę na zapis liczby rzeczywistej.

## 4.3. Komentarze

**Komentarz** to tekst zawarty w kodzie programu, który nie jest analizowany przez interpreter. Wykorzystywany jest on do komentowania programu, w którym został umieszczony. Aby tekst został potraktowany jako komentarz, oznacza się go odpowiednimi znakami. Istnieją dwa rodzaje komentarzy, co zostało przedstawione w tabeli 4.2. Stosuj komentarze zawsze, gdy chcesz dopisać swoją uwagę, zrobić notatkę lub wyłączyć z działania programu fragment kodu.

Tabela 4.2. Komentarze w języku Python

Polecenie	Opis polecenia
<code># komentarz</code>	Jednowierszowy komentarz
<code>""" komentarz """</code> <code>''' komentarz '''</code>	Wielowierszowe komentarze ograniczone przez " lub '

## 4.4. Operatory arytmetyczne

**Operatory arytmetyczne** (tabela 4.3) to operatory dwuargumentowe realizujące operacje arytmetyczne.

Tabela 4.3. Zestawienie operatorów arytmetycznych

Polecenie	Opis polecenia	Polecenie	Opis polecenia
<code>+</code>	Dodawanie	<code>//</code>	Dzielenie całkowite
<code>-</code>	Odejmowanie	<code>%</code>	Reszta z dzielenia
<code>*</code>	Mnożenie	<code>**</code>	Potęgowanie
<code>/</code>	Dzielenie		

### Przykład 4.2.

Poniżej podano przykłady zastosowania i poprawnego zapisu operacji arytmetycznych:

```
a = 5 + 3 - b
b = a / 4
c = a * (b - 1)
d = 2 ** a
```

W wyrażeniach możemy stosować nawiasy.

### Ćwiczenie 4.3.

Napisz program, w którym wykonasz działania podane w przykładzie 4.2 dla danych wprowadzanych z klawiatury. Przykładowy program powinien mieć taką postać:

```
b = int(input("podaj b: "))
a = 5 + 3 - b
print("a = ", a)
```

Po uruchomieniu programu możesz go przetestować:

```
podaj b: 7
a = 1
```

### Zadanie 4.1.

Napisz programy, w których wykonasz działania podane poniżej dla danych wprowadzanych z klawiatury:

a)  $a = b^3 + 5$ ,      b)  $a = \frac{4-b}{5}$ ,      c)  $a = \frac{2^3 + b}{3^2}$ .

## ..... Temat 5. Złożone operatory przypisania

**Złożone operatory przypisania** stosuje się do zapisywania wyrażeń  $X = X \cdot Y$  w postaci  $X \cdot = Y$ , gdzie  $\cdot$  to operator dwuargumentowy. Do najważniejszych z nich należą:

`+=`   `-=`   `*=`   `/=`   `//=`   `%=`   `**=`

### Przykład 5.1.

W tabeli 5.1 podano przykłady zastosowania złożonych operatorów przypisania w konstruowaniu wyrażeń.

**Tabela 5.1.** Przykłady zastosowania złożonych operatorów przypisania

Wyrażenie	Zapis wyrażenia bez złożonego operatora przypisania
<code>a *= 4 * b - 1</code>	<code>a = a * (4 * b - 1)</code>
<code>m %= 3 - n / 2</code>	<code>m = m % (3 - n / 2)</code>
<code>b -= 3 + d * 4</code>	<code>b = b - (3 + d * 4)</code>

Zwróć uwagę na kolejność wykonywanych działań.

