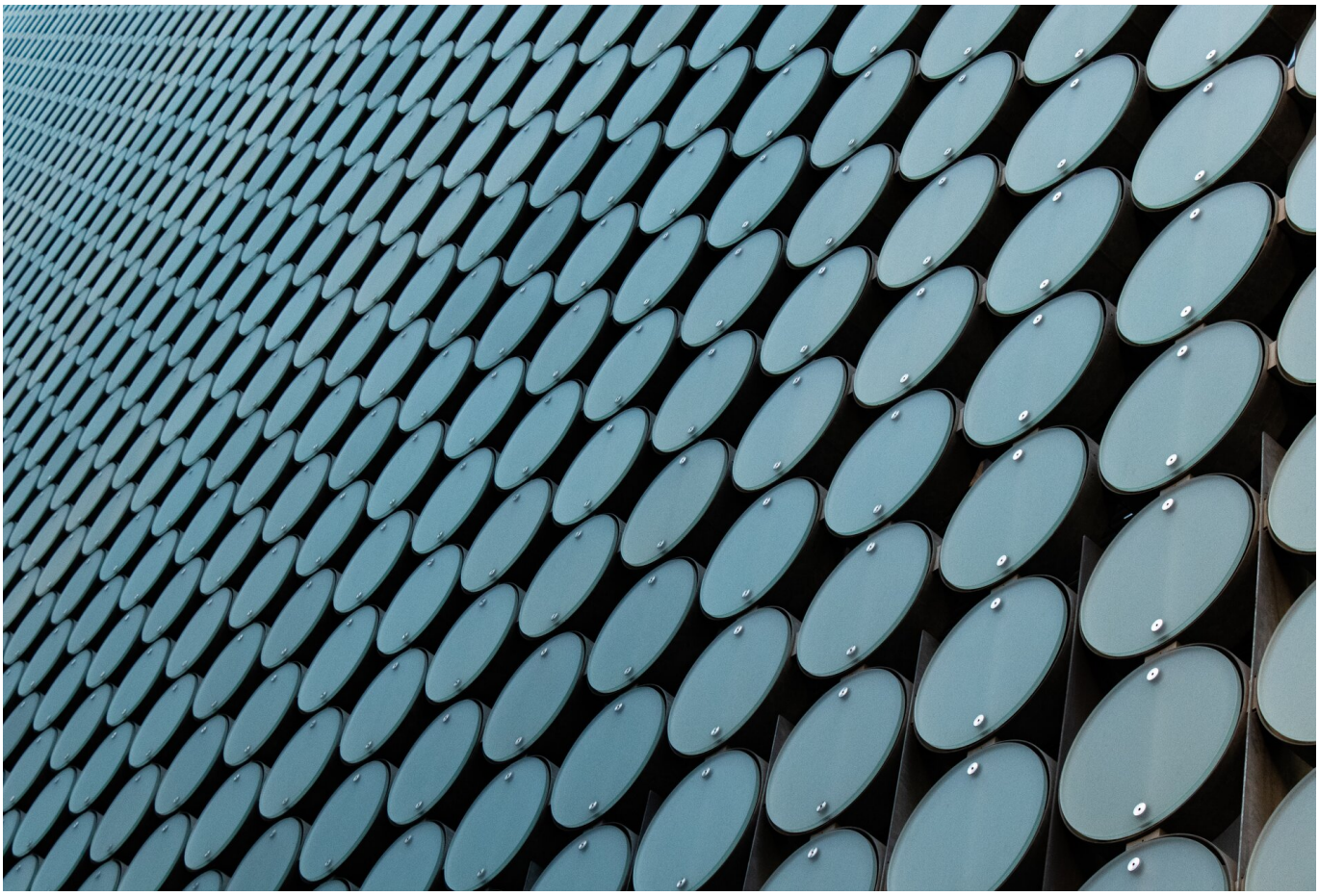


Analiza podejścia rekurencyjnego i iteracyjnego w języku Python

Analiza podejścia rekurencyjnego i iteracyjnego w języku Python

Źródło: Mitchell Luo, domena publiczna.

Wiesz już, czym charakteryzuje się podejście rekurencyjne, a czym iteracyjne. Czy potrafisz jednak na podstawie analizowanego problemu wskazać, które podejście wybrać? W tym e-materiale omówimy różnice w implementacji programu wykorzystującego każde z tych rozwiązań w języku Python.

O tym, jak zagadnienie rekurencji wyjaśnia matematyka, przeczytasz w e-materiałach:

- Ciąg określony rekurencyjnie,
- Ciąg geometryczny określony rekurencyjnie,
- Wzór ogólny ciągu określonego rekurencyjnie,
- Ciąg arytmetyczny określony wzorem rekurencyjnym.

