Konstruktory i destruktory w języku Python

Źródło: Matthew Hamilton, domena publiczna.

W trakcie pracy z aplikacjami zorientowanymi obiektowo powinniśmy zdawać sobie sprawę z roli specyficznych funkcji: konstruktora i destruktora obiektów w języku Python. W tym e-materiale przyjrzymy się im bliżej.

Więcej informacji o programowaniu obiektowym znajdziesz w e-materiale: Programowanie obiektowe – projekt, etap I. Natomiast z teorią na temat konstruktorów i destruktorów możesz zapoznać się w: Konstruktory i destruktory.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Zostały one omówione w dwóch pozostałych materiałach z tej serii:

- Konstruktory i destruktory w języku C++,
- Konstruktory i destruktory w języku Java.

Twoje cele

- Wyjaśnisz, jakie funkcje spełniają konstruktory i destruktory klas w języku Python.
- Prześledzisz, jak stworzyć konstruktor oraz destruktor klasy w języku Python.
- Przeanalizujesz działanie programów z zaimplementowanymi konstruktorami oraz destruktorami.
- Zaimplementujesz w języku Python programy wykorzystujące konstruktor oraz destruktor.

Przeczytaj

Wiemy, jak zdefiniować klasę w języku Python (informacje na ten temat znajdziesz między innymi w e-materiale Wstęp do programowania obiektowego w języku Python). Potrafimy też tworzyć obiekt odwołujący się do niej. Zbadamy teraz, w jaki sposób taki obiekt powstaje. Z tego e-materiału dowiemy się również, jakie czynności są wykonywane podczas usuwania go z przestrzeni nazw.

Stwórzmy klasę, która będzie przechowywać dane pracowników firmy. Każdy obiekt powinien mieć następujące pola:

- imię,
- nazwisko,
- płeć (kobieta/mężczyzna),
- numer telefonu.

> **Przykład 1**
>
> Zdefiniujmy klasę `Pracownik`. Konstruktor tworzymy przy użyciu specjalnej funkcji dunder o nazwie `__init__()`. Dzięki niej utworzymy i zainicjalizujemy pola instancji klasy.

```
class Pracownik:
    def __init__(self, imie: str, nazwisko: str, plec: str, tel
        self.imie = imie
        self.nazwisko = nazwisko
        self.plec = plec
        self.telefon = telefon

    def pokaz_informacje(self):
        print("Pracownik:",self.imie,self.nazwisko,",płeć:",sel
```

> **Ważne!**
>
> `__init__()` pełni rolę konstruktora – specjalnej funkcji dunder, która jest wykonywana podczas tworzenia obiektu. Może ona służyć do inicjowania pól powiązanych z danym obiektem oraz wykonywać inne dowolne operacje związane z nim lub z całą klasą.
> **Ważne!**
>
> Słowo kluczowe `self` – wskazuje na konkretny obiekt, który tworzymy.

Przykład 2

Utwórzmy obiekt klasy `Pracownik`, a następnie wywołajmy metodę `pokaz_informacje()`.

```
michal = Pracownik("Michał", "Adamczyk", "M", 421236790)
michal.pokaz_informacje()
# Pracownik: Michał Adamczyk ,płeć: M ,numer telefonu: 42123679
```

Przykład 3

Do konstruktora dodamy nowe funkcjonalności. Możemy na przykład skorzystać z domyślnych argumentów. Gdy użytkownik nie wprowadzi płci do konstruktora, możemy domyślnie ustawić ją na `"M"`, a w przypadku telefonu domyślą wartością może być `None`. Poza tym możemy sprawdzać, czy użytkownik podał poprawne dane. Przykładowo możemy nie pozwalać na ustawienie płci innej niż `"M"` i `"K"`.

```
class Pracownik:
    def __init__(self, imie: str, nazwisko: str, plec: str = "M
        self.imie = imie
        self.nazwisko = nazwisko
        # sprawdzamy poprawność płci
        if plec != "K" and plec != "M":
            print("Niepoprawny format płci!")
            self.plec = "M"
        else:
            self.plec = plec
        # sprawdzamy poprawność numeru telefonu
        if (len(str(telefon)) != 9 or type(telefon) != int) and
            print("Niepoprawny format telefonu!")
            self.telefon = None
        else:
            self.telefon = telefon

    def pokaz_informacje(self):
        print("Pracownik:", self.imie, self.nazwisko, ",płeć:",
```

Przetestujmy, jak działa nowy konstruktor:

```
adam = Pracownik("Adam", "Woźniak")
```

```
adam.pokaz_informacje()
# Pracownik: Adam Woźniak ,płeć: M ,numer telefonu: None

agata = Pracownik("Agata", "Lewandowska", "kobieta", 123)
# Niepoprawny format płci!
# Niepoprawny format telefonu!
agata.pokaz_informacje()
# Pracownik: Agata Lewandowska ,płeć: M ,numer telefonu: None
```

Ważne!

