Python i Pygame

Programiranje arkadnih igrica u Pythonu koristeći Pygame

Uvod u klase

Zašto klase?

- Svaki lik u igrici ima niz podataka koji ga opisuju: ime, poziciju na ekranu, snagu, smjer u kojem se kreće, itd.
  Također, likovi izvršavaju razne radnje: trče, skaču, udaraju, pričaju, itd.
- Varijable koje sadrže podatke mogu izgledati ovako:

```
ime = "Link"
spol = "Male"
max_hit_bodovi = 50
trenutno_hit_bodovi = 15
```

- Kako bi napravili bilo kakvu akciju sa likom, trebamo prenijeti te podatke nekoj funkciji koja definira tu akciju, npr:

```
def prikazi_lik(ime, spol, max_hit_bodovi, trenutno_hit_bodovi):
    print(ime, spol, max_hit_bodovi, trenutno_hit_bodovi)
```

- Broj podataka i funkcija u igrici može biti vrlo veliki, pa je to moguće napraviti na jednostavniji način – uvođenjem klasa i objekata

Klasa

- Klasa je struktura koja u sebi sadrži sva podatke koji opisuju neki lik, ali i sve akcije koje taj lik može napraviti, npr:

```
class Lik():
    """ Ovo je klasa koja opisuje glavnog lika u igrici """
    def __init__(self):
        """ Ovdjer se opisuju podaci lika """
        self.ime = "Link"
        self.spol = "Male"
        self.max_hit_bodovi = 50
        self.trenutno_hit_bodovi = 15
```

- Običaj je da se imena klasa pišu sa velikim slovom
- Podaci definirani unutar klase se obično nazivaju atributima klase
- def __init__(self): je posebna funkcija koja mora biti unutar svake klase i naziva se konstruktor
- Nakon što definiramo skup podataka (klasu) koja opisuje lik u igrici, potrebno je kreirati i samog lika

Objekt

- Lik je objekt koji se kreira na temelju definicija koje se nalaze u klasi, na slijedeći način:

```
glavni_lik = Lik()
```

- Varijabla glavni_lik sadrži referencu (memorijsku adresu) na objekt Lik
- Nakon kreiranja lika (objekta), tom liku se mogu dodijeliti podaci na slijedeći način:

```
glavni_lik.ime = "Han Solo"
glavni_lik.spol = "Male"
glavni_lik.max_hit_bodovi = 1500
glavni_lik.trenutno_hit_bodovi = 168
```

Dodavanje akcije klasi

- Akcije su funkcije koje se definiraju unutar klase i odnose se na objekte klase. Nazivaju se metodama

```
class Dog():
    def __init__(self):
        self.age = 0
        self.name = ""
        self.weight = 0

    def bark(self):
        print("Woof")
```

- Prvi parametar svake metode mora biti self
- Primjer korištenja klase Dog:

```
my_dog = Dog()

my_dog.name = "Fido"
my_dog.weight = 20
my_dog.age = 3

my_dog.bark()
```

Reference

- Referenca je pokazivač (pointer) na lokaciju u memoriji gdje se nalazi objekt. Primjer:

```
class Person():
    def __init__(self):
        self.name = ""
        self.money = 0
bob = Person()
bob.name = "Bob"
bob.money = 100
nancy = bob
nancy.name = "Nancy"
print(bob.name, "has", bob.money, "dollars.")
print(nancy.name, "has", nancy.money, "dollars.")
```

- bob je referenca na objekt u memoriji. nancy = bob dodjeljuje varijabli nancy istu adresu u memoriji koja je spremjena u varijabli bob. Zato je rezultat:

```
Nancy has 100 dollars.
Nancy has 100 dollars.
```

Funkcije i reference

- Referenca se može prenijeti kao parametar u funkciju. Primjer:

```
def give_money(person):
    person.money += 100

class Person():
    def __init__(self):
        self.name = ""
        self.money = 0

bob = Person()
bob.name = "Bob"
bob.money = 100

give_money(bob)
print(bob.money)
```

Ispisuje 200

Konstruktor

- Konstruktor je posebna funkcija klase koja se poziva svaki puta kada se kreira objekt iz neke klase. Konstruktor uvijek nosi naziv __init__(). Primjer:

```
class Dog():
    def __init__(self):
        self.age = 0
        self.name = "Fido"
        self.weight = 0
        print("A new dog is born!")

my_dog = Dog()

print(my_dog.name)
```

Ispisuje:
A new dog is born!
Fido

Nasljeđivanje (Inheritance)

- Nasljeđivanje je mogućnost kreiranja nove klase na osnovu postojeće klase. Nova klasa od postojeće nasljeđuje atribute i metode. Primjer:

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

class Employee(Person):
    def __init__(self):
        # Call the parent/super class constructor first
        super().__init__()
        # Now set up our variables
        self.job_title = ""
class Customer(Person):
    def __init__(self):
        super().__init__()
        self.email = ""

john_smith = Person()
john_smith.name = "John Smith"

jane_employee = Employee()
jane_employee.name = "Jane Employee"
jane_employee.job_title = "Web Developer"

bob_customer = Customer()
bob_customer.name = "Bob Customer"
bob_customer.email = "send_me@spam.com"
```

Varijable – Statične i objekta

- Varijable objekta su različite za svaki objekt. Statične varijable su iste za sve objekte. Primjer:

```
# Primjer varijable objekta
class ClassA():
    def __init__(self):
        self.y = 3

# Primjer statične varijable
class ClassB():
    x = 7

# Kreiranje objekata
a = ClassA()
b = ClassB()

# Statična varijabla se može ispisati na dva načina – drugi je školski.
print(b.x)
print(ClassB.x)

# Objektna varijabla se može ispisati na samo jedan način
print(a.y)
```

Kviz

- Slijedi link ispod:

  http://programarcadegames.com/quiz/quiz.php?file=classes&lang=en

Kviz odgovori

- P1: Select the best class definition for an alien?
  - class alien.name = ""
    class alien.height = 7.2
    class alien.weight = 156
  - . class alien():
      def __init__(self):
        self.name = ""
        self.height = 7.2
        self.weight = 156
  - class Alien():
      def __init__(self):
        self.name = ""
        self.height = 7.2
        self.weight = 156.
  - class alien(
      def __init__(self):
        self.name = ""
        self.height = 7.2
        self.weight = 156
)
- P2: What does this code do?

```
d1 = Dog()
d2 = Dog()
```

  - Creates two classes, of type Dog.
  - Creates one object, of type Dog.
  - Creates two objects, of type Dog.

Kviz odgovori, nastavak

- P3: What does this code do?

```
d1 = Dog()
d2 = d1
```

  - Creates two classes, of type Dog.
  - Creates one object, of type Dog.
  - Creates two objects, of type Dog.

- P4: What is wrong with the following code:

```
class Book():
    def open(self):
        print("You opened the book")
    def __init__(self):
        self.pages = 347
```

  - The __init__ with attributes should be listed first.
  - Book should not be capitalized.
  - There should be a self. in front of pages.
  - open should be capitalized.

Kviz odgovori, nastavak

- P5: What is wrong with the following code:

```
class Ball():
    def __init__(self):
        self.x = 0
        self.y = 0
        self.change_x = 0
        self.change_y = 0
    x += change_x
    y += change_y
```

  - The variables should be set equal to ""
  - The ball should not be at location 0, 0
  - All classes must have at least one method
  - The code to add to x and y must be in a method.

- P6: What is wrong with the following code:

```
class Ball():
    def __init__(self):
        self.x = 0
        self.y = 0
Ball.x = 50
Ball.y = 100
```

  - Ball. should be lower case.
  - Lines 3 and 4 should not have self. in front.
  - Ball. does not refer to an instance of the class.
  - Lines 3 and 5 should be used to set x and y to 50 and 100.

Kviz odgovori, nastavak

- P7: What is wrong with the following code:

```
class Ball():
    def __init__(self):
        self.x = 0
        self.y = 0
        b = Ball()
        b.x = 50
        b.y = 100
```

  - Line 6 should have self in between the parenthesis.
  - Lines 7 and 8 should have self. instead of b.
  - Lines 6-8 should not be indented.
  - Lines 6-8 should be in a method.

- P8: What will this print?

```
class Ball():
    def __init__(self):
        self.x = 0
        self.y = 0
b1 = Ball()
b2 = b1
b1.x = 40
b2.x = 50
b1.x += 5
b2.x += 5
print(b1.x, b2.x)
```

  - 55 55
  - 60 60
  - 40 40
  - 40 50

Kviz odgovori, nastavak

- P9: What will this print?

```
class Account():
    def __init__(self):
        self.money = 0
    def deposit(self, amount):
        self.money += amount
account = Account()
money = 100
account.deposit(50)
print(money, account.money)
```

  - 50 50
  - 150 150
  - 100 100
  - 100 50

- P10: What is wrong with the following:

```
class Dog():
    def __init__(self, new_name):
        """ Constructor.
        Called when creating an object of this type """
        name = new_name
        print("A new dog is born!")
# This creates the dog
my_dog = Dog("Rover")
```

  - On line 6, there should be a self. in front of name
  - On line 6, there should be a self. in front of new_name
  - Lines 9 and 10 should be indented.
  - Lines 6 to 7 should not be indented.