**Модуль 4. Основы ООП.   
Обработка файлов. Мультимедиа**

Выполнил Жуков Артём Викторович

## Урок 2. ООП. Инкапсуляция

**Задание 1**

Напишите класс прямоугольника Rectangle, экземпляры которого при инициализации принимают два аргумента weight и height – ширину и высоту конкретного прямоугольника (соответствующие поля имеют такие же названия). Оба значения должны быть проверены на неотрицательность, если какой-то из переданных аргументов меньше нуля, то соответствующий атрибут объекта (экземпляра) должен принимать значение 0.

Класс Rectangle должен иметь следующие методы:

* area(), который возвращает площадь прямоугольника;
* perimeter(), который возвращает периметр прямоугольника;
* is\_square(), который возвращает True, если прямоугольник является квадратом, иначе – False;
* is\_in\_circle(radius), который проверяет можно ли поместить данный прямоугольник в круг с радиусом radius.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| rec = Rectangle(5, 3)  print(rec.area())  print(rec.perimeter())  if not rec.is\_square():  rec.weight, rec.height = (  max(rec.weight, rec.height),  max(rec.weight, rec.height)  )  print(rec.area())  print(rec.perimeter())  r = 1  while not rec.is\_in\_circle():  r += 1  print(r) | 15  16  25  20  8 |

**import** math

**class** Rectangle():

**def** \_\_init\_\_(self, weight, height):

**if** weight < 0:

self.weight = 0

**else**:

self.weight = weight

**if** height < 0:

self.height = 0

**else**:

self.height = height

**def** area(self):

**return** self.weight \* self.height

**def** perimeter(self):

**return** self.weight \* 2 + self.height \* 2

**def** is\_square(self):

**return** self.weight == self.height

**def** is\_in\_circle(self, radius):

g = math.sqrt(self.height \*\* 2 + self.weight \*\* 2)

**return** radius == (g // 2)

rec = Rectangle(5, 3)

print(rec.area())

print(rec.perimeter())

**if not** rec.is\_square():

rec.weight, rec.height = (

max(rec.weight, rec.height),

max(rec.weight, rec.height)

)

print(rec.area())

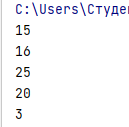
print(rec.perimeter())

r = 1

**while not** rec.is\_in\_circle(r):

r += 1

print(r)



## Урок 3. ООП. Полиморфизм

**Задание 1**

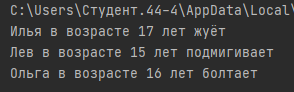
Известно, что в Стране ОЗ (или в Волшебной стране) живут Жевуны (Munchkin) Мигуны (Winkie) и Болтуны (Quadling). Каждый представитель этих народов имеет возраст и имя, но кое-что их разделяет: жевуны – жуют, мигуны – мигают, болтуны – болтают.

Создайте три класса Munchkin, Winkie и Quadling. Экземпляры этих классов при инициализации должны принимать по два аргумента: имя (name) и возраст (age). Данные классы должны иметь по два метода:

* info(), который возвращает строку вида «<name> в возрасте <age> лет»;
* act(), который возвращает строку вида «<действие>», где в качестве действия:
  + для Жевунов «жуёт»;
  + для Мигунов «подмигивает»;
  + для Болтунов «болтает».

|  |
| --- |
| **Ввод** |
| persons = [  Munchkin("Илья", 17), Winkie("Лев", 15), Quadling("Ольга", 16)  ]  for person in persons:  print(person.info(), person.act()) |
| **Вывод** |
| Илья в возрасте 17 лет жуёт  Лев в возрасте 15 лет подмигивает  Ольга в возрасте 16 лет болтает |

class Munchkin():  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def info(self):  
 return f'{self.name} в возрасте {self.age} лет'  
  
 def act(self):  
 return 'жуёт'  
  
  
class Winkie():  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def info(self):  
 return f'{self.name} в возрасте {self.age} лет'  
  
 def act(self):  
 return 'подмигивает'  
  
  
class Quadling():  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def info(self):  
 return f'{self.name} в возрасте {self.age} лет'  
  
 def act(self):  
 return 'болтает'  
  
  
persons = [  
 Munchkin("Илья", 17), Winkie("Лев", 15), Quadling("Ольга", 16)  
]  
  
for person in persons:  
 print(person.info(), person.act())



## Урок 4. ООП. Наследование

**Задание 1**

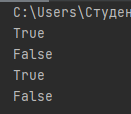
Органические циклические соединения (OrganicCyclic), о которых Вы узнаете (или уже знаете) из школьного курса химии, делятся на карбоциклические (CarboCyclic) и гетероциклические (HeteroCyclic) соединения. При этом карбоциклические часто разделяют на алициклические (AliCyclic) и ароматические (AromaticCyclic).

