Lab 9.1 Создать класс CustomButton, у которого есть:

1. метод __init__, принимающий один обязательный аргумент текст кнопки, его необходимо сохранить в атрибут text. И также в метод может поступать произвольное количество ***именованных*** аргументов. Их необходимо сохранять в атрибуты экземпляра под тем же названием;
2. метод config, который принимает произвольное количество именованных атрибутов. Он должен создать атрибут с указанным именем или, если этот атрибут уже присутствовал в экземпляре, изменить его на новое значение;
3. метод click, который должен выполнить следующую строчку - self.command().

   Здесь command является не методом, а атрибутом, который вызывают. В момент выполнения этой строчки может произойти ситуация:

   - атрибут command может отсутствовать у экземпляра и тогда возникнет исключение AttributeError

   Эту ситуацию вам необходимо обработать в блоке try-except и нужно вывести сообщение «Кнопка не настроена», если вызвано исключение.

Пример использования класса CustomButton

```
def func():
   print('Готово')

btn = CustomButton(text="Hello", bd=20, bg='#ffaaaa')
btn.click()  # Кнопка не настроена
btn.config(command=func)
btn.click()  # Готово
```

Lab 9.2 Создать класс Customer, который содержит:

1. метод __init__, принимающий на вход имя пользователя и необязательный аргумент баланс его счета (по умолчанию 0). Эти значения необходимо сохранить в атрибуты name и balance;
2. статический метод check_type , принимающий на вход одно значение. Если оно не является числом (не принадлежит классу int или float) необходимо вызывать исключение TypeError ('Банк работает только с числами'). Метод check_type должен только вызывать исключение в случае неправильного типа, возвращать ничего не должен;
3. метод withdraw, принимающий на вход значение для списания. Необходимо сперва проверить переданное значение на тип при помощи метода check_type. Если исключений не возникло, необходимо проверить что у покупателя достаточно средств на балансе. Если денег хватает, то необходимо уменьшить баланс. Если средств не хватает, нужно вызвать исключение ValueError('Сумма списания превышает баланс');
4. метод deposit , принимающий на вход значение для зачисления на баланс. При помощи метода check_type проверьте, что передано число. Если исключений не возникло, увеличьте значение баланса покупателя на указанную сумму.

Пример использования класса Customer:

```
bob = Customer('Bob Odenkirk')
bob.deposit(200)
print(bob.balance)  # 200
bob.withdraw(150)
print(bob.balance)  # 50
# bob.deposit('hello') # TypeError: Банк работает только с числами
# bob.withdraw(300) # ValueError: Сумма списания превышает баланс
```

Lab 9.3 Создать dataclass: класс Point, который должен хранить два целых атрибута x и y.

На основании класса Point создайте

1. точку с координатами (5, 7) и сохраните ее в переменную point1
2. точку с координатами (-10, 12) и сохраните ее в переменную point2

Выведите сперва point1 и на отдельной строке point2.

Lab 9.4 Создайте датакласс Location.

В нем должны быть описаны следующие атрибуты:

1. name - обязательный, тип строка;
2. longitude - необязательный, вещественный тип, значение по умолчанию 0;
3. latitude - необязательный, вещественный тип, значение по умолчанию 11.5.

Создайте ЭК Location со значениями name='Stonehenge', longitude=51, latitude=1.5 и сохраните его в переменную stonehenge.

Выведите ЭК Location.