### Ćwiczenie 5.1.

Napisz program, w którym wykonasz działania podane w przykładzie 5.1 dla danych wprowadzanych z klawiatury. Przykładowy program powinien wyglądać tak:

```
a = int(input("podaj a: "))
b = int(input("podaj b: "))
a *= 4 * b - 1
print("a = ", a)
```

Po uruchomieniu programu możesz go przetestować:

*podaj a: 2*

*podaj b: 3*

*a = 22*

### Zadanie 5.1.

Napisz programy, w których wykonasz działania podane poniżej z wykorzystaniem złożonych operatorów przypisania dla danych wprowadzanych z klawiatury:

$$\text{a) } a = a \cdot (3^3 + b), \quad \text{b) } k = k - (i \cdot 5 + 2), \quad \text{c) } w = \frac{w}{4 + z^2}.$$

### Zadanie 5.2.

Podaj specyfikację zadania i napisz program wykonujący podstawowe działania arytmetyczne na liczbach wprowadzonych z klawiatury. Przetestuj program dla różnych danych. Poniżej pokazano przykładowy wynik działania programu:

*podaj a = 3*

*podaj b = 2*

*3 + 2 = 5*

*3 - 2 = 1*

*3 \* 2 = 6*

*3 / 2 = 1.5*

*3 // 2 = 1*

*3 % 2 = 1*

*3 \*\* 2 = 9*

### Zadanie 5.3.

Podaj specyfikację zadania i skonstruuuj algorytm w postaci schematu blokowego oraz programu, obliczający średnią arytmetyczną trzech liczb całkowitych  $a$ ,  $b$  i  $c$  wprowadzonych z klawiatury.



## ..... Temat 6. Funkcje matematyczne w języku Python

W języku Python mamy dostęp do wielu **funkcji matematycznych**, które są niezbędne przy wykonywaniu niektórych obliczeń. Zawarte są one w bibliotece `math`, którą należy zadeklarować.

W tabeli 6.1 przedstawiono dostęp do tej biblioteki i zestawienie podstawowych funkcji matematycznych przez nią oferowanych.

**Tabela 6.1.** Zestawienie podstawowych funkcji matematycznych biblioteki `math`

Polecenie	Opis polecenia
<code>from math import *</code>	Dostęp do biblioteki matematycznej
<code>a = sqrt(49)</code>	Zastosowanie funkcji <code>sqrt()</code> , obliczającej pierwiastek kwadratowy, wynik: <code>a = 7</code>
<code>b = fabs(-8)</code>	Zastosowanie funkcji <code>fabs()</code> , obliczającej wartość bezwzględną, wynik: <code>b = 8</code>
<code>a = sin(1)</code> <code>b = cos(1)</code> <code>c = tan(1)</code>	Zastosowanie funkcji trygonometrycznych <code>sin()</code> , <code>cos()</code> i <code>tan()</code> (argumenty podajemy w radianach), wyniki: <code>a = 0.8414709848078965</code> <code>b = 0.5403023058681398</code> <code>c = 1.5574077246549023</code>

### Przykład 6.1.

Obliczmy wartość podanego wyrażenia w języku Python:

$$w = \sqrt{\frac{7}{a^3 + \cos(b)}}$$

Wartości zmiennych `a` i `b` wprowadzamy z klawiatury, a następnie wykonujemy obliczenia. Przyjrzyj się, w jaki sposób zastosowano funkcje matematyczne języka Python w programie.

```
from math import *
a = float(input("podaj liczbę a = "))
b = float(input("podaj liczbę b = "))
w = sqrt(7 / (a ** 3 + cos(b)))
print("wynik = ", w)
```

Przykładowe wyniki:

```
podaj liczbę a = 4
podaj liczbę b = 5
wynik = 0.3299884314684228
```

Przetestuj ten program dla różnych danych.

### Zadanie 6.1.

Średnią geometryczną dwóch liczb nieujemnych  $a$  i  $b$  nazywamy pierwiastek kwadratowy iloczynu tych liczb. Określ specyfikację zadania i skonstruuj algorytm w postaci programu, obliczający średnią geometryczną dwóch liczb wprowadzonych z klawiatury.

