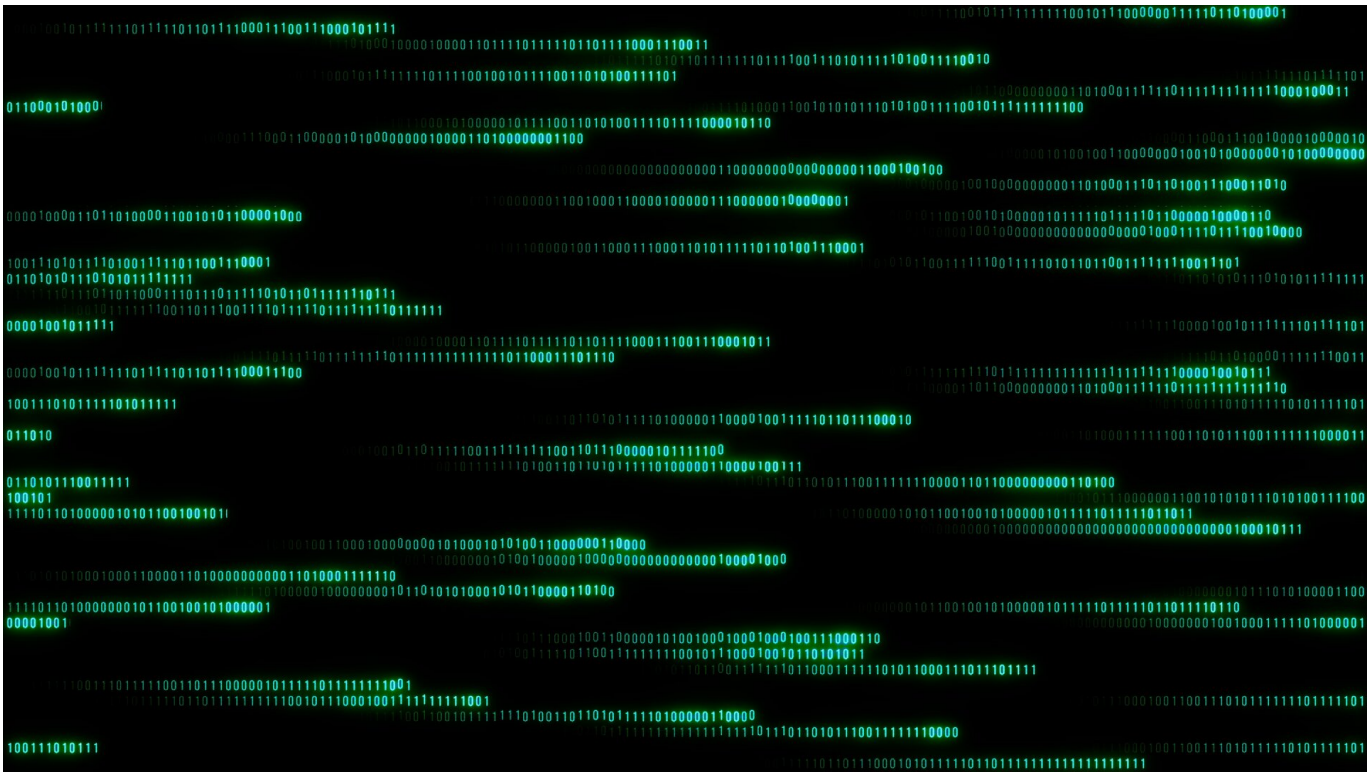