Analizę algorytmów iteracyjnych i rekurencyjnych w pozostałych językach programowania znajdziesz w e-materiałach:

- Analiza podejścia rekurencyjnego i iteracyjnego w języku C++,
- Analiza podejścia rekurencyjnego i iteracyjnego w języku Java.

Więcej zadań? Przejdź do e-materiału Analiza podejścia rekurencyjnego i iteracyjnego – zadania maturalne.

> **Twoje cele**

- Zaimplementujesz funkcję obliczającą wartość potęgi x^n metodą iteracyjną.
- Zaimplementujesz funkcję obliczającą wartość potęgi x^n metodą rekurencyjną.
- Przeanalizujesz i porównasz oba rozwiązania.
- Utrwalisz umiejętność posługiwania się biblioteką `matplotlib`.

Film samouczek

Problem 1

Napisz program obliczający potęgę danej liczby naturalnej wskazanymi metodami – rekurencyjną i iteracyjną.

Specyfikacja problemu:

Dane:

- `baza` – liczba naturalna dodatnia; podstawa potęgi
- n – liczba naturalna; wykładnik potęgi

Wynik:

- `potega` – liczba całkowita; wynik potęgowania

Metoda rekurencyjna. Przetestuj działanie swojego programu dla 3^8.

```
# Tutaj dodaj własny kod. Użyj funkcji
# print()
```

```
```

Metoda iteracyjna. Przetestuj działanie swojego programu dla 3^4.

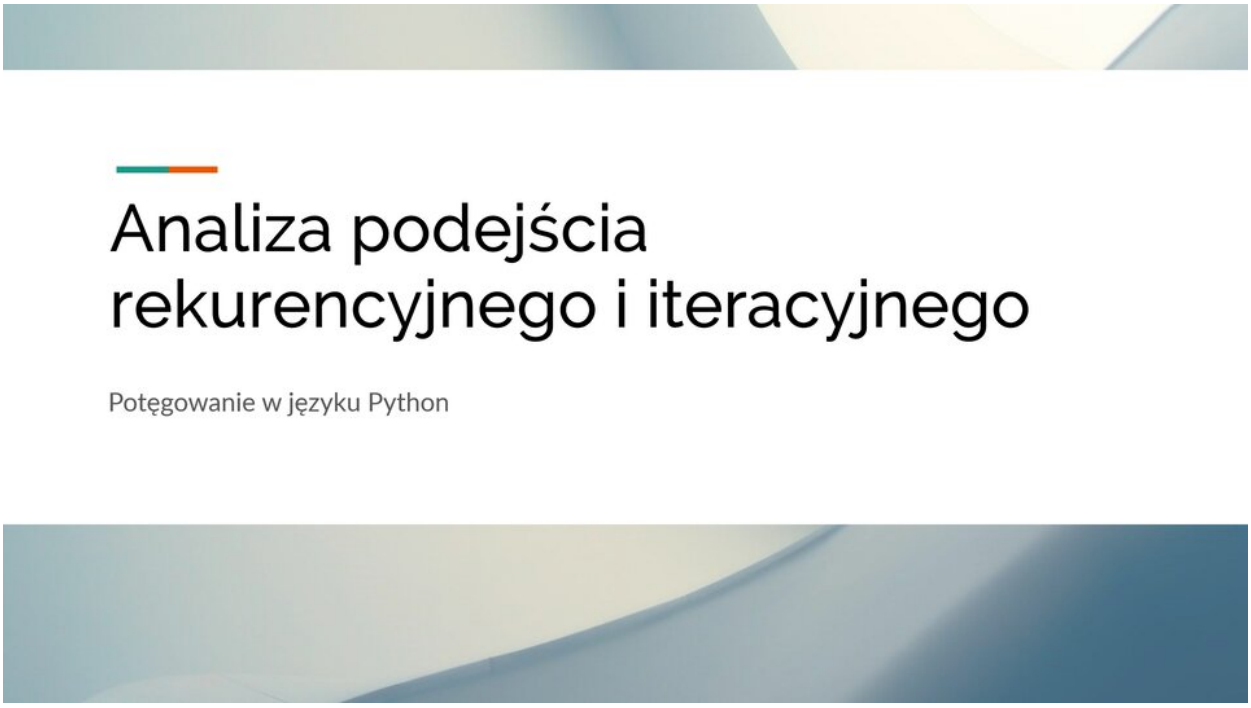
```
# Tutaj dodaj własny kod. Użyj funkcji
# print()
```

```
```

Polecenie 1

Porównaj swoje rozwiązania z przedstawionym w filmie.

