-\*- coding: utf-8 -\*-  
  
# Существуют специальные методы, их вызов "встроен" в интерперетатор -  
# он автоматически их вызывает в определенных ситуациях.  
# Каких?  
# создание/удаление объектов  
# вызовы встроенных функций  
# преобразований объектов (приведения типов)  
# выполенение операторов языка  
# эмуляция вызова функции  
# работа с аттрибутами объектов  
  
# Такие методы выглядят как \_\_имя\_\_()  
  
  
# Рассмотрим онструктор \_\_init\_\_()  
# автоматичекси вызывается при создании объекта-экземпляра  
class Backpack:  
 *""" Рюкзак """* def \_\_init\_\_(self):  
 self.content = []  
  
 def add(self, item):  
 *""" Положить в рюкзак """* self.content.append(item)  
 print("В рюкзак положили:", item)  
  
 def inspect(self):  
 *""" Проверить содержимое """* print("В рюкзак лежит:")  
 for item in self.content:  
 print(' ', item)  
  
  
my\_backpack = Backpack()  
my\_backpack.add(item='ноутбук')  
my\_backpack.add(item='зарядка для ноутбука')  
my\_backpack.inspect()  
  
  
# \_\_init\_\_ может иметь параметры  
class Backpack:  
 *""" Рюкзак """* def \_\_init\_\_(self, gift=None):  
 self.content = []  
 if gift is not None:  
 self.content.append(gift)  
  
 def add(self, item):  
 *""" Положить в рюкзак """* self.content.append(item)  
 print("В рюкзак положили:", item)  
  
 def inspect(self, ):  
 *""" Проверить содержимое """* print("В рюкзак лежит:")  
 for item in self.content:  
 print(' ', item)  
  
  
my\_backpack = Backpack(gift='флешка')  
my\_backpack.add(item='ноутбук')  
my\_backpack.add(item='зарядка для ноутбука')  
my\_backpack.inspect()  
  
  
# аналогичный метод  
# object.\_\_del\_\_(self) - вызывается перед уничтожением объекта  
  
  
# рассмотрим  
# \_\_str\_\_ - вызывается при преобразовании объекта к строке str(obj)  
# например print(my\_backpack)  
  
class Backpack:  
 *""" Рюкзак """* def \_\_init\_\_(self, gift=None):  
 self.content = []  
 if gift:  
 self.content.append(gift)  
  
 def add(self, item):  
 *""" Положить в рюкзак """* self.content.append(item)  
 print("В рюкзак положили:", item)  
  
 def \_\_str\_\_(self):  
 return 'Backpack: ' + ', '.join(self.content)  
  
  
my\_backpack = Backpack(gift='телефон')  
my\_backpack.add(item='ноутбук')  
my\_backpack.add(item='зарядка для ноутбука')  
print(str(my\_backpack))  
  
  
# print(my\_backpack)  
# print(my\_backpack.\_\_str\_\_())  
  
# аналогичные методы  
# \_\_len\_\_ - вызывается для получения "размера" объекта с помощью функции len()  
# \_\_hash\_\_ - вызывается для получения уникального хэша объекта с помощью функции hash()  
# или для операций с хэширующими коллекциями - множества и словари  
# \_\_bool\_\_ - вызывается для получения "истинности" объекта с помощью функции bool()  
  
  
class Backpack:  
 *""" Рюкзак """* def \_\_init\_\_(self, gift=None):  
 self.content = []  
 if gift:  
 self.content.append(gift)  
  
 def add(self, item):  
 *""" Положить в рюкзак """* self.content.append(item)  
 print("В рюкзак положили:", item)  
  
 def \_\_str\_\_(self):  
 return 'Backpack: ' + ', '.join(self.content)  
  
 def \_\_bool\_\_(self):  
 return self.content != []  
  
 def \_\_len\_\_(self):  
 return len(self.content)  
  
  
my\_backpack = Backpack()  
# my\_backpack.add(item='ноутбук')  
print(bool(my\_backpack), len(my\_backpack))  
if my\_backpack:  
 print('Рюкзак не пуст!')  
 print('В нем лежит', len(my\_backpack), 'предметов')  
else:  
 print('Вот рюкзак пустой, он предмет простой...')  
  
  
# все специальные методы перечислены в  
# https://docs.python.org/3/reference/datamodel.html#special-method-names