**LISTA DE EXERCICIOS:  
ALUNO: VINICIUS DOMINGOS REIS – 4P**

1. O resultado do print seria “55”, pois a multiplicação de string \* int ele repete o valor da string vezes o numero informado.
2. Maior de idade.  
   Idade = 18;  
   print(“Maior de idade” if idade >= 18 else “Menor de idade”)
3. Lista = [1,2,3,4,5,6,7,8,9,10]  
   for x in lista:  
    if x == 5:  
    continue  
    print(x)
4. Cria uma função chamada saudação  
   Cria um retorno de impressão em tela escrito Olá, e o valor que passa na linha abaixo!  
   Escreve na tela Olá, Maria!
5. Alunos[“Carlos”] = 9
6. São 2 formas de implementar a camada view no Django  
     
   Class Based Views = Função muito útil e fácil de implementar. A desvantagem é em um projeto Django muito grande, há muitas semelhanças nas views. Se todos os objetos de um projeto Django geralmente têm operações CRUD, esse código é repetido várias vezes desnecessariamente e essa foi uma das razões pelas quais as visualizações baseadas em classes e visualizações genéricas foram criadas para resolver esse problema.

Function Based Views = Fornece uma maneira alternativa de implementar a view como objeto Python em vez de funções. Não substitui a CBV.

Em resumo, por exemplo, se estiver trabalhando com lista e puder fazer um ListView ótimo. Agora se estiver fazendo algo mais complexo manipulando vários forms a FBV será mais útil.

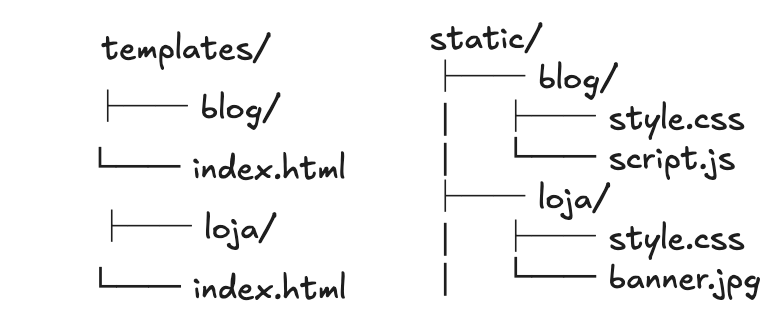
1. Nesse arquivo faz a o roteamento para as urls do projeto.

Path() = Cria o mapeamento das URLs  
Include() = Importa as urls desses APP, para o core do projeto.

1. SECRET\_KEY = Chave interna criada pelo Django para criptografar dados sensíveis, como sessões e tokens.  
   DEBUG = Exibe ou não menagens de erros detalhadas em tela.  
   ALLOWED\_HOSTS = Lista para permitir que domínios/hosts possam acessar o site.  
   INSTALLED\_APPS = Lista os Apps ativos do projeto, senão estiver na lista o Django não reconhece o App.  
   STATIC\_URL = Determina os caminhos dos arquivos estáticos (CSS, JS)
2. Template é usado para gerar paginas dinâmicas HTML.  
   A função render() que carrega o template, insere os dados dentro dele e retorna um HTML completo no navegador.

{{ }} = Exibe valor da variável ou resultado simples no HTML  
{% %} = Usado para instruções lógicos como laco FOR e condição IF.

1. Tem que configurar corretamente o setting.py na linha STATIC\_URL, STATIC\_ROOT e STATICFILES\_DIRS, para o Django entender aonde procurar.  
   Organizar a pasta STATIC dentro do App ou do CORE.
2. Só muda a nomenclatura e a organização de responsabilidade. Torna o desenvolvimento WEB mais limpo, modular e pratico de manutenção.
3. Namespaces = São subpastas com nomes únicos. Organizam os arquivos de cada APP separadamente, permite que diferentes apps tenham arquivos com o mesmo nome sem conflito.

Ex:   


8) O resultado será uma pagina renderizada home.html com um contexto: Bem-vindo ao meu site!  
 Se retirar o dicionário ele não exibe a mensagem, exibe uma tela em branco pois não tem nada.

9) urlpatterns = [ path('blog/', include('blog.urls')), ]

10) Ele exibe o valor da variável mensagem que é = Bem vindo ao site!

11) Tem que passar a data no contexto e utilizar com filtro date no template.  
<p>Data atual: {{ data\_atual|date:"d de F Y" }}</p>

12) from django.shortcuts import render  
 def contato(request):  
 return render(request, 'contato.html', {'mensagem': 'Obrigado pela sua mensagem!'})

13) Criar uma view customizada e um template personalizado.  
from django.shortcuts import render  
 def page\_not\_found(request, exception):  
 return render(request, '404.html', status=404)

14) Configuracao do arquivo setting.py na linha STATIC\_URL, STATIC\_ROOT.  
 Estrutura de arquivos estáticos  
 Uso no templates com {% %}

15) Permite que crie uma pagina de estrutura, e herde ele em outros APPs evitando repetição de código.

16) Usar links nos templates para navegar entre diferentes views.