W filmie mówimy o tym, że każda liczba podniesiona do potęgi 0 daje 1. Jest to prawdą w stosunku do każdej liczby różnej od 0. W przypadku 0^0 nie jest to tak jednoznaczne, jednak nie będziemy zajmować się tą kwestią w tym e-materiale.



Film dostępny pod adresem /preview/resource/Rg8dz0Xo40dUM

Film dotyczy analizy podejścia rekurencyjnego i iteracyjnego.

Plik o rozmiarze 389.00 B w języku polskim

Polecenie 2

Dodaj do swojego programu komentarze tak, żeby był zrozumiały dla osoby, która nie potrafi programować.

Przeczytaj

Obliczanie wartości potęgi o podstawie i wykładniku naturalnym

Specyfikacja problemu:

Dane:

- p – liczba naturalna; podstawa potęgi; $p > 0$
- w – liczba naturalna; wykładnik potęgi; $w > 0$

Wynik:

- program zwraca wynik potęgowania p^w

Sposób iteracyjny

```
def potega_iter(p, w):
    wynik = 1
    while w > 0:
        wynik = wynik * p
        w -= 1

    return wynik
```

Sposób rekurencyjny

```
def potega_rekur(p, w):
    if w == 1:
        return p

    return p * potega_rekur(p, w - 1)
```

Obliczanie silni

Specyfikacja problemu:

Dane:

- `liczba` – liczba naturalna

Wynik:

- program zwraca wartość silni podanej liczby naturalnej

Sposób iteracyjny

```
def silnia(liczba):
    if liczba < 2:
        return 1
    else:
        for i in range(2,liczba):
            liczba *= i
        return liczba
```

Sposób rekurencyjny

```
def silnia (liczba):
    if liczba < 1:
        return 1
    else:
        return liczba * silnia(liczba - 1)
```

Analiza i porównanie

Możemy zauważyć, że liczba operacji mnożenia, jaką wykonują obie wersje funkcji, jest identyczna. Natomiast różny jest czas niezbędny do ich wykonania. W przypadku funkcji rekurencyjnej obserwujemy znacznie większy czas niezbędny do wykonania obliczenia. Wynika to z faktu, że wykonywanie funkcji rekurencyjnych obarczone jest zawsze narzutem czasowym związanym z obsługą stosu, a więc dostępem do pamięci, w której zapamiętywane są adresy powrotu z kolejnych wywołań rekurencyjnych. W przypadku wersji iteracyjnej czas niezbędny na wykonywanie mnożeń jest znacznie krótszy. W obu przypadkach obserwujemy liniowy wzrost czasu wraz ze wzrostem liczby mnożeń, co jest normalną sytuacją.

Możemy również zauważyć, że w wersji rekurencyjnej funkcja ma krótszy kod (choć w tym przypadku nie jest on znacznie krótszy).

> **Już wiesz**
>
> - Czas konieczny na wykonanie funkcji może zależeć od jej rodzaju.

Słownik

iteracja

powtarzanie tej samej operacji określoną liczbę razy lub do momentu, w którym zadany warunek zostanie spełniony

matplotlib

biblioteka służąca do przedstawienia obrazów złożonych z punktów o współrzędnych `x` oraz `y` (wykresów, histogramów, rozkładów itp.); moduł `matplotlib` nie jest dostępny w standardowej instalacji języka Python; należy go zainstalować, korzystając z mechanizmu `pip`

rekurencja

proces polegający na wywoływaniu funkcji przez siebie samą (zazwyczaj z innymi parametrami) do momentu rozwiązania określonego problemu

stos

liniowa struktura danych, której dane są pobierane zgodnie z zasadą LIFO (dane dołączone jako ostatnie, pobierane są jako pierwsze)

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Zdefiniuj funkcję.

> Przykładowe zadanie pochodzi ze *Zbioru zadań maturalnych z informatyki – materiały pomocnicze dla uczniów i nauczycieli*, opublikowanego przez CKE w 2015 r.

Niech dana będzie liczba naturalna x, której zapis dziesiętny ma n cyfr:

$$x = a_{n-1}10^{n-1} + a_{(n-2)}10^{n-2} + \ldots + a_1 10 + a_0$$

$$(a_{n-1} \neq 0)$$

Powiemy, że liczba x jest narcystyczna, jeśli suma jej cyfr podniesionych do potęgi n-tej jest równa x, tzn.

$$a_{n-1}^n + a_{n-2}^n + \ldots + a_1^n + a_0^n = x$$

Na przykład liczba 1634 jest narcystyczna, ponieważ:

$$1^4 + 6^4 + 3^4 + 4^4 = 1634$$

Zdefiniujmy funkcję `czy_narcystyczna(liczba)`, która sprawdzi, czy dana liczba jest narcystyczna. Wykorzystajmy funkcję obliczającą potęgę opisaną w sekcji „Przeczytaj".

