СРЕДСТВА ПРОГРАММНОЙ РАЗРАБОТКИ, ВАРИАНТ НА БУКВУ "О"

**Exercise\_3**

Пример программы на Python, которая запрашивает у пользователя строку и выводит ее в обратном порядке с использованием цикла.

|  |
| --- |
| # Запрашиваем у пользователя строку  original\_string = input("Введите строку: ")  # Используем цикл для обратного вывода строки  print("Обратная строка:", end=" ")  for i in range(len(original\_string)-1, -1, -1):  print(original\_string[i], end="")  # Переход на новую строку для лучшего визуального восприятия  print() |

Вы можете запустить этот код в вашей среде выполнения Python, введя строку при запросе. Программа затем выведет введенную строку в обратном порядке.

**Exercise\_6**

Программа, которая создает два списка, заполняет их случайными элементами, и создает словарь, используя в качестве значений элементы первого списка, а в качестве ключей элементы из второго.

|  |
| --- |
| import random  # Создаем два списка со случайными элементами  list1 = [random.randint(1, 10) for \_ in range(5)] # пример: случайные числа от 1 до 10, 5 элементов  list2 = [random.choice("abcdefghij") for \_ in range(5)] # пример: случайные буквы a-j, 5 элементов  # Создаем словарь, используя в качестве ключей элементы из list2 и в качестве значений элементы из list1  dictionary = dict(zip(list2, list1))  # Выводим списки и словарь  print("Список 1:", list1)  print("Список 2:", list2)  print("Словарь:", dictionary) |

Обратите внимание, что приведенные значения в примерах создания списков (random.randint(1, 10) и random.choice("abcdefghij")) могут быть изменены в зависимости от ваших конкретных требований.

**Exercise\_9**

Пример создания класса "Товар" с атрибутами "название" и "цена", а затем создание объектов этого класса и вывод информации о товарах. Этот код создает класс "Товар", и затем создает три объекта этого класса, представляющих разные товары. Информация о каждом товаре выводится на экран.

|  |
| --- |
| class Product:  def \_\_init\_\_(self, name, price):  self.name = name  self.price = price  # Создаем объекты класса "Товар"  product1 = Product("Ноутбук", 1200.0)  product2 = Product("Смартфон", 600.0)  product3 = Product("Наушники", 80.0)  # Выводим информацию о товарах  print("Товар 1:")  print("Название:", product1.name)  print("Цена:", product1.price)  print("\nТовар 2:")  print("Название:", product2.name)  print("Цена:", product2.price)  print("\nТовар 3:")  print("Название:", product3.name)  print("Цена:", product3.price) |

Вы можете адаптировать код и добавить другие атрибуты или методы в класс в зависимости от ваших потребностей.

**Exercise\_15**

Для создания модели "Auto" в Django и проведения миграций, вам нужно выполнить следующие шаги:

Определите модель в файле models.py вашего Django-приложения.

|  |
| --- |
| # models.py  from django.db import models  class Auto(models.Model):  brand = models.CharField(max\_length=50)  model = models.CharField(max\_length=50)  year = models.PositiveIntegerField()  color = models.CharField(max\_length=20)  def \_\_str\_\_(self):  return f"{self.year} {self.brand} {self.model} ({self.color})" |

Создайте миграции для вашей модели с помощью команды makemigrations.

|  |
| --- |
| python manage.py makemigrations |

Примените миграции к базе данных с помощью команды migrate.

|  |
| --- |
| python manage.py migrate |

После выполнения этих шагов, модель "Auto" будет создана в вашей базе данных, и соответствующие таблицы будут созданы с учетом определенных вами полей и их типов. Вы можете использовать эту модель для хранения информации о различных автомобилях в вашем Django-приложении.

**Exercise\_18**

Для создания формы и представления для добавления данных об автомобилях в базу данных в Django, выполните следующие шаги:

Создайте форму в файле forms.py вашего Django-приложения.

|  |
| --- |
| # forms.py  from django import forms  from .models import Auto  class AutoForm(forms.ModelForm):  class Meta:  model = Auto  fields = ['brand', 'model', 'year', 'color'] |

Создайте представление для обработки данных из формы и сохранения их в базе данных в файле views.py.

|  |
| --- |
| # views.py  from django.shortcuts import render, redirect  from .forms import AutoForm  def add\_auto(request):  if request.method == 'POST':  form = AutoForm(request.POST)  if form.is\_valid():  form.save()  return redirect('success') # перенаправление на страницу "success" после успешного сохранения  else:  form = AutoForm()  return render(request, 'add\_auto.html', {'form': form}) |

Создайте HTML-шаблон для отображения формы в файле add\_auto.html.

|  |
| --- |
| <!-- add\_auto.html -->  <!DOCTYPE html>  <html>  <head>  <title>Add Auto</title>  </head>  <body>  <h2>Add Auto</h2>  <form method="post" action="{% url 'add\_auto' %}">  {% csrf\_token %}  {{ form.as\_p }}  <button type="submit">Add Auto</button>  </form>  </body>  </html> |

Обновите файл urls.py вашего приложения, чтобы добавить маршрут для представления.

|  |
| --- |
| # urls.py  from django.urls import path  from .views import add\_auto  urlpatterns = [  path('add\_auto/', add\_auto, name='add\_auto'),  # Добавьте другие маршруты, если необходимо  ] |

Не забудьте подключить приложение в файле settings.py.

|  |
| --- |
| # settings.py  INSTALLED\_APPS = [  # ...  'your\_app\_name', # Замените 'your\_app\_name' на имя вашего приложения  # ...  ] |

Теперь, когда вы посещаете страницу "/add\_auto/", вы увидите форму для добавления данных об автомобиле в базу данных. После заполнения формы и отправки, данные будут сохранены в базе данных. Обратите внимание, что вы также можете настроить маршруты, представления и шаблоны в соответствии с вашими потребностями.