Класове и обекти в Python

В темата ще бъдат разгледани следните основни въпроси:

- Същност на Python като обектно-ориентиран език
- Класове и обекти
- Наследяване

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

1. Същност на Python като обектно-ориентиран език

Програмният език Python е *обектно-ориентиран език* за програмиране и *всичко в Python е обект*.

В обектно-ориентираните езици за програмиране като Python, обектът е *инстанция*, който съдържа данни заедно със свързани с тях метаданни и/или функционалности. В Python всичко е обект, което означава, че всеки обект има някои метаданни, наречени атрибути, и свързани с тях функционалности, наречени методи. Тези атрибути и методи са достъпни чрез синтаксиса точка „ . “.

Например, в примера от фиг. 3.12. бе показано, че списъците имат метод за добавяне append(), който добавя елемент към списъка и е достъпен чрез синтаксиса точка „ **.** “.

Преди въвеждането на класовете и обектите в Python, за по-доброто им възприемане, първо ще бъде пояснена важна характеристика на променливите в Python.

В Python, противоположно на C++, *променливите е най-добре да се разглеждат като указатели, а не като контейнери*. Така че в Python, когато се пише

x = 4

по същество се *дефинира указател с име x*, който сочи към някакъв друг контейнер в паметта, съдържащ стойността 4. Тъй като променливите на Python просто сочат към различни обекти, те няма нужда да се „декларират“.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Дори не се изисква променливата да сочи винаги еднотипна информация. Поради тази причина се казва, че Python е *динамично типизиран*, а именно *имената на променливите могат да сочат към обекти от всякакъв тип*.

Това ще бъде пояснено с примера на фиг. 3.17.

```
x = 4
print(type(x))

x = 'hello'
print(type(x))

x = 3.14159
print(type(x))
```

а)

```
<class 'int'>
<class 'str'>
<class 'float'>
```

б)

Фиг. 3.17. Пример за динамично типизиране в Python – променливата x сочи към обекти от различен тип **а.** Код на Python **б.** Изход от програмата

2. Класове и обекти

Класът е прототип, по който могат да се конструират множество обекти или „план“ за създаване на обекти. Класът се декларира със запазената дума class, а за създаването на обект се извиква конструктора на класа (фиг. 3.18).

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
class MyClass: # Създаване на клас с име MyClass и атрибут x
  x = 7

MyC = MyClass() # Създаване на обект MyC от класа MyClass
print(MyC.x)
```

а)

```
7
```

б)

Фиг. 3.18. Пример за създаване на клас и обект с атрибут x

а. Код на Python **б.** Изход от програмата

Примерът 3.18 създава клас и обект в най-простата възможна форма и не е полезен в реалните приложения.

За да се разбере значението на класовете, трябва да обърне внимание на тяхната вградена функция __init__(). Тя винаги се изпълнява, когато се създава обект от съответния клас, което се нарича инициализиране на класа. Функцията __init__() се използва, за да се присвоят стойности на атрибутите (свойствата) на обекта или други операции, които е необходимо да се изпълнят, когато обектът се създава. Функцията __init__() се извиква автоматично всеки път, когато класът се използва за създаване на нов обект, дори тялото и да е празно.

Пример за използване на функцията __init__() в класа Person с два атрибута, име на човека и възраст на човека, е показан на фиг. 3.19.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Както се вижда от изхода на програмата, когато се опитва да изведе стойността на обекта p1, Python извежда стойността на адреса на първия байт в паметта (0x7f88ab1a4400), където е разположен обекта, в шестнадесетичен код, защото обектът p1 реално е указател. Третият ред от изхода е вградената в езика Python стрингово представяне на обекта. Ако програмистът желае да предостави ново стрингово представяне на обекта трябва да пренапише вградената функция __str__().

Пример за използване на функциите __init__() и __str__() в класа Person с два атрибута, име на човека и възраст на човека, е показан на фиг. 3.20.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("Иван", 23)

print(p1.name)
print(p1.age)
print(p1)
```

а)

```
Иван
23
<__main__.Person object at 0x7f88ab1a4400>
```

б)

Фиг. 3.19. Пример за използване на функцията __init__() в класа Person
а. Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def __str__(self):
    return f"{self.name} ({self.age} г.)"

p1 = Person("Иван", 23)

print(p1.name)
print(p1.age)
print(p1)
```

а)

```
Иван
23
Иван (23 г.)
```

б)

Фиг. 3.20. Пример за използване на функциите __init__() и __str__() в класа Person **а.** Код на Python **б.** Изход от програмата

За да се добави допълнителна функционалност на класа се използват методи, които са дефинирани в класа функции. Пример за дефиниране на функция Hello, представяне на човека е показана на фиг. 3.21.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def Hello(self):
    print("Здравейте. Моето име е " + self.name + " и съм
на " + str(self.age) + " г.")

p1 = Person("Иван", 23)
p1.Hello()
```

а)

```
Здравейте. Моето име е Иван и съм на 23 г.
```

б)

Фиг. 3.21. Пример за използване на метода Hello(self) в класа Person и извикването му с обекта p1 **а.** Код на Python **б.** Изход от програмата

Използваният до тук параметър self е указател към текущия екземпляр на класа и се използва за достъп до променливи, които принадлежат на класа.