Specyfikacja problemu:

Dane:

- `liczba` – liczba naturalna; liczba do sprawdzenia (należy sprawdzić, czy jest liczbą narcystyczną)

Wynik:

- wartość logiczna `prawda`, jeśli liczba jest narcystyczna
- wartość logiczna `fałsz`, jeśli liczba nie jest narcystyczna

Swój program przetestuj dla liczby 1634.

Twoje zadania

1. Program sprawdza, czy `liczba` jest liczbą narcystyczną.

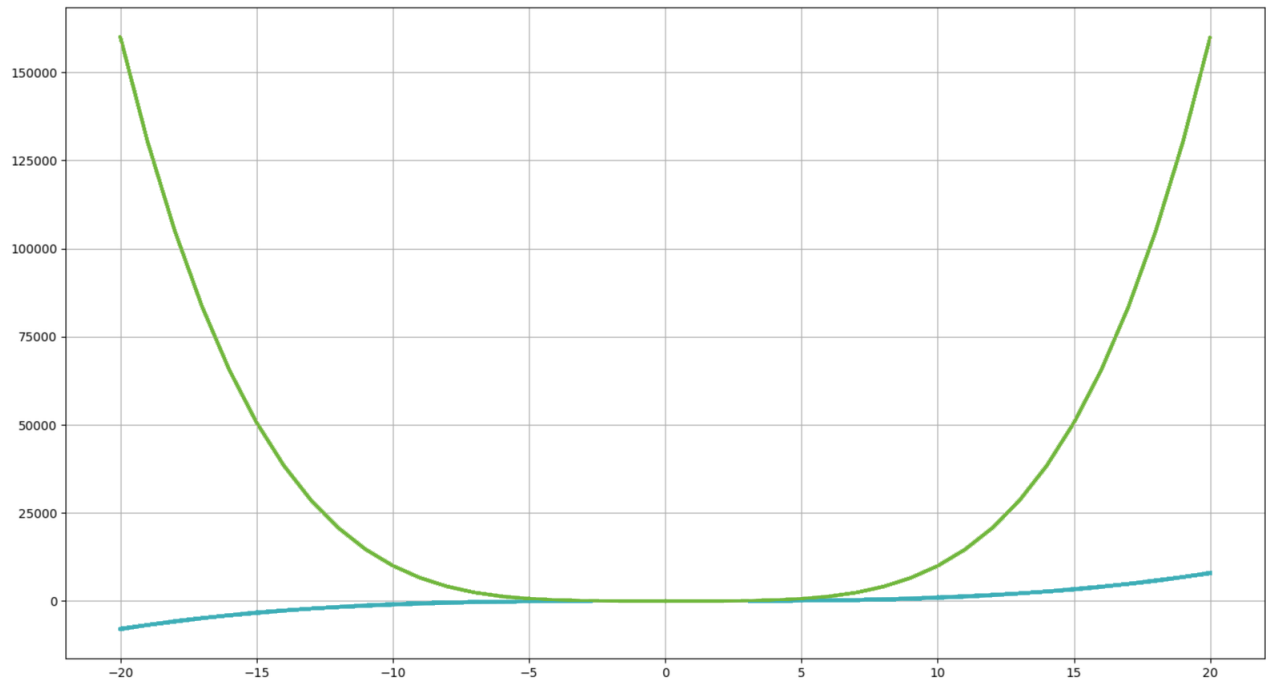
```
def czy_narcystyczna(liczba):
    # tutaj Twój kod
    return None

narcystyczna = czy_narcystyczna(1634)
print(narcystyczna)
```

```
1
```

Ćwiczenie 2

W środowisku Python wykonaj kod zamieszczony pod wykresem, dobierając kolejne wersje generowania wartości `X`, `Y1`, `Y2`.

Porównaj wyniki z poniższym wykresem, a następnie wykonaj polecenie. To ćwiczenie ma za zadanie utrwalić umiejętności posługiwania się biblioteką `matplotlib` oraz analizy informacji i czytania ze zrozumieniem.