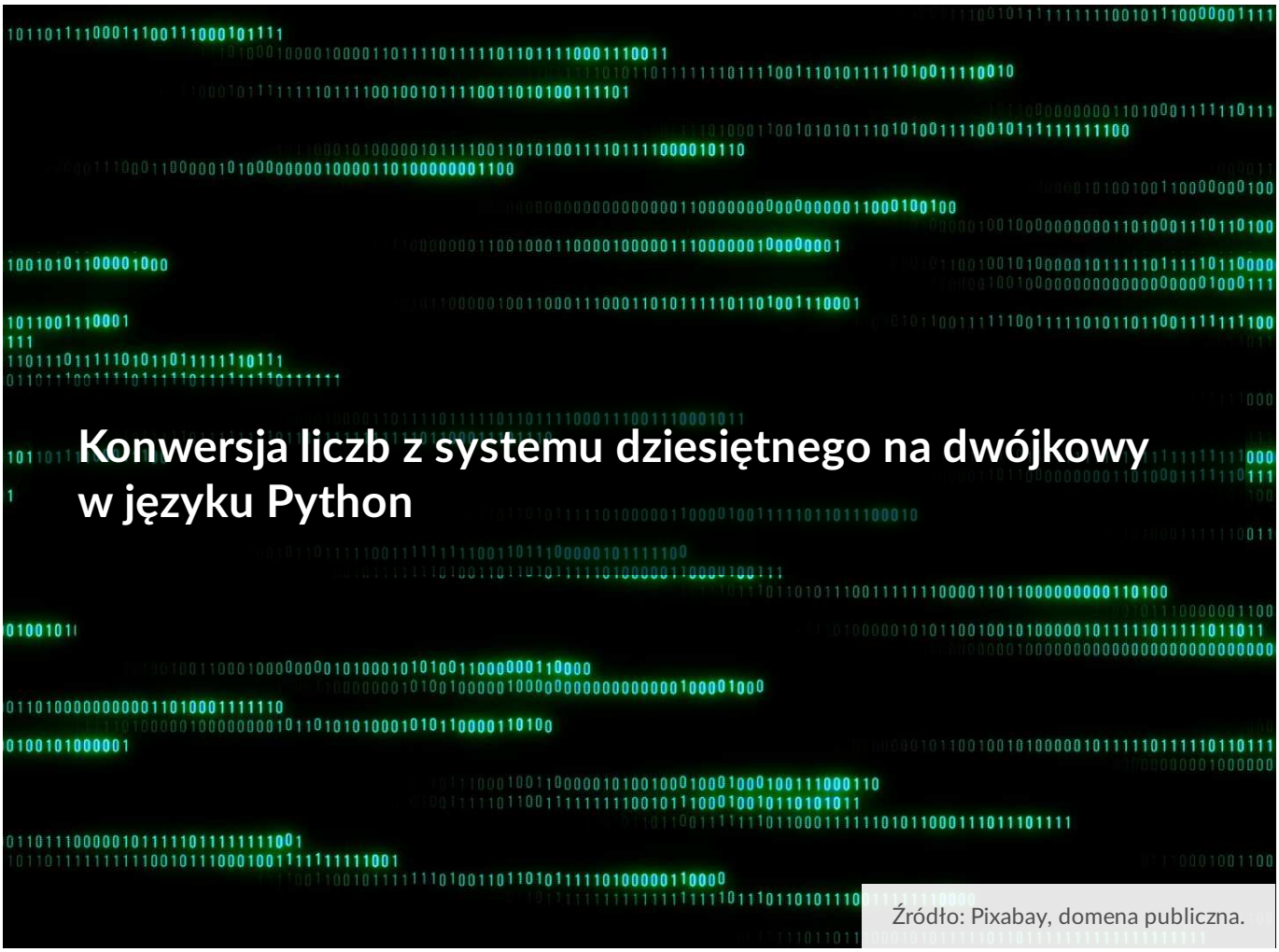


