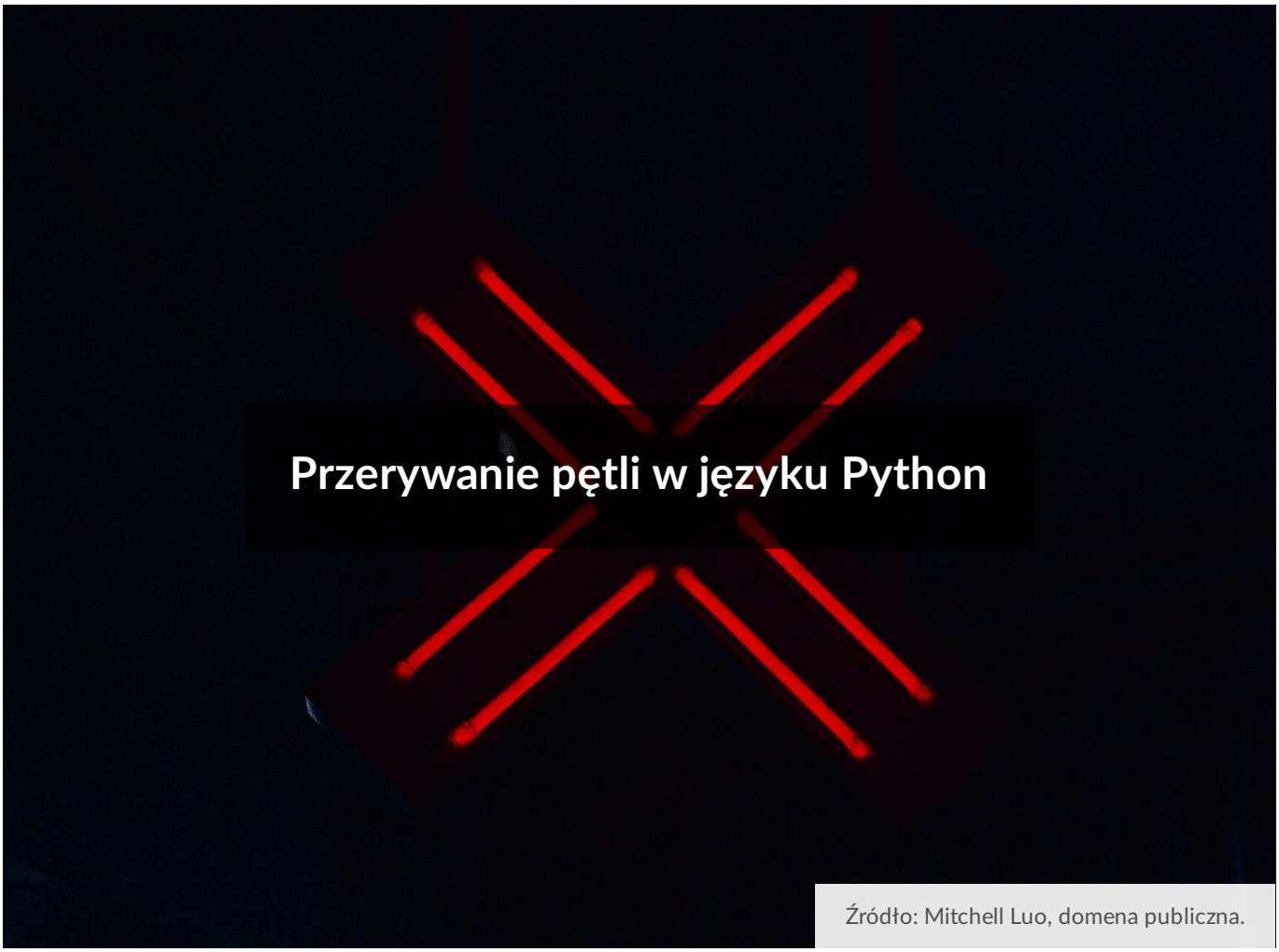
Przerywanie pętli w języku Python



Źródło: Mitchell Luo, domena publiczna.

W e-materiale Przerywanie pętli poznaliśmy optymalizujące algorytm instrukcje `break` oraz `continue` . Są to dodatkowe instrukcje sterujące zachowaniem pętli, które pozwalają wymusić zakończenie iteracji i ominąć pewne instrukcje lub całkowicie zatrzymać działanie pętli.

