

Universidad Nacional de Ingeniería

Facultad de Ciencias



PROYECTO FORMATIVO

Integrantes:

- Luigui Jesus Cruz Sandiga
- Martel Balvin Isaac Antonio
- Joaquin Humberto Ramirez Caruajulca

Docente:

- Americo Andres Chulluncuy Reynoso

Introducción

En este proyecto formativo exploramos y comparamos dos lenguajes de programación ampliamente utilizados, Python y C++, así como dos frameworks populares para desarrollo web en Python, Django y Flask. Esta comparación se centra en identificar las fortalezas y debilidades de cada opción para ayudar en la selección adecuada según las necesidades específicas de cada proyecto.

Fundamento Teórico

Python vs. C++

Python y C++ son lenguajes de programación con enfoques y características muy diferentes, cada uno con sus propias ventajas y áreas de aplicación preferidas.

Python

Ventajas:

- Sintaxis sencilla y legible.
- Ampla variedad de bibliotecas y frameworks.
- Ideal para desarrollo rápido y prototipado.

Desventajas:

- Rendimiento relativamente menor en comparación con C++.
- Gestión de memoria menos eficiente.



Figura 1: Comparación Python vs. C++

C++

Ventajas:

- Alto rendimiento y control cercano al hardware.
- Eficiente gestión de memoria.
- Ampliamente utilizado en sistemas embebidos y juegos.

Desventajas:

- Sintaxis más compleja y propensa a errores.
- Curva de aprendizaje más pronunciada.



Figura 2: Comparación Flask vs. Django

Django vs. Flask

Django y Flask son frameworks para desarrollo web en Python, cada uno con características únicas que los hacen adecuados para diferentes tipos de proyectos.

Django

Ventajas:

- Framework completo con muchas funcionalidades integradas.
- Facilita el desarrollo rápido y estructurado de aplicaciones complejas.
- Ideal para proyectos que requieren seguridad y escalabilidad.

Desventajas:

- Puede ser excesivo para proyectos pequeños o simples.
- Curva de aprendizaje más pronunciada que Flask.

Flask

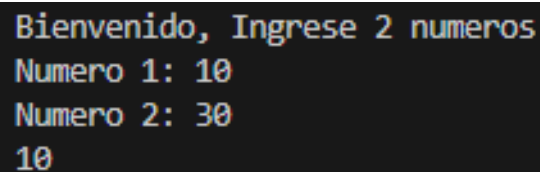
Ventajas:

- Ligero y flexible.
- Fácil de aprender y comenzar a usar.
- Ideal para proyectos pequeños y prototipos rápidos.

Desventajas:

- Requiere más configuración manual para funcionalidades avanzadas.
- Menos integrado que Django en términos de características integradas.

Funciones en Python

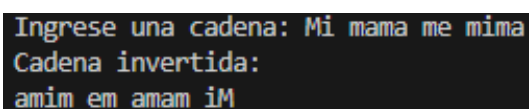


```
Bienvenido, Ingrese 2 numeros
Numero 1: 10
Numero 2: 30
10
```

```
def mcd(numero1, numero2):
    if numero1 == 0:
        return numero2
    return mcd(numero2 % numero1, numero1)
    if __name__ == '__main__':
        print("Bienvenido, Ingrese 2 numeros")

num1 = int(input("Numero 1: "))
num2 = int(input("Numero 2: "))
print(mcd(num1, num2))
```

Cadenas en Python



```
Ingrese una cadena: Mi mama me mima
Cadena invertida:
amim em amam iM
```

```
def invertCad(cad):
    newCad = ""
    for i in cad:
        newCad = i + newCad
    return newCad

if __name__ == '__main__':
    cadena = input("Ingrese una cadena: ")
    print("Cadena invertida:")
    print(invertCad(cadena))
```