Ваша задача реализовать описанное дерево наследования, создав советующие классы; названия приведены в скобочках. Классы должны быть пустыми (для этого воспользуйтесь заглушкой pass).

|  |
| --- |
| **Ввод** |
| print(issubclass(AliCyclic, OrganicCyclic))  print(issubclass(OrganicCyclic, AliCyclic))  print(issubclass(AliCyclic, CarboCyclic))  print(issubclass(AromaticCyclic, HeteroCyclic)) |
| **Вывод** |
| True  False  True  False |

*Примечание*: функция issubclass(class1, class2) в данном конкретном случае проверяет, является ли class1 подклассом класса class2.

class OrganicCyclic():  
 pass  
  
class CarboCyclic(OrganicCyclic):  
 pass  
  
class HeteroCyclic(OrganicCyclic):  
 pass  
  
class AliCyclic(CarboCyclic):  
 pass  
  
class AromaticCyclic(CarboCyclic):  
 pass  
  
print(issubclass(AliCyclic, OrganicCyclic))  
print(issubclass(OrganicCyclic, AliCyclic))  
print(issubclass(AliCyclic, CarboCyclic))  
print(issubclass(AromaticCyclic, HeteroCyclic))



## Урок 5-6. ООП. Определение операторов

**Задание 1**

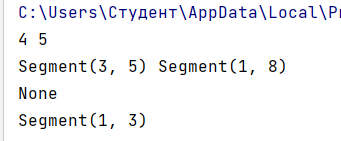
Напишите класс отрезков на прямой Segment. Класс должен иметь два поля: start\_point и end\_point – точки начала и конца отрезка. Экземпляры данного класса должны инициализироваться с помощью двух вещественных чисел-параметров, описывающих начало и конец отрезка. Если в качестве аргументов были переданы числа в порядке убывания, то началом отрезка считается наименьшее из них, концом – наибольшее.

Определите для этого класса операторы:

* \_\_len\_\_(), который возвращает вещественное число – длину отрезка;
* \_\_and\_\_(), который возвращает пересечение отрезков (новый отрезок), если отрезки пересекаются, иначе возвращает None;
* \_\_or\_\_(), который возвращает объединение отрезков (новый отрезок), если отрезки пересекаются, иначе возвращает None.
* \_\_str\_\_(), который возвращает строку вида «Segment(<start\_point>, <end\_point>)».

|  |
| --- |
| **Ввод** |
| first\_seg = Segment(1, 5)  second\_seg = Segment(3, 8)  print(len(first\_seg), len(second\_seg))  print(first\_seg & second\_seg, first\_seg | second\_seg)  print(Segment(3, 4) & Segment(1, 2))  print(Segment(1, 2) | Segment(2, 3)) |
| **Вывод** |
| 4 5  Segment(3, 5) Segment(1, 8)  None  Segment(1, 3) |

**class** Segment():  
 **def** \_\_init\_\_(self, start\_point, end\_point):  
 self.start\_point, self.end\_point = sorted([start\_point, end\_point])  
  
 **def** \_\_len\_\_(self):  
 **return** self.end\_point - self.start\_point  
  
 **def** \_\_and\_\_(self, other):  
 points = [self.start\_point, self.end\_point, other.start\_point, other.end\_point]  
 **if** points[0] > points[2]:  
 points[0], points[1], points[2], points[3] = \  
 points[2], points[3], points[0], points[1]  
 **if** points[1] < points[2]:  
 **return None  
 if** points[1] == points[2]:  
 **return None  
 return** Segment(points[1], points[2])  
  
 **def** \_\_or\_\_(self, other):  
 points = [self.start\_point, self.end\_point, other.start\_point, other.end\_point]  
 **if** points[0] > points[2]:  
 points[0], points[1], points[2], points[3] = \  
 points[2], points[3], points[0], points[1]  
 **if** points[1] < points[2]:  
 **return None  
 return** Segment(min(points[0], points[2]), max(points[1], points[3]))  
  