### Zadanie 6.2.

Poniżej podano wyrażenia, których wartość należy obliczyć, pisząc program w języku Python. Znasz już funkcje matematyczne w tym języku (tabela 6.1). Zastosuj je i oblicz wartości podanych wyrażen dla danych wprowadzanych z klawiatury.

$$\text{a) } w = a^3 + \cos(b) \cdot \sqrt{a+b}, \quad \text{b) } w = |a-b| + \sin(a) \cdot \sqrt{b},$$

$$\text{c) } w = \sqrt{\frac{3 + \sqrt{a \cdot b}}{|b^2 - 20|}}, \quad \text{d) } w = \sin\left(\frac{(a+b)^4}{\sqrt{11 + \sin(b)}}\right),$$

$$\text{e) } w = \left(\frac{\cos(a+1)}{|\sqrt{5-b}|}\right)^3.$$

## ..... Temat 7. Algorytmy z warunkami w języku Python

Na lekcjach informatyki w szkole podstawowej pojawiły się polecenia, które realizowały algorytmy z warunkami. Takie instrukcje występują zarówno w językach wizualnych, jak i tekstowych. Na rysunku 7.1 przedstawiono bloki warunkowe języka, który znasz ze szkoły podstawowej.



Rysunek 7.1. Instrukcje warunkowe w języku Scratch

Przykładem algorytmu z warunkami jest omówiony w temacie 2. algorytm rozwiązujący równanie liniowe  $ax + b = 0$ , przedstawiony w postaci listy kroków, schematu blokowego (rysunek 2.1) oraz programów w językach wysokiego poziomu: Python, C++ i Pascal.

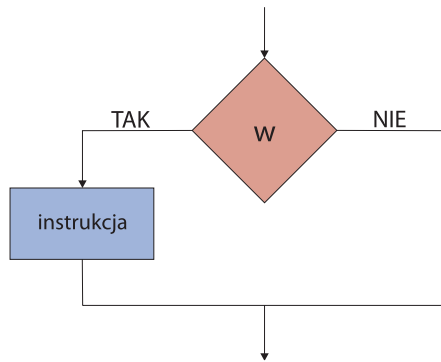
## 7.1. Instrukcje warunkowe

**Instrukcje warunkowe** znajdują zastosowanie w sytuacji, gdy konstruowany algorytm zawiera warunki, od których spełnienia zależy kolejność wykonywanych działań. W tabelach 7.1, 7.2, 7.3 i 7.4 zawarte są informacje dotyczące zapisu instrukcji warunkowych w języku Python. Na rysunkach 7.2, 7.3, 7.4 i 7.5 przedstawiono schematy blokowe tych instrukcji. Przeanalizuj zapis instrukcji i porównaj go ze schematami blokowymi.

**Tabela 7.1.** Instrukcja warunkowa skrócona

Polecenie	Opis polecenia	Znaczenie
<code>if w: instrukcja</code>	Instrukcja warunkowa skrócona	Jeśli warunek <i>w</i> jest spełniony, wykonaj instrukcję <i>instrukcja</i> , w przeciwnym razie kontynuuj program

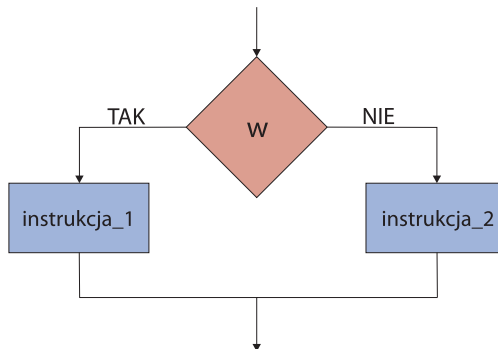
**Rysunek 7.2.** Schemat blokowy skróconej instrukcji warunkowej



**Tabela 7.2.** Instrukcja warunkowa pełna

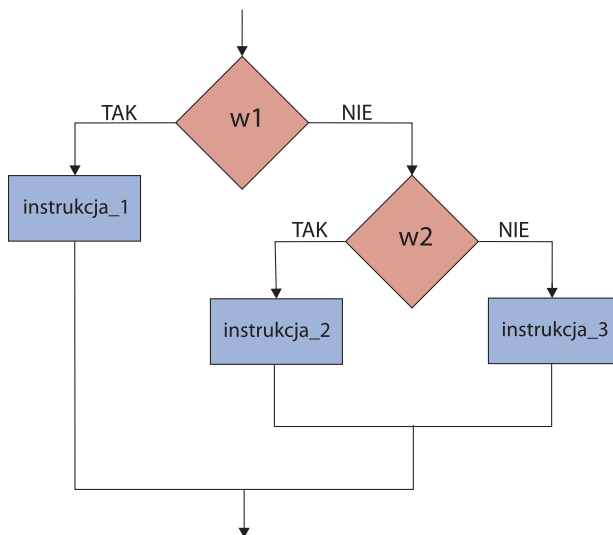
Polecenie	Opis polecenia	Znaczenie
<code>if w: instrukcja_1 else: instrukcja_2</code>	Instrukcja warunkowa pełna	Jeśli warunek <i>w</i> jest spełniony, wykonaj instrukcję <i>instrukcja_1</i> , w przeciwnym razie wykonaj <i>instrukcja_2</i>