Konwersja liczb z systemu dziesiętnego na dwójkowy w języku Python



Źródło: Pixabay, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Poznaliśmy już algorytm konwersji liczb z systemu dziesiętnego na dwójkowy (binarny). Przedstawiliśmy ręczną metodę przeprowadzania takiej zamiany oraz pseudokod opisujący niezbędne czynności.

W tym e-materiale przejdziemy do implementacji tego zagadnienia w języku Python. Ma on wbudowane funkcje do konwertowania liczb pomiędzy różnymi systemami liczbowymi. Aby opanować to zagadnienie, musisz najpierw rozumieć, jakie wartości dana funkcja zwraca, i wiedzieć, jak otrzymać pożądany wynik, używając prostego algorytmu.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Konwersja liczb z systemu dziesiętnego na dwójkowy w języku C++,
- Konwersja liczb z systemu dziesiętnego na dwójkowy w języku Java.

Więcej zadań? Sięgnij do Konwersja liczb z systemu dziesiętnego na dwójkowy – zadania maturalne.

> **Twoje cele**

- Zrozumiesz, jakie są zależności pomiędzy dziesiętnym a dwójkowym systemem liczbowym.
- Wykorzystasz wbudowane funkcje języka Python do zamiany liczby dziesiętnej na odpowiednik binarny.
- Nauczysz się konwertować liczby z systemu dziesiętnego na dwójkowy.

Film samouczek

Polecenie 1

Obejrzyj film i opisz proces zamiany liczby dziesiętnej na binarną w pięciu krokach.

Trwa wczytywanie danych ..