 **def** \_\_str\_\_(self):  
 **return f"Segment({**self.start\_point**}, {**self.end\_point**})"**first\_seg = Segment(1, 5)  
second\_seg = Segment(3, 8)  
  
print(len(first\_seg), len(second\_seg))  
print(first\_seg & second\_seg, first\_seg | second\_seg)  
  
print(Segment(3, 4) & Segment(1, 2))  
print(Segment(1, 2) | Segment(2, 3))



## Задания к уроку 15-16. Линейная программа. Регулирование вывода через sep и end

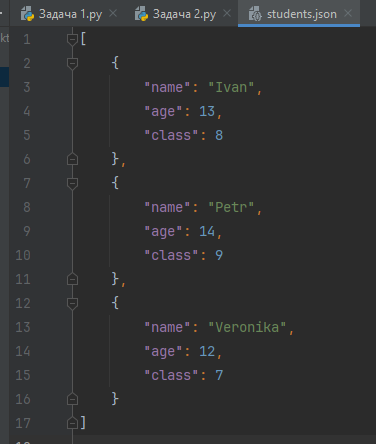
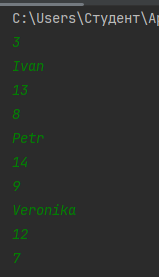
**Задание 1**

Напишите программу, которая позволит пользователю ввести информацию о нескольких учениках (имя, возраст, класс) и сохранить эту информацию в JSON-файл с названием **«students.json»**.

Программа должна принимать в первой строке число n – количество учеников, далее для каждого ученика программа принимает по три строки – имя, возраст, класс. Программа должна записать полученную информацию в JSON-файл.

|  |  |
| --- | --- |
| **Ввод** | **students.json** |
| 3  Ivan  13  8  Petr  14  9  Veronika  12  7 | [  {  "name": "Ivan",  "age": 13,  "class": 8  },  {  "name": "Petr",  "age": 14,  "class": 9  },  {  "name": "Veronika",  "age": 12,  "class": 7  }  ] |

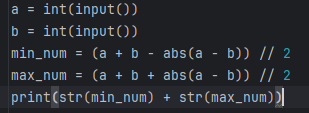
import json  
  
data=[]  
n=int(input())  
for \_ in range(n):  
 name,age,cls=input(),int(input()),int(input())  
 data.append({"name": name, "age": age, "class": cls})  
  
file\_name = "students.json"  
with open(file\_name, "w", encoding="utf-8") as json\_file:  
 json.dump(data, json\_file, indent=4)

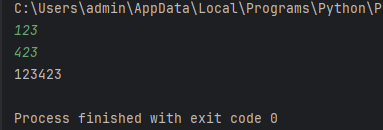


**Задание 5 (\*)**

*Без использования условных операторов, операторов отношения, функций min и max*, Вам необходимо написать программу, которая принимает в качестве входных данных два числа, после чего последовательно выводит наименьшее из низ, затем — наибольшее. Вывод происходит в одну строку без пробелов.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| 123  423 | 123423 |
| 265472 321 | 321265472 |



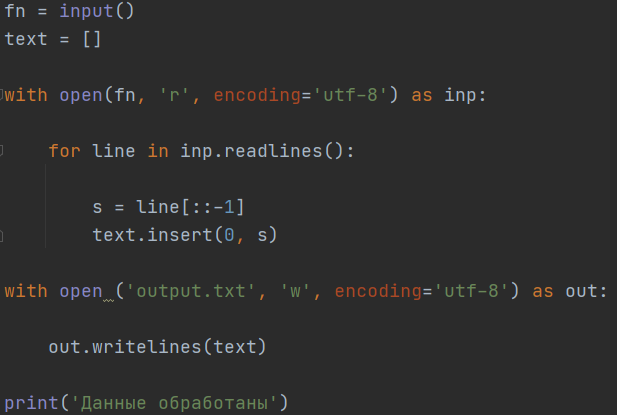


## Урок 17-19. Обработка файлов. Решение задач

**Задание 1**

Вам необходимо написать программу, которая принимает в качество входных данных одну строку – file\_name – название текстового файла, который содержит последовательность символов.