Diccionarios en Python

```
--- Agenda Telefónica ---
1. Añadir contacto
2. Buscar contacto
3. Editar contacto
4. Eliminar contacto
5. Mostrar todos los contactos
6. Salir
Seleccione una opción (1-6): 3
Ingrese el nombre del contacto que desea editar: Pepito
Ingrese el nuevo número de teléfono para Pepito: 987633554
Contacto Pepito actualizado con éxito.
```

```
# Definición de la agenda telefónica como un diccionario
agenda = {}
```

```
# Función para mostrar el menú
def mostrar_menu():
    print("\n--- Agenda Telefónica ---")
    print("1. Añadir contacto")
    print("2. Buscar contacto")
    print("3. Editar contacto")
    print("4. Eliminar contacto")
    print("5. Mostrar todos los contactos")
    print("6. Salir")
```

```
# Función para añadir un contacto
def añadir_contacto():
    nombre = input("Ingrese el nombre del contacto: ")
    telefono = input("Ingrese el número de teléfono: ")
    agenda[nombre] = telefono
    print(f"Contacto {nombre} añadido con éxito.")
```

```
# Función para buscar un contacto
```

```
def buscar_contacto():
    nombre = input("Ingrese el nombre del contacto que desea buscar: ")
    if nombre in agenda:
        print(f"El número de teléfono de {nombre} es {agenda[nombre]}.")
    else:
        print(f"El contacto {nombre} no se encuentra en la agenda.")

# Función para editar un contacto
def editar_contacto():
    nombre = input("Ingrese el nombre del contacto que desea editar: ")
    if nombre in agenda:
        telefono = input(f"Ingrese el nuevo número de teléfono para {nombre}: ")
        agenda[nombre] = telefono
        print(f"Contacto {nombre} actualizado con éxito.")
    else:
        print(f"El contacto {nombre} no se encuentra en la agenda.")

# Función para eliminar un contacto
def eliminar_contacto():
    nombre = input("Ingrese el nombre del contacto que desea eliminar: ")
    if nombre in agenda:
        del agenda[nombre]
        print(f"Contacto {nombre} eliminado con éxito.")
    else:
        print(f"El contacto {nombre} no se encuentra en la agenda.")

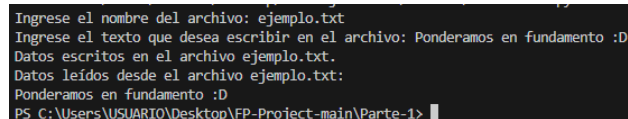
# Función para mostrar todos los contactos
def mostrar_todos_los_contactos():
    if agenda:
        print("\n--- Lista de Contactos ---")
        for nombre, telefono in agenda.items():
            print(f"Nombre: {nombre}, Teléfono: {telefono}")
    else:
        print("La agenda está vacía.")

# Función principal
def main():
    while True:
        mostrar_menu()
        opcion = input("Seleccione una opción (1-6): ")
        if opcion == '1':
            añadir_contacto()
        elif opcion == '2':
            buscar_contacto()
        elif opcion == '3':
            editar_contacto()
        elif opcion == '4':
            eliminar_contacto()
```

```
elif opcion == '5':
    mostrar_todos_los_contactos()
elif opcion == '6':
    print("Saliendo de la agenda. ¡Hasta luego!")
    break
else:
    print("Opción no válida. Inténtelo de nuevo.")

# Ejecución del programa principal
if __name__ == "__main__":
    main()
```

Archivos en Python



```
Ingrese el nombre del archivo: ejemplo.txt
Ingrese el texto que desea escribir en el archivo: Ponderamos en fundamento :D
Datos escritos en el archivo ejemplo.txt.
Datos leídos desde el archivo ejemplo.txt:
Ponderamos en fundamento :D
PS C:\Users\USUARIO\Desktop\FP-Project-main\Parte-1>
```

```
def escribir_datos(nombre_archivo, datos):
    try:
        with open(nombre_archivo, 'w') as archivo:
            archivo.write(datos)
        print(f"Datos escritos en el archivo {nombre_archivo}.")
    except Exception as e:
        print(f"Ocurrió un error al escribir en el archivo: {e}")
    def leer_datos(nombre_archivo):
        try:
            with open(nombre_archivo, 'r') as archivo:
                datos = archivo.read()
            print(f"Datos leídos desde el archivo {nombre_archivo}:")
            print(datos)
            return datos
        except Exception as e:
            print(f"Ocurrió un error al leer el archivo: {e}")
            return None
    # Solicitar al usuario el nombre del archivo y los datos a escribir
    nombre_archivo = input("Ingrese el nombre del archivo: ")
    datos_a_escribir = input("Ingrese el texto que desea escribir en el archivo: ")

    # Escribir datos en el archivo
    escribir_datos(nombre_archivo, datos_a_escribir)

    # Leer datos desde el archivo
    datos_leidos = leer_datos(nombre_archivo)
```

Clases en Python

```
Ingrese el nombre: Pablo
Ingrese la edad: 18
Nombre: Pablo, Edad: 18
```

```
class Persona:
    def __init__(self, nombre, edad): #No es necesario agregar el __init__, pero es
        self.nombre = nombre
        self.edad = edad

    def mostrar_datos(self):
        print(f"Nombre: {self.nombre}, Edad: {self.edad}")

# Solicitar datos al usuario
nombre = input("Ingrese el nombre: ")
edad = int(input("Ingrese la edad: "))

# Crear una instancia de la clase Persona con los datos ingresados
persona1 = Persona(nombre, edad)

# Mostrar los datos
persona1.mostrar_datos()
```