Jeśli potrzebujemy informacji o skasowaniu obiektu (np. chcemy utworzyć dziennik z odpowiednimi informacjami), mamy do dyspozycji specjalną metodę dunder o nazwie `__del__()`. Jest to tzw. `destruktor`. Zostaje on wykonany w momencie, w którym obiekt danej klasy przestaje istnieć (jest usuwany z pamięci).

Przykład 4

Rozszerzymy klasę `Pracownik` o nowe funkcjonalności. Zdefiniujmy w niej słownik przechowujący liczbę osób aktualnie pracujących w firmie. Wywołanie konstruktora tej klasy oznacza wzrost liczby pracowników (np. zatrudnienie nowego pracownika), co powoduje konieczność wzrostu liczby pracowników przechowywanej w stworzonym wcześniej słowniku. Uzyskamy to poprzez inkrementację tej wartości w konstruktorze klasy pod koniec inicjalizacji obiektu (w kodzie przedstawionym poniżej jest za to odpowiedzialna linijka 21). Wywołanie destruktora oznacza działanie przeciwne do konstruktora, czyli spadek liczby pracowników (np. zwolnienie pracownika), co też powoduje konieczność zmiany przechowywanej wartości. W definicji destruktora zawrzemy fragment kodu, który będzie odpowiedzialny za dekrementację przechowywanej w słowniku liczby pracowników. Tworzony przez nas destruktor, oprócz zmniejszenia wartości zmiennej symbolizującej liczbę pracowników (w kodzie poniżej jest za to odpowiedzialna linijka 26), powinien również wyświetlić komunikat, jaki obiekt jest usuwany. Oprócz konstruktora i destruktora dodamy również funkcję wypisującą na standardowe wyjście aktualną liczbę pracowników, funkcję tę wykorzystamy również w destruktorze do wyświetlenia zmodyfikowanego stanu zmiennej.

```
class Pracownik:
    # Tworzymy słownik do przechowywania liczby pracowników
    slownik = {"liczba pracownikow": 0}

```

```
    def __init__(self, imie: str, nazwisko: str, plec: str = "M
        self.imie = imie
        self.nazwisko = nazwisko
        # sprawdzamy poprawność płci
        if plec != "K" and plec != "M":
            print("Niepoprawny format płci!")
            self.plec = "M"
        else:
            self.plec = plec
        # sprawdzamy poprawność numeru telefonu
        if (len(str(telefon)) != 9 or type(telefon) != int) and
            print("Niepoprawny format telefonu!")
            self.telefon = None
        else:
            self.telefon = telefon
        # Tworzymy nowego pracownika, więc zwiększamy ich liczb
        self.slownik["liczba pracownikow"] += 1

    def __del__(self):
        print("Usunięto pracownika", self.imie, self.nazwisko)
        # Usuwamy pracownika, więc musimy zmniejszyć ich liczbę
        self.slownik["liczba pracownikow"] -= 1
        # wypisujemy liczbę pracowników po dekrementacji
        self.liczba_pracownikow()

    def pokaz_informacje(self):
        print("Pracownik:", self.imie, self.nazwisko, ",płeć:",

    # Pobieramy ze słownika liczbę pracowników i wypysujemy ją
    def liczba_pracownikow(self):
        pracownicy = self.slownik.get("liczba pracownikow")
        print("Aktualna liczba pracowników:", pracownicy)
```

Przetestujemy teraz, jak działa licznik osób w firmie, a także usuwanie obiektów pracowników, wykorzystując funkcję `del()`.

```
# Metodę liczba_pracownikow() możemy wywołać dla klasy, jeśli u
Pracownik.liczba_pracownikow(Pracownik)
adam = Pracownik("Adam", "Woźniak")
Pracownik.liczba_pracownikow(Pracownik)
agata = Pracownik("Agata", "Lewandowska", "K", 123456789)
```

```
agata.liczba_pracownikow()
michal = Pracownik("Michał", "Adamczyk", "M", 421236790)

# Metodę  liczba_pracownikow() możemy wywołać na dowolnym obiek
agata.liczba_pracownikow()

# Usuwamy obiekt
del(michal)
```

Wynik działania programu:

```
Aktualna liczba pracowników: 0
Aktualna liczba pracowników: 1
Aktualna liczba pracowników: 2
Aktualna liczba pracowników: 3
Usunięto pracownika Michał Adamczyk
Aktualna liczba pracowników: 2
```