Film dostępny pod adresem /preview/resource/RiZiznPbpf7T4

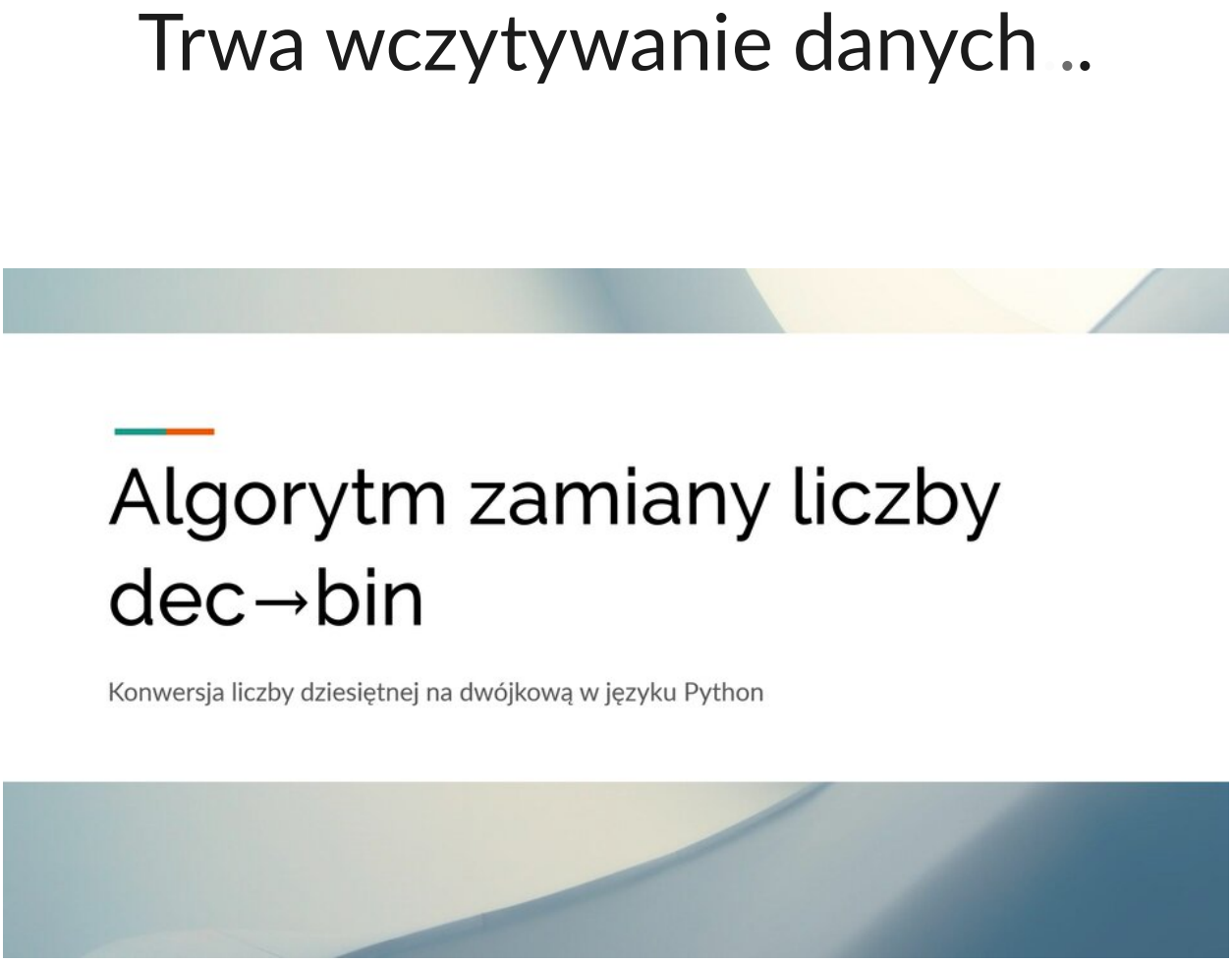
Film nawiązujący do treści materiału

Problem 1

Specyfikacja problemu:

Polecenie 2

Porównaj swoje rozwiązanie z animacją.



Film dostępny pod adresem /preview/resource/RQAQ0YxgdPxU8

Nagranie filmowe: proces zamiany liczby dziesiętnej na binarną w pięciu krokach.

Przeczytaj

Podobnie jak w przypadku różnych języków na przestrzeni lat powstawały różne systemy liczbowe. Najbardziej znany i codziennie wykorzystywany jest dziesiętny system liczbowy, a więc taki, w którym do zapisywania liczb używamy 10 różnych cyfr: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Jest to jeden z wielu systemów pozycyjnych, czyli metod zapisywania liczb tak, by pozycja cyfry w ciągu oznaczała wielokrotność potęgi liczby będącej podstawą tego systemu.
W przypadku dziesiętnego systemu liczbowego (ang. *decimal numeral system*) podstawą jest liczba 10, w związku z tym zapis „153" otrzymujemy obliczając sumę wartości na kolejnych pozycjach:

$$1 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^0 = 100 + 50 + 3 = 153$$

System dziesiętny a dwójkowy

Systemem używanym powszechnie w informatyce jest dwójkowy system liczbowy, nazywany również **binarnym**. Podobnie jak system dziesiętny, jest on systemem pozycyjnym. Do zapisania każdej liczby w tym systemie możemy użyć tylko dwóch cyfr: 0 i 1, a podstawą jest liczba 2.

Jak w takim razie zinterpretować „153" w systemie binarnym, skoro do dyspozycji mamy tylko dwie cyfry 0 i 1?

Musimy dokonać konwersji tej liczby pomiędzy dwoma systemami. Taką operację możemy porównać do tłumaczenia słowa z jednego języka na inny.

> **Ważne!**
>
> Podczas pracy z różnymi systemami liczbowymi łatwo się pomylić, dlatego często obok liczby zapisuje się, w jakim systemie jest ona zapisana. Przykładowo 10 w systemie dziesiętnym oznacza zupełnie inną liczbę w systemie dwójkowym.
>
> $$10_{(10)} \neq 10_{(2)}$$

Funkcje wbudowane

Większość języków programowania posiada gotowe funkcje do przeprowadzania konwersji pomiędzy różnymi systemami liczbowymi. W języku Python możemy wykorzystać wbudowaną funkcję bin( ).

Wykorzystaj kod napisany poniżej i sprawdź, jakie wyniki otrzymasz dla różnych liczb.

```
n = 153          # Do zmiennej n przypisz liczbę 153
print(bin(n))    # Zapisz i wydrukuj liczbę w postaci binarnej
```

> **Ważne!**
>
> Python, tak jak wiele innych języków, używa prefiksów do oznaczania systemów liczbowych innych niż dziesiętne, dlatego wynik podany w systemie binarnym zawsze zaczyna się od znaków „0b”.