Не е задължително той да се нарича self, а може да му се даде каквото и да е друго име, но той трябва задължително да е първия параметър на всяка функция в класа. Указателят self е подобен на указателя this в C++.

За изтриване на атрибут или на целия обект се използва ключовата дума del. Пример за изтриване на атрибута age показан на фиг. 3.22, а на целия обект p1 – на фиг. 3.23.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def Hello(self):
    print("Здравейте. Моето име е " + self.name + " и съм
на " + str(self.age) + " г.")

p1 = Person("Иван", 23)
del p1.age
print(p1.age)
```

а)

```
Traceback (most recent call last):
  File "main.py", line 19, in <module>
    print(p1.age)
AttributeError: 'Person' object has no attribute 'age'
```

б)

Фиг. 3.22. Пример за използване на ключовата дума del за изтриване на атрибута age **а.** Код на Python **б.** Изход от програмата

3. Наследяване

Наследяването позволява да се дефинира клас, който наследява всички методи и атрибути от друг клас.

Родителският клас е класът, от който се наследява, и се нарича още *базов клас*.

Наследственият клас е класът, който наследява от друг клас, и се нарича още *дъщерен, производен клас*.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

В Python всеки клас може да бъде родителски клас, така че синтаксисът за дефиниране на родителски клас е същият като при създаването на всеки друг клас.

За да се създаде наследствен клас, който наследява функционалността от друг родителски клас, се използва родителския клас като параметър, когато се създава дъщерния клас.

Ако се създаде клас Student, наследник на Person, с празно тяло, той ще наследи всички атрибути и методи на класа Person (фиг. 3.24)

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def Hello(self):
    print("Здравейте. Моето име е " + self.name + " и съм
на " + str(self.age) + " г.")

p1 = Person("Иван", 23)
del p1
print(p1)
```

а)

```
Traceback (most recent call last):
  File "main.py", line 21, in <module>
    print(p1)
NameError: name 'p1' is not defined
```

б)

Фиг. 3.23. Пример за използване на ключовата дума del за изтриване на обекта p1 **а.** Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def Hello(self):
    print("Здравейте. Моето име е " + self.name + " и съм
на " + str(self.age) + " г.")

class Student(Person):
  pass

s1 = Student("Иван Иванов", 21)
s1.Hello()
```

а)

```
Здравейте. Моето име е Иван Иванов и съм на 21 г.
```

б)

Фиг. 3.24. Пример за използване на родителски клас Person и наследствен клас Student **а.** Код на Python **б.** Изход от програмата

Обикновено наследствените класове добавят някакви атрибути (свойства) на обекта и свързани с тях методи за обработка. В този случай е необходимо да се напише функция __init__(), която да инициализира новите атрибути.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Когато се добави функцията __init__() в дъщерния клас, той вече няма да наследява функцията __init__() на родителя. За да се запази наследяването на функцията __init__() на родителя, е необходимо да се добави извикване на функцията __init__() на родителя (фиг. 3.25)

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def Hello(self):
    print("Здравейте. Моето име е " + self.name + " и съм
на " + str(self.age) + " г.")

class Student(Person):
  def __init__(self, name, age, specialty):
    Person.__init__(self, name, age)
    self. specialty = specialty

s1 = Student("Иван Иванов", 21, "КСТ")
print(s1.name)
print(s1.age)
print(s1.specialty)
```

а)

```
Иван Иванов
21
КСТ
```

б)

Фиг. 3.25. Пример за използване на родителски клас Person и наследствен клас Student с допълнителен атрибут specialty

а. Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Python има вградена функция super(), която ще накара дъщерния клас да наследи всички методи и свойства от своя родител. Като се използва функцията super(), не е необходимо да се използва името на родителския клас (виж фиг. 3.26). На фиг. 3.26 е показано и добавянето на метод welcome за обработка на атрибутите на дъщерния клас Student.

Ако се добави метод в дъщерния клас със същото име като функция в родителския клас, наследяването на родителския метод ще бъде заменено.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def Hello(self):
    print("Здравейте. Моето име е " + self.name + " и съм
на " + str(self.age) + " г.")

class Student(Person):
  def __init__(self, name, age, specialty):
    super().__init__(name, age)
    self.specialty = specialty
  def welcome(self):
    print("Добре дошли", self.name, "в специалност",
self.specialty)

s1 = Student("Иван Иванов", 21, "КСТ")
s1.welcome()
```

а)

```
Добре дошли Иван Иванов в специалност КСТ
```

б)

Фиг. 3.26. Пример за използване на родителски клас Person и наследствен клас Student с допълнителен атрибут specialty
а. Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.