W tym e-materiale zaimplementujemy omawiane instrukcje w języku Python.

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

- Przerywanie pętli w języku C++,
- Przerywanie pętli w języku Java.

Więcej zadań? Sięgnij do Przerywanie pętli – zadania maturalne.

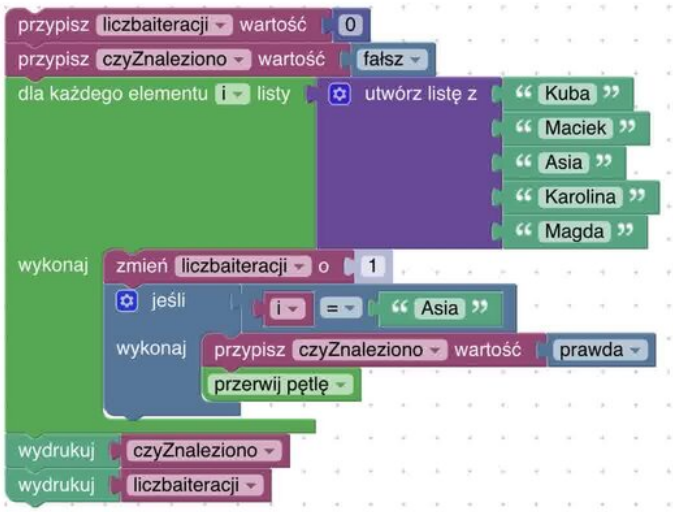
> **Twoje cele**
>
> - Poznasz instrukcje `break` i `continue`.
> - Poznasz przykłady, kiedy warto zastosować te instrukcje.
> - Wykorzystasz `break` i `continue` do optymalizacji algorytmów.

Animacja

Polecenie 1

Przeanalizuj poniższą animację i przygotuj dla niej narrację, w której wyjaśnisz metodę tworzenia instrukcji `break` i `continue`. Zapisz wynik swojej pracy w poniższym formularzu.

Wystąpił błąd



Film dostępny pod adresem /preview/resource/RoZgxytVc4nJ4

Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Animacja przedstawiająca "Instrukcje break i continue w Pythonie".

Ćwiczenie 1

Problem 1
Specyfikacja problemu:

Wypisz i zsumuj kolejne liczby nieparzyste od 1 do 100. Operację przerwij, gdy suma przekroczy liczbę 50.

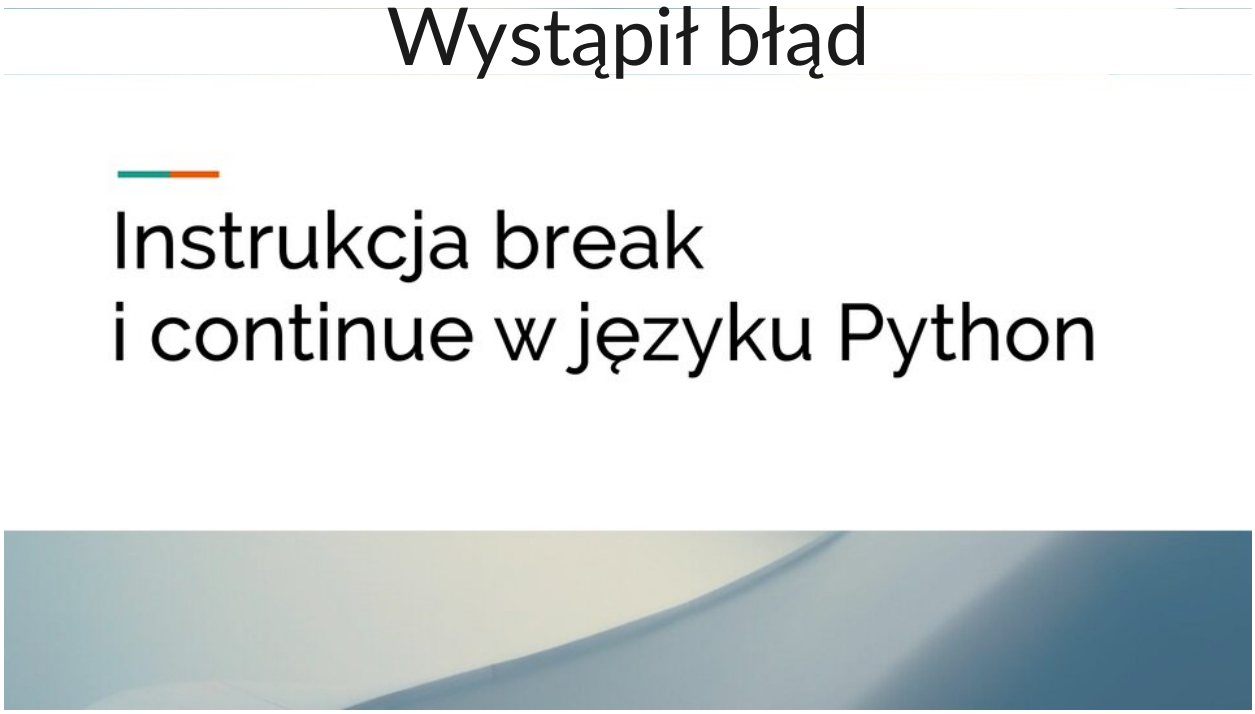
Dane:

Liczba jest parzysta, jeżeli dzieli się przez 2 bez reszty.

Wynik:

Na konsoli wyświetlą się liczby nieparzyste, a pętla zostanie przerwana gdy suma będzie większa niż 50.

Polecenie 2
Porównaj swoje rozwiązanie z filmem poniżej.



Film dostępny pod adresem /preview/resource/R1MY9j9NF1M3u

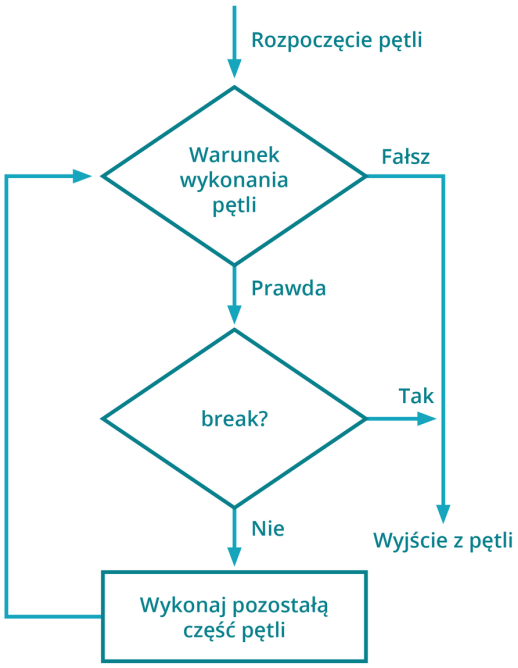
Film nawiązujący do treści materiału

Przeczytaj

Instrukcja `break`

Instrukcja `break` kończy pętlę, w której się znajduje. Program wychodzi z pętli i jest wykonywany dalej. Instrukcję `break` można stosować zarówno w pętli `for`, jak i `while`.

Jeśli `break` znajduje się w zagnieżdżonej pętli, kończy tylko jedną pętlę – tę, w której się znajduje.



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Spójrzmy na przykładowe wykorzystanie `break` w kodzie. Poniżej znajduje się program sprawdzający, czy „Asia” znajduje się na liście imion.

```
imiona = ["Kuba", "Asia", "Karolina", "Maciek", "Magda"]
czyZnaleziono = False

for imie in imiona:
    print("Imie: " + imie)

    if imie == "Asia":
        print("Znaleziono!")
        czyZnaleziono = True

```

```
print("Czy znaleziono? " + str(czyZnaleziono))
```

Wynik będzie następujący:

```
Imie: Kuba
Imie: Asia
Znaleziono!
Imie: Karolina
Imie: Maciek
Imie: Magda
Czy znaleziono? True
```

Jak widać, mimo że imię zostało znalezione już na początku pętli, była ona wykonywana do końca. Wykorzystując instrukcję `break`, możemy przerwać pętlę, kiedy imię będzie znalezione. Poniższy kod przedstawia program z dodaną instrukcją `break`:

```
imiona = ["Kuba", "Asia", "Karolina", "Maciek", "Magda"]
czyZnaleziono = False

for imie in imiona:
    print("Imie: " + imie)

    if imie == "Asia":
        print("Znaleziono!")
        czyZnaleziono = True
        break

print("Czy znaleziono? " + str(czyZnaleziono))
```

Jej wynik wygląda następująco:

```
Imie: Kuba
Imie: Asia
Znaleziono!
Czy znaleziono? True
```

Dzięki tej operacji nasz program został zoptymalizowany i nie wykonuje niepotrzebnie kodu. Tak samo możemy zrobić w pętli `while`.