Jeżeli po uruchomieniu powyższego kodu, pozwolimy mu zakończyć działanie bez przerywania (np. punktami przerwań w trybie *debug* kompilatora), zauważymy, że na standardowym wyjściu oprócz oczekiwanych komunikatów pojawią się również następujące:

> **Ważne!**
>
> Wystąpienie wspomnianego wcześniej komunikatu zależy od używanego interpretera. W przypadku używania interpretera IDLE komunikat się nie pojawi, jednak jeśli używasz interpretera PyCharm – tak.

```
# Usunięto pracownika Adam Woźniak
# Aktualna liczba pracowników: 1
# Usunięto pracownika Agata Lewandowska
# Aktualna liczba pracowników: 0
```

Jest to spowodowane faktem, że pod koniec wykonywania skryptu wciąż istniejące obiekty wychodzą z zasięgu, co powoduje, że ich destruktory zostają wywołane podczas działania odśmiecacza.

> **Ważne!**

- Bez jawnie zdefiniowanego konstruktora obiekt zostanie utworzony za pomocą konstruktora domyślnego stosowanego w takich sytuacjach przez język Python do tworzenia obiektów. Konstruktor taki nie robi z obiektem nic, poza umożliwieniem jego stworzenia (nie zmienia jego zmiennych wewnętrznych i nie wywołuje metod).
- Bez jawnie zdefiniowanego destruktora język Python zastosuje do obsługi danego obiektu destruktor domyślny, który wykona jedynie podstawowe zadania destruktora.
- Konstruktor i destruktor mogą zawierać kod odnoszący się do elementów konkretnego obiektu oraz do współdzielonych elementów klasy.

Już wiesz

- Konstruktora używamy, aby zainicjować pola nowego obiektu klasy.
- Konstruktor może zawierać dowolny blok kodu.
- Destruktora używamy, jeśli chcemy wykonać blok kodu w momencie usuwania obiektu z pamięci.

Słownik

dunder

(ang. *double under*) funkcja, metoda lub pole specjalnego przeznaczenia w języku Python

odśmiecacz

(ang. *garbage collector*) aplikacja odpowiadająca za automatyczne czyszczenie pamięci, gdy znajdują się w niej niepotrzebne już dane

przestrzeń nazw

(ang. *namespace*) pakiet nazw zmiennych wraz z ich wartościami, przechowywanych w postaci słownika `dict`; przestrzenie nazw są przypisane m.in. do funkcji, modułów, klas

Prezentacja multimedialna

Polecenie 1

Przeanalizuj działanie konstruktora i destruktora dla obiektu klasy `Samochod`. Zdefiniuj kilka własnych klas, zawierających co najmniej dwa pola, konstruktor, destruktor oraz jedną dodatkową metodę, a następnie przeanalizuj ich działanie.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Stwórzmy klasę `Samochod` zawierającą pola pozwalające na opisanie danego pojazdu. Niech będą to: marka, liczba drzwi i rodzaj silnika. Klasa ta powinna zawierać również konstruktor, destruktor i funkcję pozwalającą na wyświetlenie tych informacji. Niech konstruktor i destruktor wyświetlają komunikaty sygnalizujące tworzenie i usuwanie obiektu. Definicja tej klasy w praktyce może wyglądać następująco:

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Korzystając z konstruktora, tworzymy nowy obiekt o określonych wartościach pól.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Funkcja inicjująca tworzy obiekt – jako parametr `self` przypisuje odpowiednią nazwę obiektu. Dodatkowo konstruktor wypisuje informacje na ekranie. Z reguły przy kolejnych wywołaniach wartości zwracane przez funkcję `id()` będą się różnić, ponieważ jest to dynamicznie przydzielany adres obiektu w pamięci. Komunikat z przykładową wartością `id` prezentuje się następująco:

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Możemy sprawdzić, jak zapisane są wartości, które podaliśmy przy inicjacji obiektu, wykorzystując metodę `info()`.

Zwróci ona następujący komunikat:

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

W momencie usuwania obiektu z pamięci funkcją `del(obiekt)` wywoływany jest destruktor.

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Zatem usunięcie obiektu poleceniem:

poskutkuje następującym komunikatem:

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Stwórzmy obiekt z wykorzystaniem konstruktora z parametrami.

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Funkcja inicjująca tworzy obiekt – jako parametr `self` przypisuje odpowiednią nazwę obiektu. Dodatkowo konstruktor wypisuje informacje na ekranie. Jak już wspomniano, funkcja `id()` zwraca różne wartości w kolejnych wywołaniach, zatem wartość `id` wyświetlona przy wykonywaniu kodu może się różnić od następującej:

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Możemy sprawdzić, jak zapisane są wartości, które podaliśmy przy inicjacji obiektu, wykorzystując metodę `info()`.