**Rysunek 7.3.** Schemat blokowy pełnej instrukcji warunkowej



**Tabela 7.3.** Instrukcja warunkowa złożona

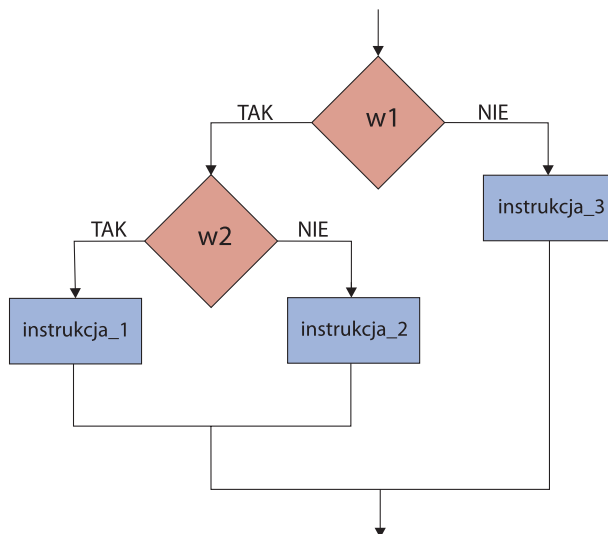
Polecenie	Opis polecenia	Znaczenie
<pre>if w1: instrukcja_1 elif w2: instrukcja_2 else: instrukcja_3</pre>	Złożona instrukcja warunkowa ( <b>alternatywa</b> )	Jeśli warunek <i>w1</i> jest spełniony, wykonaj instrukcję <i>instrukcja_1</i> , w przeciwnym razie, jeśli warunek <i>w2</i> jest spełniony, wykonaj <i>instrukcja_2</i> , w przeciwnym razie wykonaj <i>instrukcja_3</i>



**Rysunek 7.4.** Schemat blokowy złożonej instrukcji warunkowej

**Tabela 7.4.** Zagnieżdżenie instrukcji warunkowych

Polecenie	Opis polecenia	Znaczenie
<pre>if w1:     if w2: instrukcja_1     else: instrukcja_2 else: instrukcja_3</pre>	Zagnieżdżenie instrukcji warunkowych ( <b>koniunkcja</b> )	Jeśli warunek <i>w1</i> jest spełniony i warunek <i>w2</i> jest spełniony, wykonaj instrukcję <i>instrukcja_1</i> , w przeciwnym razie, jeśli spełniony jest tylko warunek <i>w1</i> , wykonaj <i>instrukcja_2</i> , a jeśli nie jest spełniony warunek <i>w1</i> , wykonaj <i>instrukcja_3</i>



**Rysunek 7.5.** Schemat blokowy przedstawiający zagnieżdżenie instrukcji warunkowych

W języku Python dowolna niezerowa wartość w przypadku wyrażeń liczbowych jest interpretowana jako prawda (czyli `True`), a wartość 0 jako fałsz (czyli `False`). W związku z tym w miejscu warunku może pojawić się zmienna lub wyrażenie, których wartość będzie decydować o tym, czy warunek jest spełniony.

## 7.2. Operatory relacyjne

W tabeli 7.5 podano operatory relacyjne, z których można korzystać przy zapisywaniu warunków. Zapoznaj się z nimi. Będziesz je stosować w większości programów.