Instrukcja `continue`

Instrukcja `continue` działa podobnie, ale nie kończy działania całej pętli, a jedynie kończy aktualną iteracją – kod wewnątrz pętli, który jest po `continue`, nie zostanie wykonany w tej iteracji.

Można to wykorzystać w sytuacji, kiedy element wewnątrz pętli nie spełnia założonych warunków i nie ma potrzeby wykonywania kolejnych instrukcji w tej iteracji pętli.

Poniższy program drukuje silnię tylko parzystych liczb z podanego zakresu.

```
for x in range(1, 10):
    if x % 2 != 0:
        continue # liczba nie jest parzysta - nie ma sensu wykony
    silnia = x
    for y in range(1, x):
        silnia *= y

    print("Silnia liczby {0} wynosi {1}".format(x, silnia))
```

Jak widać, jeśli liczba jest nieparzysta, to dalsza część pętli nie jest wykonywana, tylko natychmiast wykonywana jest kolejna iteracja.

Czy `break` i `continue` są konieczne?

Powyższe programy można również napisać, nie korzystając z instrukcji `break` i `continue`. Piszesz po prostu odpowiednie instrukcje warunkowe.

Instrukcje `break` i `continue`, jeśli są wykorzystywane w skomplikowanych algorytmach, w wielu miejscach, sprawiają, że kod staje się nieczytelny. Łamie to zasady pisania dobrego kodu i sprawia problem innym w zrozumieniu programu.

Z tego względu powinno się korzystać z instrukcji przerwania z umiarem – tylko wtedy, kiedy jest to konieczne.

Słownik

> **zagnieżdżenie pętli**
>
> wywołanie jednej pętli wewnątrz drugiej
> **optymalizacja**

> poprawa wydajności programu komputerowego lub algorytmu poprzez zmniejszenie liczby operacji potrzebnych do otrzymania wyniku

Sprawdź się

Pokaż ćwiczenia:

- Ćwiczenie 1
- Ćwiczenie 2
- Ćwiczenie 3
- Ćwiczenie 4
- Ćwiczenie 5
- Ćwiczenie 6
- Ćwiczenie 7
- Ćwiczenie 8
- Ćwiczenie 9

Dla nauczyciela

Autor: Jakub Kamiński
Przedmiot: Informatyka

Temat: Przerywanie pętli w języku Python

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Poznasz instrukcje `break` i `continue`.
- Poznasz przykłady, kiedy warto zastosować te instrukcje.
- Wykorzystasz `break` i `continue` do optymalizacji algorytmów.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

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

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Przerywanie pętli w języku Python". Nauczyciel prosi uczniów o zapoznanie się z multimedium w sekcji „Przeczytaj".

Faza wstępna:

1. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. Uczniowie analizują animację i przygotowują dla niej narrację, w której wyjaśniają metodę tworzenia instrukcji `break` i `continue`
2. **Praca z tekstem.**Nauczyciel wyświetla zawartość sekcji „Przeczytaj". Na forum klasy uczniowie analizują przedstawione w niej treści.
3. **Ćwiczenie umiejętności.** Prowadzący zapowiada uczniom, że w kolejnym kroku będą rozwiązywać ćwiczenia nr 1-9 z sekcji „Sprawdź się". Każdy z uczniów robi to samodzielnie. Po ustalonym czasie wybrani uczniowie przedstawiają rozwiązania.

Nauczyciel w razie potrzeby koryguje odpowiedzi, dopowiada istotne informacje, udziela uczniom informacji zwrotnej.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".
2. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 1 z sekcji „Animacja". Piszą własny scenariusz narracji dla animacji: „Instrukcje break i continue w Pythonie".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Nauczyciel może wykorzystać multimedium w sekcji „Przeczytaj" do pracy przed lekcją. Uczniowie zapoznają się z jego treścią i przygotowują do pracy na zajęciach w ten sposób, żeby móc samodzielnie rozwiązać zadania dołączone do e-materiału „Przerywanie pętli w języku Python".