# -\*- coding: utf-8 -\*-  
  
from random import randint, choice  
  
  
# К атрибутам класса можно обратится и через объект  
class Lemming:  
 names = ['Peter', 'Anna', 'Nik', 'Sofi', 'Den', 'Lora', 'Bred', ]  
 tail\_length = 20  
  
 def \_\_init\_\_(self):  
 self.name = choice(Lemming.names)  
  
 def \_\_str\_\_(self):  
 return 'Lemming ' + self.name + ' with tail ' + str(self.tail\_length)  
  
  
print(Lemming.tail\_length)  
  
new\_lemming = Lemming()  
print(new\_lemming.tail\_length)  
print(new\_lemming)  
  
  
# Атрибут объекта перекрывает атрибут класса  
class Lemming:  
 names = ['Peter', 'Anna', 'Nik', 'Sofi', 'Denn', 'Lora', 'Bred', ]  
 tail\_length = 20  
  
 def \_\_init\_\_(self):  
 self.tail\_length = randint(15, 25)  
 self.name = choice(Lemming.names)  
  
 def \_\_str\_\_(self):  
 return 'Lemming ' + self.name + ' with tail ' + str(self.tail\_length)  
  
  
print(Lemming.tail\_length)  
  
new\_lemming = Lemming()  
print(new\_lemming.tail\_length)  
print(new\_lemming)  
  
  
# типичная ошибка  
class Lemming:  
 names = ['Peter', 'Anna', 'Nik', 'Sofi', 'Denn', 'Lora', 'Bred', ]  
 total = 0  
 tail\_length = 20  
  
 def \_\_init\_\_(self):  
 self.tail\_length = randint(15, 25)  
 self.name = choice(Lemming.names)  
 self.total = self.total + 1  
  
 def \_\_str\_\_(self):  
 return 'Lemming ' + self.name + ' with tail ' + str(self.tail\_length)  
  
  
burrow = []  
burrow\_depth = randint(90, 100)  
while len(burrow) < burrow\_depth:  
 family = []  
 family\_size = randint(16, 32)  
 while len(family) < family\_size:  
 new\_lemming = Lemming()  
 family.append(new\_lemming)  
 burrow.append(family)  
print(Lemming.total)  
print(len(burrow))  
  
  
# А что с обычными переменными? все так же как для функций  
  
  
class SomeClass:  
  
 def method\_one(self):  
 # x = 23  
 print('method\_one', x)  
  
 def method\_two(self):  
 # x = 34  
 def func\_one():  
 # x = 56  
 print('func\_one', x)  
 func\_one()  
 print('method\_two', x)  
  
  
x = 12  
obj = SomeClass()  
obj.method\_one()  
obj.method\_two()  
print('global', x)