**Tabela 7.5.** Zestawienie operatorów relacyjnych

Polecenie	Opis polecenia
<	Mniejsze
<=	Mniejsze lub równe
==	Równe
!=	Różne
>=	Większe lub równe
>	Większe

## A

adiustacja, 159  
adware, 68  
aktualizowanie spisu treści, 146  
algorytm, 10  
    Euklidesa, 54, 57, 61  
    liniowy, 61  
algorytmika, 10  
algorytmy  
    analiza własności, 12  
    lista kroków, 13  
    przedstawianie, 13  
    rekurencyjne, 45, 48  
    schemat blokowy, 14  
    testowanie rozwiązania, 11  
    z warunkami, 26  
    zapisywanie, 11  
aliasing, 80, 103  
alternatywa, 33  
antialiasing, 80  
argumenty funkcji, 47  
automatyczny spis treści, 146

## B

Bézier Pierre Étienne, 99  
bezpieczeństwo, 63  
biblioteka  
    math, 25  
    turtle, 71  
bitmapa, 73, 86  
blokowanie reklam, 68  
bryły, 122  
    część wspólna, 123  
    dodawanie, 122, 123  
    odejmowanie, 122

## C

CAD, Computer Aided Design, 99  
ciąg Fibonacciego, 50, 51, 61  
ciągi  
    n-wyrazowe, 41  
    liczbowe, 41, 42  
ciasteczka, cookies, 68  
CSG, Constructive Solid Geometry, 121  
czcionka, 155

## D

dane  
    kontaktowe, 69  
    wejściowe, 11  
    wyjściowe, 11  
definicja  
    funkcji, 46  
    rekurencyjna, 53  
dodawanie  
    obiektów 3D, 111  
    podpisu, 148, 150  
    warstwy tła, 84  
Dokumenty Google, 144  
dokument tekstowy, 143  
    kolumny, 151  
    komentarze, 160  
    modyfikowanie stylu, 156  
    porównanie wersji, 161, 162  
    recenzje, 158, 160, 164  
    sekcje, 151, 153  
    szablony, 154, 157  
    śledzenie zmian, 158, 159  
    własne style, 154, 155  
dostęp do biblioteki matematycznej, 25  
drukarka 3D, 142

## E

edycja węzłów, 100, 101  
edytor  
    Geany, 21  
    Mu, 21  
    tekstu, 95, 145  
EPS, Encapsulated PostScript, 92

## F

Fibonacci Leonardo, 50  
format  
    BMP, 86  
    CPT, 79  
    EPS, 92  
    GIF, 86  
    GIMP, 88  
    JFIF, 89  
    JPEG, 87, 88  
    JPG, 90  
    PNG, 89  
    POV, 116  
    RTF, 132, 134  
    SVG, 92, 94, 95  
    TIFF, 89  
    XCF, 79  
formatowanie  
    dokumentu tekstowego, 153  
    prezentacji, 128  
formaty grafiki wektorowej, 91  
funkcja  
    range(), 35  
    fabs(), 25  
    sqrt(), 25  
funkcje, 46  
    matematyczne, 25  
    parametry aktualne, 47  
    parametry formalne, 47  
    rekurencyjne, 48, 49, 53  
    trygonometryczne, 25

## G

geometria trójwymiarowa, 107  
GIF, Graphics Interchange Format, 86  
GIMP, 73, 75  
graficzny interfejs użytkownika, GUI, 72  
grafika  
    bitmapowa, 103  
    komputerowa, 71, 72  
    płaska, 2D, 72  
    rastrowa, 72, 73, 80  
    SVG, 95  
    trójwymiarowa, 3D, 72, 105, 112  
    wektorowa, 91, 103  
        wady, 104  
        zalety, 104

## H

hasła, 69  
hiperłącza, 138, 147

## I

IDLE, 20  
    tryb interaktywny, 20  
    tryb skryptowy, 20  
iloczyn n-wyrazowego ciągu, 44, 45  
ilustracje, 147  
Indeks pojęć, 163  
informacja o kolorze, 95  
informatyka, 10  
Inkscape, 100, 102  
instrukcja warunkowa, 26, 27  
    pełna, 27  
    złożona, 28  
    if, 40  
instrukcja iteracyjna, 35, 41  
    for, 36  
    while, 37–39  
instrukcje, 17  
interpretery, 17  
iteracja, 17, 41, 61