Ваша программу должна создать новый файл с названием *«output.txt»*, который будет состоять из символов изначального файла, записанных в обратном порядке.



|  |  |
| --- | --- |
| **input.txt** | |
| Говорят, мы бяки-буки,  Как выносит нас земля?  Дайте что ли карты в руки  Погадать на короля. | |
| **Ввод** | **output.txt** |
| input.txt | .ялорок ан ьтадагоП  икур в ытрак ил отч етйаД  ?ялмез сан тисоныв каК  ,икуб-икяб ым ,тяровоГ |

**Задание 4 (\*)**

Вам необходимо написать программу, которая считывает данные из JSON-файла, содержащего информацию о различных категориях продуктов. Затем программа должна разделить данные по категориям и сохранить их в виде отдельных CSV-файлов, где каждый CSV-файл представляет одну категорию продуктов.

Программа принимает в качестве входных данных строку – имя JSON-файла. Создаваемые CSV-файлы должны иметь имена в соответствии с названиями категорий, при этом сами обозначения категорий в CSV-файлы не вносятся.

|  |  |  |
| --- | --- | --- |
| **products.json** | | |
| [  {  "name": "Product A",  "price": 10.99,  "category": "Electronics"  },  {  "name": "Product B",  "price": 5.99,  "category": "Clothing"  },  {  "name": "Product C",  "price": 7.49,  "category": "Electronics"  },  {  "name": "Product D",  "price": 2.99,  "category": "Books"  }  ] | | |
| **Ввод** | | |
| products.json | | |
| **electronics.csv** | **clothing.csv** | **books.csv** |
| name;price  Product A;10.99  Product C;7.49 | name;price  Product B;5.99 | name;price  Product D;2.99 |

**Задание 5 (\*)**

У вас есть CSV-файл с данными о продажах, а также JSON-файл с информацией о клиентах. Напишите программу, которая связывает данные по общему ключу и создает новый JSON с объединенными данными для дальнейшего анализа.

Предположим, что у нас есть CSV-файл **«sales.csv»**, который содержит информацию о продажах, включая идентификатор заказа, идентификатор клиента, продукт и сумму:

|  |
| --- |
| **sales.csv** |
| order\_id;customer\_id;product;amount  1;C101;Widget A;100  2;C102;Widget B;150  3;C103;Widget A;200 |

И JSON-файл **«customers.json»**, в котором содержится информация о клиентах, где идентификатор клиента связывается с именем, электронной почтой и номером телефона клиента:

|  |
| --- |
| **customers.json** |
| {  "C101": {  "name": "John Doe",  "email": "john@example.com",  "phone": "123-456-7890"  },  "C102": {  "name": "Jane Smith",  "email": "jane@example.com",  "phone": "987-654-3210"  },  "C103": {  "name": "Michael Johnson",  "email": "michael@example.com",  "phone": "555-555-5555"  }  } |

Предполагается, что оба файла содержат идентификатор клиента **customer\_id**, который может быть использован как ключ для связи данных.

Тогда результат, который Вы должны получить, должен быть следующим:

|  |
| --- |
| **merged\_data.json** |
| {  "C101": {  "sale": {  "order\_id": "1",  "customer\_id": "C101",  "product": "Widget A",  "amount": "100"  },  "customer": {  "name": "John Doe",  "email": "john@example.com",  "phone": "123-456-7890"  }  },  "C102": {  "sale": {  "order\_id": "2",  "customer\_id": "C102",  "product": "Widget B",  "amount": "150"  },  "customer": {  "name": "Jane Smith",  "email": "jane@example.com",  "phone": "987-654-3210"  }  },  "C103": {  "sale": {  "order\_id": "3",  "customer\_id": "C103",  "product": "Widget A",  "amount": "200"  },  "customer": {  "name": "Michael Johnson",  "email": "michael@example.com",  "phone": "555-555-5555"  }  }  } |

## Урок 20-22. Подключение библиотеки playsound. Воспроизведение звука из файла

**Задание 1**

Напишите программу, которая принимает название аудиофайла с расширением «.mp3» или «.wav» и воспроизводит его с помощью библиотеки playsound.

Программа принимает в одной строке название (путь) аудиофайла, после чего воспроизводит его.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| audio\_file\_1.mp3 | \* воспроизводится audio\_file\_1.mp3 |
| audio\_file\_2.wav | \* воспроизводится audio\_file\_2.wav |

import playsound  
  
file = input("Введите путь к файлу: ")  
try:  
 print("\* воспроизводится")  
 print(file)  
 playsound.playsound(file)  
except Exception as e:  
 print(f"Ошибка воспроизведения {e}")