Wiemy już zatem, jak zaprezentować naszą liczbę w systemie binarnym:

$$153_{(10)} = 10011001_{(2)}$$

Czy potrzebna jest nam zatem wiedza, jak dokonywać konwersji ręcznie? Wiedza ta może przydać się między innymi podczas programowania mikrokontrolerów, ale również posłuży nam jako podstawa do nauki bardziej skomplikowanych systemów liczbowych używanych w informatyce.

Konwersja

„Tłumaczenie” liczby dziesiętnej zaczynamy od poszukiwania największej potrzebnej wielokrotności podstawy systemu, czyli 2.

Wypiszmy kolejne potęgi liczby 2:

$$2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

$$2^5 = 32$$

$$2^6 = 64$$

$$2^7 = 128$$

$$2^8 = 256$$

$$2^9 = 512$$

$$2^{10} = 1024$$

Widzimy zatem, że nasza liczba $153_{(10)}$ znajduje się w następującym przedziale:

$$2^7 \leq 153 < 2^8$$

To oznacza, że żeby zapisać liczbę $153_{(10)}$, będziemy potrzebować aż 8 cyfr w systemie dwójkowym.

> **Ciekawostka**
>
> Istnieje wzór matematyczny, dzięki któremu możemy policzyć, ile cyfr potrzebnych nam będzie do zapisania danej cyfry:
>
> $$\log_2 n$$
>
> Gdzie n jest liczbą, którą próbujemy zapisać.
>
> Zatem z tego wzoru możemy policzyć, że dla naszej liczby 153:
>
> $$\log_2 153 \approx 7{,}26$$
>
> Ponieważ nie możemy wykorzystać tylko fragmentu cyfry, sięgamy do pierwszej liczby całkowitej, która zmieści w sobie ten wynik i otrzymamy odpowiedź – potrzebujemy 8 cyfr.

Analogiczne do systemu dziesiętnego musimy sprawdzić, jaka wielokrotność potęgi dwójki pojawi się na danej pozycji. W przypadku systemu dwójkowego zadanie jest ułatwione, ponieważ każda pozycja może przyjąć tylko dwie wartości: 0 i 1. Zaczynając od najwyżej wartości widzimy zatem, że:

$$1 \cdot 2^7 = 128$$

Pierwszą cyfrą naszej liczby w systemie binarnym jest zatem 1. Odejmijmy tę wartość od początkowej liczby, żeby zobaczyć, jaka jest pozostała wartość, którą potrzebujemy przekonwertować.

$$153 - 128 = 25$$

Ponieważ w wartości 25 nie jesteśmy w stanie „zmieścić" kolejnej potęgi o wartości 64, ta pozycja przyjmie wartość 0. Ta sama sytuacja powtórzy się w przypadku następnej pozycji o wartości 32, więc ta pozycja również przyjmie wartość 0.

$$0 \cdot 2^6 = 0$$

$$0 \cdot 2^5 = 0$$

Kolejna potęga, czyli 2^4 ma wartość 16, 25 jest większe od 16 więc:

$$1 \cdot 2^4 = 16$$

Teraz odejmijmy tę wartość od 25 i otrzymamy:

$$25 - 16 = 9$$

W następnym kroku stwierdzamy, że 9 jest większe od kolejnej potęgi, czyli 2^3, zatem wykonajmy znane nam już czynności:

$$1 \cdot 2^3 = 8$$

$$9 - 8 = 1$$

Dokończmy obliczenia dla pozostałych cyfr:

$$0 \cdot 2^2 = 0$$

$$0 \cdot 2^1 = 0$$

$$1 \cdot 2^0 = 1$$

Widzimy, że po zapisaniu cyfry na ostatniej pozycji pozostała wartość wyniosła zero. Oznacza to, że udało nam się poprawnie dokonać konwersji pomiędzy systemem dziesiętnym i dwójkowym. Wykonane operacje możemy zatem zapisać jako następującą sumę:

$$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 153$$

$$128 + 0 + 0 + 16 + 8 + 0 + 0 + 1 = 153$$

$$10011001_{(2)} = 153_{(10)}$$

Optymalizacja

Skoro wykonywane obliczenia są powtarzalne, chcąc dokonać konwersji pisząc algorytm można wykorzystać pętlę. Możemy zauważyć, że wykonując obliczenia w przypadku liczby $153_{(10)}$ dzieliliśmy je poprzez kolejne potęgi liczby 2, aby sprawdzić, czy się w nich „zmieszczą". Przyjrzyjmy się zatem, jak wyglądałby początek konwersji, jeśli użyjemy tej metody:

$$153 : 2^7 \approx 1{,}195$$

$$25 : 2^6 \approx 0{,}391$$

$$25 : 2^5 \approx 0{,}781$$

Liczba w części całkowitej prawidłowo wskazuje wartości kolejnych pozycji w systemie binarnym. Wciąż mamy jednak część dziesiętną, która przechowuje wartościowe

informacje. Dlaczego? Spójrzmy na poniższe działania zapisane w postaci ułamków prostych:

$$\frac{153}{128} = \frac{128}{128} + \frac{25}{128}$$

$$\frac{25}{64} = \frac{0}{64} + \frac{25}{64}$$

$$\frac{25}{32} = \frac{0}{32} + \frac{25}{32}$$

Możemy zauważyć, że część ułamkowa, będąca resztą z dzielenia, zawiera w sobie informację potrzebną nam w kolejnym kroku działania, a część dziesiętna zawiera kolejne cyfry rozwinięcia binarnego.

Do znajdowania kolejnych cyfr liczby w systemie dwójkowym możemy wykorzystać działanie *modulo* 2, czyli obliczanie reszty z dzielenia przez podstawę systemu liczbowego.

Nasz algorytm możemy zapisać przy pomocy następującego pseudokodu:

```
while n > 0 do
    wynik = n mod 2 + wynik
    n = n / 2
```

Gdzie *n* jest konwertowaną liczbą dziesiętną, *mod* operacją wyznaczania reszty z dzielenia, a znak „/” oznacza dzielenie całkowite.

Słownik

podstawa systemu liczbowego

nazywana też bazą systemu liczbowego. Podstawa określa ilość cyfr dostępnych do zapisywania liczb w danym systemie oraz służy do wyznaczania cyfr na poszczególnych pozycjach.

mikrokontroler

układ cyfrowy z mikroprocesorym zawarty w jednym układzie scalonym. Tego typu układy są powszechnie używane między innymi w sprzęcie RTV i AGD, urządzeniach podłączanych do komputerów (takich jak urządzenia peryferyjne czy karty rozszerzeń) i układach pomiarowych

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Uzupełnij podany kod, aby otrzymać działającą implementację algorytmu konwersji liczby całkowitej zapisanej w systemie dziesiętnym do systemu binarnego..

Twoje zadania

1. Zdefiniowanie funkcji `konwertuj_liczbe_dziesietna_do_binarnej`.
2. Funkcja zwraca binarną reprezentację liczby 241.

```
def konwertuj_liczbe_dziesietna_do_binarnej(liczba):
    wynik = ""
    reszta = 0
    while liczba > 0:
        # Tu uzupełnij kod
    return wynik
```

```
```

Ćwiczenie 2

Uzupełnij podany kod, aby otrzymać funkcję konwertującą liczbę całkowitą zapisaną w systemie dziesiętnym do postaci binarnej.

Twoje zadania

1. Zdefiniowanie funkcji `konwertuj_ulamek`.
2. Funkcja zwraca postać binarną liczby 0.434 z precyzją 5 miejsc po przecinku.

```
def konwertuj_ulamek(liczba, precyzja):
    precyzja_start = 0
    wynik = "0,"
    while precyzja_start < precyzja:
        # Tu uzupełnij kod

        precyzja_start += 1
    return wynik

liczba = 0.434
precyzja = 5
```

```
1
```

Ćwiczenie 3

Napisz program, który przekonwertuje podaną liczbę większą od jeden z systemu dziesiętnego na binarny. Użyj w programie obu poznanych funkcji: do konwersji liczby całkowitej i do konwersji części ułamkowej. Opracuj sposób rozbicia liczby na część całkowitą i część ułamkową; pomocna może być funkcja podłogi `floor`.