## J

- jednostka
  - centimeters, 76
  - dpc, 74
  - dpi, 74
  - lpi, 74
  - ppc, 74
  - ppi, 74
- języki programowania, 16
  - deklaratywne, 17
  - imperatywne, 17
  - obiektywne, 18
  - podział, 17
  - proceduralne, 18
  - strukturalne, 18
  - wewnętrzne, 16
  - zewnętrzne, 16
- JFIF, JPEG File Interchange Format, 89
- JPEG, Joint Photographic Expert Group, 87

## K

- kadrowanie
  - obrazu, 75, 81, 82
  - w stałych proporcjach, 76
- kamera, 116, 117
- Kanwa, 112
- kartka świąteczna, 126
- keyloggers, 68
- klonowanie, 88
- kodowanie kolorów, 87
- kody sterujące, 134
- kolor, 95
- kolory indeksowane, 87
- kolumny, 151
- komentarze, 22, 160
- kompilatory, 17
- kompresja
  - bezstratna LZW, 89
  - JPEG, 87
  - stratna JPEG, 89

- koniunkcja, 33
- konspekt prezentacji, 131
- konstruktywna geometria brył, 121
- konto, 69
- kontur, 93
- krzywa Beziera, 99
- Kształt 3D, 110
- kula, 116

## L

- linia prosta, 99

## M

- modele 3D, 109
- Modeler, 106
- modyfikowanie stylu, 156

## N

- nagłówek, 144
- najmniejsza wspólna wielokrotność, NWW, 54, 57, 58
- największy wspólny dzielnik, NWD, 54, 55, 57
- narzędzie
  - Edycja węzłów, 100, 101
  - Kadrowanie, 75
  - Klonowanie, 88
  - Pióro, 101
  - Przesunięcie, 78
  - Różdzka, 81, 83
  - Skalowanie, 78, 85
  - Uniwersalne przekształcenie, 85
  - Wskaźnik, 101
  - Wycinanie, 94
  - Zaznaczenie, 101
- Następna strona, 153
- negacja, 33
- niestandardowe pokazy slajdów, 138
- Notepad++, 95
- Numer strony, 131, 144
- numery kolorów, 98



## O

- obiekty, 18
- obrót, 120
- ochrona
  - danych osobowych, 66, 68
  - wizerunku, 65
- odczyt koloru, 98
- Oderwane zaznaczenie, 77
- odwołania, 144, 146, 163
- opcje formatowania, 154
- operacje wejścia-wyjścia, 21
- operatory
  - arytmetyczne, 22
  - logiczne, 33
  - przypisania, 23
  - relacyjne, 29
- oprogramowanie
  - do blokowania reklam, 68
  - szpiegujące, 68
- orientacje stron, 154
- osobisty numer identyfikacyjny, PIN, 69

## P

- Paint 3D, 107, 108
  - dodawanie koloru, 108
  - dodawanie obiektów, 111
  - interfejs programu, 108
  - obiekt 3D, 109
  - szkic 3D, 110
- parametry
  - aktualne, 47
  - formalne, 47
- PESEL, 69
- Pędzel 3D, 110
- pętla, 17
  - while, 37–39
  - for, 36
- pętle w grafice 3D, 127
- pikseloza, 80
- PIN, personal identification number, 69
- PNG, Portable Network Graphics, 89

- podpisywanie
  - rysunku, 148
  - tabeli, 150
- podział
  - języków, 17
  - na kolumny, 152
  - strony, 144
  - tekstu na kolumny, 151
- pola, 18
- pole trójkąta, 35
- polecenie
  - and, 33
  - elif, 28
  - else, 28
  - fabs, 25
  - for, 36
  - if, 27, 40
  - input, 21
  - not, 33
  - or, 33
  - print, 21, 37
  - range, 35
  - return, 46
  - sin, 25
  - sqrt, 25
  - while, 37–39
- porównanie wersji dokumentu, 161, 162
- POV-Ray, 113
  - część wspólna, intersection, 123
  - geometria brył, 121
  - konstrukcja pętli, 127
  - konstruktywna geometria brył, 121
  - obiekty, 115
  - obserwowanie sceny, 117
  - suma, union, 123
  - transformacje brył, 120
  - wyrenderowany obraz kuli, 116
- PowerPoint, 128, 133, 136
- powielenie warstwy obrazu, 79
- poziomy konspektu, 130
- praca grupowa, 165
- prawo autorskie, 64

