Задание 1.

*"""  
Реализовать создание, запись, чтение и удаление файла с данными о пользователе.  
пользователь выбирает действие самостоятельно, а так же указывает путь к размещению файла.  
  
Обработать ошибки  
  
"""***import** os  
**from** random **import** randint  
  
  
**class** MetaClass():  
 **def** \_\_init\_\_(self):  
 d = {  
 0: AttributeError,  
 1: IOError,  
 2: IndexError,  
 3: KeyError,  
 }  
 key = randint(0, 3)  
 self.random\_class = d[key]  
  
  
**class** ExceptionClass(IndexError):  
 **def** \_\_init\_\_(self, description):  
 self.description = description  
  
 **def** \_\_str\_\_(self):  
 print(**"ExceptionClass \_\_str\_\_"**)  
 **return** self.description  
  
  
**class** FileProcessing():  
 **def** \_\_init\_\_(self):  
  
 select\_d = {  
 **"1"**: self.file\_add,  
 **"2"**: self.file\_remove,  
 **"3"**: self.file\_read,  
 **"4"**: self.file\_write,  
 }  
  
 self.file\_name = input(**"Введите название файла для записи -> "**)  
  
 input\_str = **""  
 while** input\_str != **"0"**:  
 input\_str = input(  
 **"1. Добавление файла\n2. Удаление файла\n3. Чтение из файла\n4. Запись файла\n0. Выход\n-> "**)  
 **try**:  
 select\_d[input\_str]()  
 **except** KeyError:  
 **if** input\_str != **"0"**:  
 print(**"Нет введёного пункта меню"**)  
  
 **def** file\_add(self):  
  
 f = open(self.file\_name, **"w"**)  
 f.close()  
  
 **def** file\_remove(self):  
 **try**:  
 os.remove(self.file\_name)  
  
 **if** self.random\_riser():  
 o = MetaClass()  
 **raise** o.random\_class(**"ну привет"**)  
  
 **except** (IOError, OSError) **as** e:  
 print(**"Ошибки удаления файла"**, e)  
 **except** KeyError **as** e:  
 print(**"KeyError:"**, e)  
 **except** Exception **as** e:  
 print(**"Вас посетила ошибка"**, type(e).\_\_name\_\_)  
  
 **def** file\_write(self):  
 *"""  
 Запись исходного выражения в файл  
 """* user\_info = input(**"Введите строку для записи -> "**)  
 **with** open(self.file\_name, **"w"**) **as** f:  
 f.write(user\_info)  
  
 **def** random\_riser(self):  
 *"""  
 Метод для рандомной отдачи 1 или 0  
 Необходим для выкидывания ошибки  
 """* **return** bool(randint(0, 1))  
  
 **def** file\_read(self):  
 *"""  
 Чтение файла self.file\_name  
 """* **try**:  
 **with** open(self.file\_name, **"r"**) **as** f:  
 print(f.read())  
  
 **if** self.random\_riser():  
 o = MetaClass()  
 **raise** o.random\_class(**"ну привет"**)  
  
 **except** FileNotFoundError:  
 print(**"Ошибка чтения файла. Файла не существует"**)  
 **except** Exception **as** e:  
 print(**"Вас посетила ошибка"**, type(e).\_\_name\_\_)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **try**:  
 FileProcessing()  
 **raise** ExceptionClass(**"привет, я кастомная ошибка"**)  
 **except** ExceptionClass **as** e:  
 print(e)  
 **except** KeyboardInterrupt:  
 print(**"\nВыход из программы, удачи!"**)

Задание 2.  
*"""  
Реализовать решение квадратного уравнения через дискриминант  
  
Обработать ошибки  
"""***import** math  
  
  
**class** DescrSolver():  
  
 **def** \_\_init\_\_(self, input\_str):  
 **try**:  
 self.input\_str = input\_str  
 **try**:  
 self.parse\_exp()  
 **except** (IndexError, IOError) **as** e:  
 print(e)  
 self.get\_descriminant()  
 **except** KeyError **as** e:  
 print(e)  
 **except** AttributeError **as** e:  
 print(e)  
  
 **def** is\_digital(self, number):  
 **try**:  
 float(number)  
 **return True  
 except** ValueError:  
 **return False  
  
 def** parse\_exp(self):  
 input\_str = self.input\_str  
  
 *# Определяем коэффициент a* symbols = 1  
 return\_flag = **False  
 while** return\_flag == **False**:  
  
 index\_a = input\_str[:symbols]  
 **if** self.is\_digital(index\_a) == **False and** symbols != 1:  
 return\_flag = **True** symbols += 1  
  
 index\_a = input\_str[:symbols - 2]  
  
 *# Определяем коэффициент b* symbols = input\_str.rindex(**"x"**)  
 buf\_index\_b = input\_str[:symbols]  
 symbols = 1  
 return\_flag = **False  
 while** return\_flag == **False**:  
  
 index\_b = buf\_index\_b[symbols:]  
  
 **if** self.is\_digital(index\_b) == **True**:  
 return\_flag = **True** symbols += 1  
  
 *# Определяем коэффициент c* symbols = input\_str.rindex(**"="**)  
 buf\_index\_c = input\_str[:symbols]  
 symbols = 1  
 return\_flag = **False  
 while** return\_flag == **False**:  
  
 index\_c = buf\_index\_c[symbols:]  
  
 **if** self.is\_digital(index\_c) == **True**:  
 return\_flag = **True** symbols += 1  
  
 print(**"Коэффициент a ="**, index\_a)  
 print(**"Коэффициент b ="**, index\_b)  
 print(**"Коэффициент c ="**, index\_c)  
  
 self.index\_a = float(index\_a)  
 self.index\_b = float(index\_b)  
 self.index\_c = float(index\_c)  
  
 **def** get\_descriminant(self):  
 a = self.index\_a  
 b = self.index\_b  
 c = self.index\_c  
  
 D = b \*\* 2 - 4 \* a \* c  
 print(**"D ="**, D)  
  
 **if** D == 0:  
 A1 = -b / (2 \* a)  
 A2 = A1  
 print(**"\nОтвет: "**)  
 print(**"Корень А1 ="**, A1)  
 print(**"Корень A2 ="**, A2)  
  
 **elif** D > 0:  
 A1 = (-b + math.sqrt(D)) / (2 \* a)  
 A2 = (-b - math.sqrt(D)) / (2 \* a)  
 print(**"\nОтвет: "**)  
 print(**"Корень А1 ="**, A1)  
 print(**"Корень A2 ="**, A2)  
  
 **else**:  
 print(**"Нет решения, D < 0"**)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **try**:  
 input\_str = input(**"Введите квадратное выражение вида ax^2+bx+c=0 -> "**)  
 DescrSolver(input\_str)  
 **except** KeyboardInterrupt:  
 print(**"Надеюсь на то, что мы еще встретимся"**)