Задание 1.
Исходные данные:
Реализовать класс «Дата», функция-конструктор которого должна принимать дату в виде строки формата «день-месяц-год». В рамках класса реализовать два метода. Первый, с декоратором @classmethod. Он должен извлекать число, месяц, год и преобразовывать их тип к типу «Число». Второй, с декоратором @staticmethod, должен проводить валидацию числа, месяца и года (например, месяц — от 1 до 12). Проверить работу полученной структуры на реальных данных.
Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Date:
...     def __init__(self, day, month, year):
...         self.day = day
...         self.month = month
...         self.year = year
...
>>> @classmethod
... def my_time_date(self):
...     return my_time_date(self.day, self.month, self.year)
...
>>> @staticmethod
... def my_time_date(cls):
...     if day==int:
...         print('Печатаем число')
...     else:
...         print('Введено неверное значение')
...
>>> @staticmethod
... def my_time_date(cls):
...     if month==int:
...         print('Печатаем число')
...     else:
...         print('Введено неверное значение')
...
>>> @staticmethod
... def my_time_date(cls):
...     if year==int:
...         print('Печатаем число')
...     else:
...         print('Введено неверное значение')
...
>>> a = dict(day=int, month=int, year=int)
>>> def my_date(a):
...     try:
...         b = int(input("Введите цифрами день: "))
...         c = int(input("Введите цифрами месяц: "))
...         d = int(input("Введите цифрами год: "))
...     except valueerror:
...         return
...     s = b, c, d
```

```
...     return s
...
>>> print(my_date(a))
Введите цифрами день: 5
Введите цифрами месяц: 9
Введите цифрами год: 2000
(5, 9, 2000)
>>>
```

Задание 2.
Исходные данные:
Создайте собственный класс-исключение, обрабатывающий ситуацию деления на ноль. Проверьте его работу на данных, вводимых пользователем. При вводе нуля в качестве делителя программа должна корректно обработать эту ситуацию и не завершиться с ошибкой.
Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Zero:
...     def __init__(self, number):
...         self.number = number
...
>>> @staticmethod
... def my_obj(cls):
...     if number>0 and number/0:
...         print('Выводить получившейся результат')
...     if number==0 and number/0:
...         print('Выводить получившейся результат')
...     if number<0 and number/0:
...         print('Выводить получившейся результат')
...
>>> a = 55
>>> def my_zero(a):
...     try:
...         if a/0:
...             a = int(input("Введите число: "))
...     except ZeroDivisionError:
...         return
...     else:
...         b = int(input("Введите число: "))
...         return
...     s = a / 0
...     return s
...     d = b / 2
...     return d
...
>>> print(my_zero(a))
None
>>>
```

Задание 3.

Исходные данные:

Создайте собственный класс-исключение, который должен проверять содержимое списка на наличие только чисел. Проверить работу исключения на реальном примере. Запрашивать у пользователя данные и заполнять список необходимо только числами. Класс-исключение должен контролировать типы данных элементов списка.

Примечание: длина списка не фиксирована. Элементы запрашиваются бесконечно, пока пользователь сам не остановит работу скрипта, введя, например, команду «stop». При этом скрипт завершается, сформированный список с числами выводится на экран.

Подсказка: для этого задания примем, что пользователь может вводить только числа и строки. Во время ввода пользователем очередного элемента необходимо реализовать проверку типа элемента. Вносить его в список, только если введено число. Класс-исключение должен не позволить пользователю ввести текст (не число) и отобразить соответствующее сообщение. При этом работа скрипта не должна завершаться.

Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Number:
...     def __init__(self, number):
...         self.number = number
...
>>> @staticmethod
... def my_obj(cls):
...     if number:
...         print('Печатать число')
...     else:
...         print('Ошибка, введите ещё раз число')
...
>>> a = 55
>>> b = 'Nika'
>>> c = 77
>>> my_list = [a, b, c]
>>> def my_list(a, b, c):
...     try:
...         if a==int:
...             a = int(input("Введите число: "))
...             print()
...         if b==str:
...             b = input("Введите текст: ")
...             print("Ошибка, введите ещё раз число")
...     except valueerror:
...         return
...     else:
...         c = int(input("Введите число: "))
...         return
...     s = my_list.append(a) and my_list.append(c)
...     return s
...
>>> print(my_list(a, b, c))
Введите число: 55
None
>>>
```

Задание 4.
Исходные данные:

Начните работу над проектом «Склад оргтехники». Создайте класс, описывающий склад. А также класс «Оргтехника», который будет базовым для классов-наследников. Эти классы — конкретные типы оргтехники (принтер, сканер, ксерокс). В базовом классе определите параметры, общие для приведённых типов. В классах-наследниках реализуйте параметры, уникальные для каждого типа оргтехники.

Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)

[GCC 9.3.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> class Store:

...     def __init__(self, orctechnics):

...         self.orctechnics = orctechnics

...

>>> class Orctechnics(Store):

...     def __init__(self, name, model):

...         self.name = name

...         self.model = model

...

>>> class Skaner(Orctechnics):

...     def __init__(self, number, type):

...         self.number = number

...         self.type = type

...         super().__init__()
```

```
...
>>> class Printer(Orctechnics):
...     def __init__(self, paper, size):
...         self.paper = paper
...         self.size = size
...         super().__init__()
...
>>> class Kseraks(Orctechnics):
...     def __init__(self, color, refueling):
...         self.color = color
...         self.refuiling = refuiling
...         super().__init__()
...
>>>
```

Задание 5.
Исходные данные:
Продолжить работу над первым заданием. Разработайте методы, которые отвечают за приём оргтехники на склад и передачу в определённое подразделение компании. Для хранения данных о наименовании и количестве единиц оргтехники, а также других данных, можно использовать любую подходящую структуру (например, словарь).
Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Store:
...     def __init__(self, orctechnics):
...         self.orctechnics = orctechnics
...
>>> class Orctechnics(Store):
...     def __init__(self, name, model):
...         self.name = name
...         self.model = model
...
>>> class Skaner(Orctechnics):
```

```
...     def __init__(self, number, type):
...         self.number = number
...         self.type = type
...         super().__init__()
...
>>> @staticmethod
... def my_skaner_params(cls):
...     if name:
...         print('Печатать название')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_skaner_params(cls):
...     if model:
...         print('Печатать модель')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_skaner_params(cls):
...     if number:
...         print('Печатать количество')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_skaner_params(cls):
...     if type:
...         print('Печатать тип сканера')
...     else:
...         print('Введены не верные данные')
...
>>> a = dict(name=str, model=str, number=int, type=str)
>>> def my_skaner(a):
...     try:
...         b = input("Введите название: ")
...         c = input("Введите модель: ")
...         d = int(input("Введите число: "))
...         f = input("Введите тип: ")
...     except valuerror and zerodivisionerror:
...         return
...     s = b, c, d, f
...     return s
...
>>> print(my_skaner(a))
Введите название: CanoSkan
Введите модель: Lida
Введите число: 1
Введите тип: Lazer
('CanoSkan', 'Lida', 1, 'Lazer')
>>>
```

```
>>> class Printer(Orctechnics):
...     def __init__(self, paper, size):
...         self.paper = paper
...         self.size = size
...         super().__init__()
...
>>> @staticmethod
... def my_printer_params(cls):
...     if name:
...         print('Печатать название')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_printer_params(cls):
...     if model:
...         print('Печатать модель')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_printer_params(cls):
...     if paper:
...         print('Печатать фирму изготовителя бумаги')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_printer_params(cls):
...     if size:
...         print('Печатать размер бумаги')
...     else:
...         print('Введены не верные данные')
...
>>> b = dict(name=str, model=str, paper=str, size=int)
>>> def my_printer(b):
...     try:
...         x = input("Введите название: ")
...         y = input("Введите модель: ")
...         z = input("Введите название фирмы производителя бумаги: ")
...         v = int(input("Введите размер бумаги: "))
...     except valueerror and zerodivisionerror:
...         return
...     s = x, y, z, v
...     return s
...
>>> print(my_printer(b))
Введите название: HP
Введите модель: lazer
Введите название фирмы производителя бумаги: inpress
Введите размер бумаги: 4
('HP', 'lazer', 'inpress', 4)
```