- prezentacje multimedialne, 71, 128
  - formatowanie, 128
  - konspekt, 129, 131
  - niestandardowy pokaz slajdów, 135, 138
  - parametry pokazu, 134
  - PowerPoint, 128, 133, 136
  - tworzenie, 128
  - wklejanie tekstu, 130
  - wstawianie hiperłącza, 138
  - wygaszanie, 139
- progowanie, 82
- program, 15, 17
  - Adobe Flash, 92
  - Adobe Illustrator, 92
  - AutoCAD, 92
  - CorelDRAW, 92
  - GIMP, 73
  - Inkscape, 92, 100
  - Modeler, 106
  - MS Word, 145
  - Notepad++, 95
  - Paint 3D, 107
  - PowerPoint, 128, 133, 136
  - POV-Ray, 113
  - Xara Xtreme, 92
- programy w języku
  - C++, 15
  - Java, 19
  - Pascal, 16
  - Prolog, 19
  - Python, 15
  - Scratch, 26
- programowanie
  - bottom-up, 18, 61
  - deklaratywne, 17
  - imperatywne, 17
  - obiektywne, 18
  - proceduralne, 18
  - strukturalne, 18
  - top-down, 18
  - wysokiego poziomu, 15

- projekt zespołowy, 165
- prymitywy, 107
- przechwycenie haseł, 68
- przekształcanie obrazu, 81
- pseudokod, 61
- punktor, 130
- Python, 20

## R

- radiosity, 106
- rasteryzacja, 103
- ray tracing, 106
- recenzje, 158, 160, 164
  - akceptacja zmian, 160
  - śledzenie zmian, 158
  - wersja końcowa, 159
  - komentarze, 160
  - porównanie wersji, 161
- rekurencja, 45
- renderowanie, 117
- rozdzielczość
  - obrazu, 73, 74
  - papieru fotograficznego, 76
- rozmiar slajdów, 134
- RTF, Rich Text Format, 134

## S

- scena trójwymiarowa, 115
- schemat blokowy, 14, 31
- sekcje, 151, 153
- skalowanie, 78, 80, 120
  - aktywnej warstwy, 78
  - grafiki wektorowej, 103
  - slajdu, 135
- slajdy, 128
  - niestandardowy pokaz, 135
  - skalowanie, 135
  - sortowanie, 133
  - zmiana rozmiaru, 134
- spam, 69
- specyfikacja zadań, 11

spis  
  ilustracji, 147  
  tabel, 149  
  treści, 144–147  
    hiperłącza, 147  
spyware, 68  
sterowanie formatowaniem tekstu, 134  
stożek, 119  
stożkowatość, 110  
style, 154, 155  
suma n-wyrazowego ciągu, 44  
SVG, Scalable Vector Graphics, 92, 94, 95  
szablon programu Word, 154, 157  
szkic 3D, 110

## Ś

śledzenie  
  promieni, 106, 113  
  zmian, 158, 159  
średnia geometryczna, 26  
środowisko programistyczne, 20

## T

tabele, 149  
  podpisywanie, 150  
tekstury, 111  
testowanie, 11  
TIFF, Tagged-Image File Format, 89  
transformacje  
  brył, 120  
  obrazu, 80  
  warstwy tła, 85  
translacja, 17, 120  
trójwymiarowa scena, 115  
tryb  
  incognito, 68  
  recenzji, 158  
tworzenie  
  grafiki 3D, 106  
  kolumn, 151  
  nowego stylu, 155

prezentacji, 128  
sceny, 117

## U

układ strony, 153  
układy współrzędnych, 107  
ustawa o prawie autorskim, 65  
ustawienia  
  ekranu, 113  
  wymiarów płótna, 84  
  wypełnienia i konturu, 93