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

Aby zainstalować opisywaną bibliotekę, musimy wydać w systemowym terminalu polecenie:

```
pip install matplotlib
```

Wskaż poprawny zapis.

```
import matplotlib.pyplot as plt

# tutaj wstawiamy kod tworzenia X oraz Y1 i Y2

plt.plot(X,Y1)
plt.plot(X,Y2)
plt.grid(True)
plt.show()
```

```
X = [ x for x in range(-20, 20) ]
Y1 = [ (x**3) - 3*x for x in X ]
Y2 = [ (x**4) - 3*x for x in X ]
```

```
X = [ x for x in range(-20, 20) ]
Y1 = [ (x**3) - 3*x for x in X ]
Y2 = [ (x**4) + 3*x for x in X ]
```

```
X = [ x for x in range(-20, 21) ]
Y1 = [ (x**3) - 3*x for x in X ]
Y2 = [ (x**4) - 3*x for x in X ]
```

Ćwiczenie 3

Zdefiniuj dwie funkcje:

- `iteracyjna_potega(podstawa, wykladnik)`
- `rekurencyjna_potega(podstawa, wykladnik)`

Każda z funkcji powinna zwracać krotkę (`tuple`) zawierającą dwie liczby:

- wartość potęgi o określonej podstawie i wykładniku,
- liczbę wykonanych operacji.

Pamiętaj o odpowiednim zdefiniowaniu zmiennej globalnej dla zliczania liczby wywołań funkcji rekurencyjnej. Niech ta zmienna nazywa się `ile_rekurencji`.

Specyfikacja problemu:

Dane:

- `podstawa` – liczba naturalna; liczba, którą należy podnieść do potęgi
- `wykladnik` – liczba naturalna; stopień potęgi

Wynik:

- `wynik` – krotka zawierająca dwie liczby naturalne; wynik potęgowania i liczba operacji

Swój program przetestuj dla obu funkcji z parametrami: `podstawa`, `wykladnik` równymi odpowiednio 2, 5.

Twoje zadania

1. Program oblicza potęgę liczby `podstawa` o stopniu `wykladnik`, a także liczbę operacji.

```python
# pamiętamy o zmiennej do zliczania wywołań rekurencyjnych
ile_rekurencji = 0

def iteracyjna_potega(podstawa, wykladnik):
    pass


def rekurencyjna_potega(podstawa, wykladnik):
    pass


# tu przykładowe wywołania
wynik = iteracyjna_potega(2, 5)
print(wynik)
```

```
1
```

Dla nauczyciela

Autor: Adam Jurkiewicz
Przedmiot: Informatyka

Temat: Analiza podejścia rekurencyjnego i iteracyjnego w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

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

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

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

e) obliczania wartości elementów ciągu metodą iteracyjną i rekurencyjną, w tym wartości elementów ciągu Fibonacciego.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

8) dyskutuje na temat roli myślenia komputacyjnego i jego metod, takich jak: abstrakcja, reprezentacja danych, dekompozycja problemu, redukcja, myślenie rekurencyjne, podejście heurystyczne w rozwiązywaniu problemów z różnych dziedzin.

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),

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):

- Zaimplementujesz funkcję obliczającą wartość potęgi x^n metodą iteracyjną.
- Zaimplementujesz funkcję obliczającą wartość potęgi x^n metodą rekurencyjną.
- Przeanalizujesz i porównasz oba rozwiązania.
- Utrwalisz umiejętność posługiwania się biblioteką `matplotlib`.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Analiza podejścia rekurencyjnego i iteracyjnego w języku Python". Uczniowie zapoznają się z multimedium w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat zajęć oraz cele. Prosi, by na ich podstawie uczniowie sformułowali kryteria sukcesu.

Faza realizacyjna:

1. **Praca z multimedium.** Nauczyciel przechodzi do sekcji „Film samouczek". Uczniowie w parach rozwiązują Problem 1. Następnie porównują swoje rozwiązanie z przedstawionym w filmie. Nauczyciel wyjaśnia ewentualne wątpliwości.
2. **Praca z tekstem.** Nauczyciel wyświetla zawartość sekcji „Film samouczek". Uczniowie powtarzają i testują na swoich komputerach przedstawione w tej sekcji iteracyjne i rekurencyjne wersje rozwiązań problemów.
3. **Ćwiczenie umiejętności.** Uczniowie implementują poznane techniki do rozwiązywania problemów informatycznych, poprzez wykonywanie indywidualnie ćwiczeń z sekcji „Sprawdź się"

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. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie odpowiadają na pytanie: Kiedy wybrać metodę rekurencyjną, a kiedy iteracyjną?

Materiały pomocnicze:

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

Wskazówki metodyczne:

- Nauczyciel może wykorzystać multimedium w sekcji „Film samouczek" 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 „Analiza podejścia rekurencyjnego i iteracyjnego w języku Python”.