Skutkiem tego polecenia będzie wypisanie wartości zapisanych w zmiennych wewnętrznych obiektu:

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

W momencie usuwania obiektu z pamięci funkcją `del(obiekt)` wywoływany jest destruktor.

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6SDfh0Ep

Podobnie jak poprzednio, usunięcie obiektu:

poskutkuje wywołaniem jego destruktora i wypisaniem stosownego komunikatu:

Tu znajduje się cały kod omówiony w prezentacji:

```
class Samochod:

    def __init__(self, marka: str, liczba_drzwi: int,
                 rodzaj_silnika: str):
        self.marka: str = marka
        self.liczba_drzwi: int = liczba_drzwi
        self.rodzaj_silnika: str = rodzaj_silnika
        print("Id nowego obiektu to:", id(self))
        print("Podano markę:",marka)

    def __del__(self):
        print("Usuwamy z pamięci obiekt o id:", id(self))

    def info(self):
        print("Samochód marki:", self.marka,"z silnikiem typu:",


samochod_1 = Samochod('Volkswagen', 4, 'diesel')
# Id nowego obiektu to: 1876826013408
# Podano markę: Volkswagen

samochod_1.info()
# Samochód marki: Volkswagen z silnikiem typu diesel.

del (samochod_1)
# Usuwamy z pamięci obiekt o id 1876826013408

samochod_2 = Samochod('Ford', 4, 'benzynowy')
# Id nowego obiektu to: 1876826013408
# Podano markę: Ford

samochod_2.info()
# Samochód marki: Ford z silnikiem typu benzynowy.

del (samochod_2)
# Usuwamy z pamięci obiekt o id 1876826013408
```

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Dokończ zdanie.
W języku Python konstruktor...

- ◯ jeśli jest zdefiniowany, nie musi być użyty.
- ◯ może zostać zdefiniowany.
- ◯ musi być zdefiniowany.

Ćwiczenie 2

Wybierz prawdziwe stwierdzenia dotyczące destruktora w języku Python

- ☐ Jest reprezentowany przez metodę `__init__()`.
- ☐ Jest wywoływany automatycznie.
- ☐ Jest wywoływany, kiedy obiekt jest usuwany lub niszczony.
- ☐ Musi być zdefiniowany.

Ćwiczenie 3

Wskaż poprawny zapis dla konstruktora.

```
imie = Nazwisko("Python")
```

- [ ]
```
class Nazwisko:
    __init__(self, podane):
        self.nazwisko = podane
```

- [ ]
```
class Nazwisko:
    def __init__(self, podane):
        self.nazwisko = podane
```

- [ ]
```
class Nazwisko(podane):
    def __init__(self):
        self.nazwisko = podane
```

Ćwiczenie 4

Zdefiniuj klasę `Prostokat`, której konstruktor nada wartość dwóm polom: `długosc` oraz `szerokosc`.

Niech utworzenie obiektu wymaga jawnego podania obu parametrów.

Zdefiniuj destruktor, który będzie wypisywał `'prostokat usuniety'`.

Swój program przetestuj dla wartości `dlugosc` równej 10 oraz `szerokosc` wynoszącej 20.

Specyfikacja problemu:

Dane:

- `dlugosc` – liczba całkowita
- `szerokosc` – liczba całkowita

Wynik:

- `dlugosc` – wartość pola `dlugosc` obiektu wypisana w formacie `'dlugosc = `*wartość*
- `szerokosc` – wartość pola `szerokosc` obiektu wypisana w formacie `'szerokosc = `*wartość*
- wypisany przez destruktor komunikat o treści podanej w poleceniu

Twoje zadania

1. Program wypisuje w oddzielnych liniach wartości pól obiektu, a następnie go usuwa.

```
class Prostokat:
    # tutaj Twój kod
    pass

```

```
5 # poniżej przykładowe wywołanie
```

```
1
```

Ćwiczenie 5

Zdefiniuj klasę `Suma(parametr1: int, parametr2: float)`, której konstruktor nada polu o nazwie `wynik` wartość będąca sumą parametrów: `parametr1` i `parametr2`.
W tworzonej klasie zdefiniuj również destruktor. Program przetestuj dla wartości 5 i 7.25.