```
>>>
>>> class Kseraks(Orctechnics):
...     def __init__(self, color, refueling):
...         self.color = color
...         self.refueling = refueling
...         super().__init__()
...
>>> @staticmethod
... def my_kseraks_params(cls):
...     if name:
...         print('Печатать название')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_kseraks_params(cls):
...     if model:
...         print('Печатать модель')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_printer_params(cls):
...     if color:
...         print('Печатать цвет')
...     else:
...         print('Введены не верные данные')
...
>>> @staticmethod
... def my_kseraks_params(cls):
...     if refueling:
...         print('Печатать насколько процентов заплнен катредж ксеракса')
...     else:
...         print('Введены не верные данные')
...
>>> c = dict(name=str, model=str, color=str, refueling=int)
>>> def my_kseraks(c):
...     try:
...         k = input("Введите название: ")
...         l = input("Введите модель: ")
...         g = input("Введите цвет: ")
...         h = int(input("Введите значение заполненности катреджа в процентах: "))
...     except valueerror and zerodivisionerror:
...         return
...     s = k, l, g, h
...     return s
...
>>> print(my_kseraks(c))
Введите название: HP
Введите модель: Lazer
Введите цвет: black
Введите значение заполненности катреджа в процентах: 90
```

```
('HP', 'Lazer', 'black', 90)
>>>
```

Задание 6.
Исходные данные:
Продолжить работу над вторым заданиемРеализуйте механизм валидации вводимых пользователем данных. Например, для указания количества принтеров, отправленных на склад, нельзя использовать строковый тип данных.
Подсказка: постарайтесь реализовать в проекте «Склад оргтехники» максимум возможностей, изученных на уроках по ООП.
Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> class Store:
...     def __init__(self, orctechnics):
...         self.orctechnics = orctechnics
...
>>> class Orctechnics(Store):
...     def __init__(self, name, model):
...         self.name = name
...         self.model = model
...
>>> class Skaner(Orctechnics):
...     def __init__(self, number, type):
...         self.number = number
...         self.type = type
...         super().__init__()
...
>>> a = dict(name=str, model=str, number=int, type=str)
>>> def my_skaner(a):
...     try:
...         if name==str:
...             print('Печатать название')
...         else:
...             print('Введены не верные данные')
...         if model==str:
...             print('Печатать модель')
...         else:
...             print('Введены не верные данные')
...         if number==int:
...             print('Печатать число')
...         else:
...             print('Введены не верные данные')
...         if type==str:
...             print('Печатать тип сканера')
...         else:
...             print('Введены не верные данные')
...     except valueerror and zerodivisionerror:
...         return
...
>>>
```

```
>>> print(a)
{'name': <class 'str'>, 'model': <class 'str'>, 'number': <class 'int'>, 'type': <class 'str'>}
>>> class printer(Orctechnics):
...     def __init__(self, paper, size):
...         self.paper = paper
...         self.size = size
...         super().__init__()
...
>>> b = dict(name=str, model=str, paper=str, size=int)
>>> def my_printer(b):
...     try:
...         if number==str:
...             print('Печатать название')
...         else:
...             print('Введены не верные данные')
...         if model==str:
...             print('Печатать модель')
...         else:
...             print('Введены не верные данные')
...         if paper==str:
...             print('Печатать фирму изготовителя бумаги')
...         else:
...             print('Введены не верные данные')
...         if size==int:
...             print('Печатать размер бумаги')
...         else:
...             print('Введены не верные данные')
...     except valueerror and zerodivisionerror:
...         return
...
>>> print(b)
{'name': <class 'str'>, 'model': <class 'str'>, 'paper': <class 'str'>, 'size': <class 'int'>}
>>>
>>> class Kseraks(Orctechnics):
...     def __init__(self, color, refueling):
...         self.color = color
...         self.refueling = refueling
...         super().__init__()
...
>>> c = dict(name=str, model=str, color=str, refueling=int)
>>> def my_kseraks(c):
...     try:
...         if name==str:
...             print('Печатать название')
...         else:
...             print('Введены не верные данные')
...         if model==str:
...             print('Печатать модель')
...         else:
...             print('Введены не верные данные')
...         if color==str:
...             print('Печатать цвет')
```

```
...         else:
...             print('Введены не верные данные')
...         if refueling==int:
...             print('Печатать значения заполнения катреджа в процентах')
...         else:
...             print('Введены не верные данные')
...     except valueerror and zerodivisionerror:
...         return
...
>>> print(c)
{'name': <class 'str'>, 'model': <class 'str'>, 'color': <class 'str'>, 'refueling': <class 'int'>}
>>>
>>>
>>>
```