## W

wady grafiki rastrowej, 104  
wektoryzacja, 103  
widok 3D, 110  
wielkość obrazu, 73  
wklejanie tekstu, 130  
Word, 145  
wstawianie  
  indeksu, 163  
  podpisu, 148  
  spisu  
    ilustracji, 149  
    tabel, 149  
    treści, 144–147  
  znaku podziału kolumny, 152  
wycinanie, 94  
wypełnienie, 93  
wypisywanie n-wyrazowego ciągu, 44  
wywołania rekurencyjne, 49, 52  
wzór  
  Herona, 35  
  rekurencyjny, 45

## Z

zaakceptowanie zmiany, 160  
zagnieżdżenie instrukcji, 29  
zalety grafiki rastrowej, 104  
zapisanie algorytmu, 11  
zasady bezpieczeństwa, 63

zaznaczenie, 100, 101  
fragmentu obrazu, 81, 83  
zmiana  
aktywnego koloru, 97  
formatowania, 155  
rozmiaru slajdów, 134  
znak podziału kolumny, 152

## Ź

źródła światła, 117

# PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion** 

Technologia informacyjna i informatyka to dziedziny, których wykorzystanie i dostępność stale rosną, a tempo zachodzących w nich zmian jest nieporównywalne z rozwojem innych dyscyplin. Umiejętność biegłego posługiwania się komputerem i urządzeniami peryferyjnymi oraz znajomość obsługi pakietów biurowych są obowiązkowe na rynku pracy. Wkrótce każdy z nas będzie musiał się również legitymować wiedzą z zakresu podstaw programowania, tworzenia grafiki czy zagadnień związanych z prawem w sieci.

## Informatyka w szkole ponadpodstawowej na poziomie podstawowym:

- jest przedmiotem obowiązkowym prowadzonym we wszystkich klasach
- nie jest przedmiotem maturalnym
- stanowi poziom podstawowy dla przedmiotu informatyka na poziomie rozszerzonym

Dzięki pracy z książką **Informatyka Europejczyka. Podręcznik dla szkół ponadpodstawowych. Zakres podstawowy. Część 1** nauczysz się szybko i sprawnie rozwiązywać problemy z użyciem komputera. Poznasz przyjazny i łatwy do opanowania język Python, co będzie świetnym wprowadzeniem do programowania. Opanujesz wiedzę z zakresu ochrony danych osobowych oraz prawa autorskiego — w efekcie będziesz bardziej świadomie i bezpiecznie korzystać z internetu. Dowiesz się, jak tworzyć atrakcyjne grafiki i prezentacje multimedialne, co z pewnością przyda Ci się przy przygotowywaniu referatów z historii, geografii i biologii. Znajomość zaawansowanych możliwości edytorów tekstu może okazać się zbawienna, gdy zechcesz napisać na przykład dłuższe wypracowanie i podczas przygotowań do matury z języka polskiego oraz potem — na studiach.

**W skład kompletnego pakietu edukacyjnego dla szkoły ponadpodstawowej wchodzi także podręczniki:**



Podręczniki z serii **Informatyka Europejczyka** ułatwią uczniom zdobywanie wiedzy i umiejętności podczas wykonywania ćwiczeń praktycznych, a nauczycielom — przekazywanie nowego materiału w interesujący i niebanalny sposób.

# Wciśnij Enter i do dzieła!

<http://edukacja.helion.pl>

<b>Helion</b> EDUKACJA	księgarnia internetowa
zamówienia telefoniczne	 <a href="http://helion.pl">http://helion.pl</a>
 <b>0 801 339900</b>	Helion SA ul. Kościuszki 1c, 44-100 Gliwice tel.: 32 230 98 63 e-mail: <a href="mailto:helion@helion.pl">helion@helion.pl</a> <a href="http://helion.pl">http://helion.pl</a>
 <b>0 601 339900</b>	
Informatyka w najlepszym wydaniu	

ISBN 978-83-283-6261-1



9 788328 362611