Specyfikacja problemu:

Dane:

- `parametr1` – podana w poleceniu liczba całkowita
- `parametr2` – podana w poleceniu liczba zmiennoprzecinkowa

Wynik:

- `wynik` – liczba zmiennoprzecinkowa; obliczona wartość pola `wynik` w formacie '`wynik` = *wartość* '

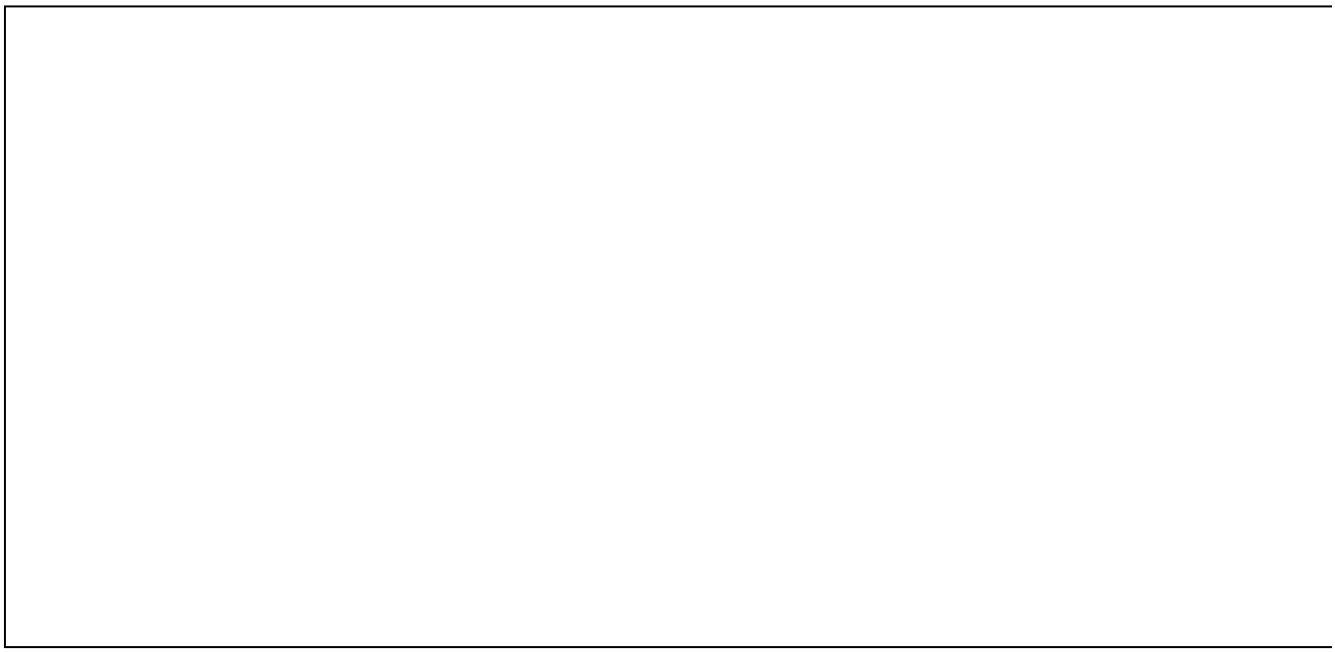
Twoje zadania

1. Program tworzy obiekt klasy `Suma`, a nastepnie wyświetla wartość jego pola `wynik`.

```
class Suma:
    # tutaj Twój kod
    pass

# poniżej przykładowe wywołanie
```

```
1
```



Dla nauczyciela

Autor: Adam Jurkiewicz
Przedmiot: Informatyka

Temat: Konstruktory i destruktory w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

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:
> >
> > > 1) projektuje i tworzy rozbudowane programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur;
> > >
> > > 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;
> > >
> > > 3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

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):

- Wyjaśnisz, jakie funkcje spełniają konstruktory i destruktory klas w języku Python.
- Prześledzisz, jak stworzyć konstruktor oraz destruktor klasy w języku Python.
- Przeanalizujesz działanie programów z zaimplementowanymi konstruktorami oraz destruktorami.
- Zaimplementujesz w języku Python programy wykorzystujące konstruktor oraz destruktor.

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;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Konstruktory i destruktory w języku Python". Uczniowie mają zapoznać się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Następnie wspólnie z uczniami ustala kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły,

a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
   W kolejnym kroku na forum klasy uczniowie analizują przedstawione w niej rozwiązania Przykładów 1, 2, 3 i 4 i powtarzają je na swoich komputerach.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie wspólnie analizują działanie konstruktora i destruktora dla obiektu opartego na klasie `Samochod`.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1 i 2 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 3 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Na koniec zajęć z programowania w Pythonie nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie wykonują ćwiczenia 4 i 5 z sekcji „Sprawdź się".

Materiały pomocnicze:

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

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.