Twoje zadania

1. Zdefiniowanie funkcji `konwertuj_liczbe`.
2. Funkcja zwraca postać binarną liczby 21.6875 z dokładnością do 6 miejsc po przecinku.

```
import math

def konwertuj_liczbe(liczba, precyzja):
    # Tu uzupełnij kod
    pass


def konwertuj_czesc_calkowita(liczba):
    # Tu uzupełnij kod
    pass

def konwertuj_ulamek(liczba, precyzja):
    # Tu uzupełnij kod
    pass
```

```

```

Dla nauczyciela

Autor: Zespół Gromar.eu

Przedmiot: informatyka

Temat: Konwersja liczb z systemu dziesiętnego na dwójkowy w języku Python.

Grupa docelowa: III etap edukacyjny, liceum, technikum

Podstawa programowa:

Zakres podstawowy

I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

a) na liczbach: badania pierwszości liczby, zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi, działań na ułamkach z wykorzystaniem NWD i NWW.

Kompetencje kluczowe:

- kompetencje w zakresie rozumienia i tworzenia informacji,
- kompetencje w zakresie wielojęzyczności,
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii,
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się,
- kompetencje cyfrowe.

Cele operacyjne:

Uczeń:

- poznaje zależności pomiędzy dziesiętnym a dwójkowym systemem liczbowym;
- wykorzystuje wbudowane funkcje języka Python do zamiany liczby dziesiętnej na odpowiednik binarny;
- konwertuje liczby z systemu dziesiętnego na dwójkowy.

Strategie nauczania:

- konstruktywizm.

Metody i techniki nauczania:

- rozmowa kierowana;
- burza mózgów;
- analiza kodu;
- programowanie.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami i dostępem do internetu, słuchawki;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg zajęć:

Faza wstępna

1. Na poprzednich zajęciach nauczyciel prosi grupę uczniów o przygotowanie prezentacji dotyczącej różnic między systemem dziesietnym i dwójkowym.
2. Przedstawienie tematu i celów zajęć oraz wspólne z uczniami ustalenie kryteriów sukcesu.
3. Prezentacja przygotowana przez uczniów. Reszta klasy analizuje materiał kolegów, wskazuje jego mocne i słabe strony.

Faza realizacyjna

1. Wspólne omówienie funkcji wbudowanej `bin`. W ramach sprawdzenia, czy materiał został zrozumiany, uczniowie porównują wynik jej działania z obliczeniami ręcznymi.
2. Praca w parach – analiza konwersji liczby dziesietnej na system binarny. Wybrane pary przedstawiają wnioski na forum klasy. Wspólne rozwiązywanie przykładów.
3. Burza mózgów: W jaki sposób dokonywać konwersji szybciej? Zapisanie pomysłów na tablicy. Po fazie twórczej następuje weryfikacja pomysłów i sformułowanie wniosków dotyczacych optymalizacji.
4. Praca z multimedium bazowym – film samouczek. Uczniowie analizują konwersję liczb dziesiętnych po optymalizacji i weryfikują swoje wnioski z burzy mózgów.
5. Uczniowie piszą funkcję `toBinary`, która jako parametr przyjmie dodatnią liczbę całkowitą w systemie dziesiętnym i zamieni ją na jej odpowiednik w systemie dwójkowym.

Faza podsumowująca

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.
2. Nauczyciel omawia przebieg zajęć, wskazuje mocne i słabe strony pracy uczniów, udzielając im tym samym informacji zwrotnej.

Materiały pomocnicze:

Dokumentacja dla Python 3

Wskazówki metodyczne opisujące różne zastosowania multimedium:

Film samouczek uczniowie mogą wykorzystać jako inspirację do stworzenia własnego samouczka lub prezentacji dotyczących implementacji algorytmu w języku Python.