Parte 2

Calculadora en Python

```
Calculadora simple:
1. Suma
2. Resta
3. Multiplicacion
4. Division
5. Salir
Seleccione la operacion que desea realizar (1-5): 3
Ingrese el valor del primer numero: 99
Ingrese el valor del segundo numero: 12
El producto de 99.0 y 12.0 es: 1188.0
```

```
def suma(x, y):
    return x + y

def resta(x, y):
    return x - y
```



```
def producto(x, y):
    return x * y

def division(x, y):
    if y != 0:
        return x / y
    else:
        return "ERROR! No se puede dividir entre 0."

def calculadora():
    while True:
        print("Calculadora simple: ")
        print("1. Suma")
        print("2. Resta")
        print("3. Multiplicacion")
        print("4. Division")
        print("5. Salir")

    op = input("Seleccione la operacion que desea realizar (1-5): ")

    while op not in ['1', '2', '3', '4', '5']:
        print("Entrada no valida. Intente nuevamente.")
        op = input("Seleccione la operacion que desea realizar (1-5): ")

    if op == '5':
        print("Saliendo...")
        break

    num1 = float(input("Ingrese el valor del primer numero: "))
    num2 = float(input("Ingrese el valor del segundo numero: "))

    if op == '1':
        resultado = suma(num1, num2)
        print(f"La suma de {num1} y {num2} es: {resultado}")
    elif op == '2':
        resultado = resta(num1, num2)
        print(f"La diferencia entre {num1} y {num2} es: {resultado}")
    elif op == '3':
        resultado = producto(num1, num2)
        print(f"El producto de {num1} y {num2} es: {resultado}")
    elif op == '4':
        resultado = division(num1, num2)
        print(f"El cociente entre {num1} y {num2} es: {resultado}")

    calculadora()
```

Parte 3: Aplicativo web

En esta sección, se presentará la estructura y funcionamiento de un aplicativo web desarrollado con Django. A continuación, se muestran ejemplos de código y capturas de pantalla del archivo ‘models.py’, así como otros componentes relevantes.

```
models.py x
Modulos > Academica > models.py > ...

4 # DATABASE DE LAS TAREAS
5 class Task(models.Model):
6     description = models.CharField(max_length=255)
7     completed = models.BooleanField(default=False)
8     student = models.ForeignKey(User, on_delete=models.CASCADE)
9
10     def __str__(self):
11         return self.description
12
13 # DATABASES que no se usaron en la funcionalidad del aplicativo, uso didactico
14 class Carrera(models.Model):
15     codigo = models.CharField(max_length=3, primary_key=True)
16     nombre = models.CharField(max_length=50)
17     duracion = models.PositiveSmallIntegerField(default=5)
18     def __str__(self):
19         txt= "{0} (Tiempo Estimado:{1}año(s))"
20         return txt.format(self.nombre,self.duracion)
21
22 class Estudiante(models.Model):
23     dni = models.CharField(max_length=8, primary_key=True)
24     apellidoPaterno = models.CharField(max_length=35)
25     apellidoMaterno = models.CharField(max_length=35)
26     nombres = models.CharField(max_length=35)
27     fechaNacimiento = models.DateField()
28     sexos = [
29         ('F', 'Femenino'),
30         ('M', 'Masculino')
31     ]
32     sexo = models.CharField(max_length=1, choices=sexos, default='F')
33     carrera = models.ForeignKey(Carrera, null=False, blank=False, on_delete=models.CASCADE)
34     vigencia = models.BooleanField(default=True)
35
36     def nombreCompleto(self):
37         txt = "{0} {1}, {2}"
38         return txt.format(self.apellidoPaterno, self.apellidoMaterno, self.nombres)
39
40     def __str__(self):
41         txt = "{0} / Carrera: {1} / {2}"
42         if self.vigencia:
43             estadoEstudiante = "Vigente"
44         else:
45             estadoEstudiante = "De Baja"
46         return txt.format(self.nombreCompleto(), self.carrera, estadoEstudiante)
47
48 class Curso(models.Model):
49     codigo = models.CharField(max_length=6, primary_key=True)
50     nombre = models.CharField(max_length=40)
51     creditos = models.PositiveIntegerField() # Cambié a PositiveIntegerField
```

Figura 3: Parte 1 del archivo models.py

Conclusión

En conclusión, tanto en el ámbito de los lenguajes de programación como en los frameworks para desarrollo web, la elección entre Python y C++, y entre Django y Flask, depende en gran medida de las necesidades específicas del proyecto. Cada opción tiene sus puntos fuertes y áreas de aplicación preferidas, lo que permite a los desarrolladores tomar decisiones informadas para maximizar la eficiencia y la efectividad en el desarrollo de software.

```
class Curso(models.Model):
    codigo = models.CharField(max_length=6, primary_key=True)
    nombre = models.CharField(max_length=40)
    creditos = models.PositiveIntegerField() # Cambié a PositiveIntegerField
    docente = models.CharField(max_length=100)
    def __str__(self):
        txt = "{0}{1}/ Docente: {2}"
        return txt.format(self.nombre, self.codigo, self.docente)

class Matricula(models.Model):
    id = models.AutoField(primary_key=True)
    estudiante = models.ForeignKey(Estudiante, null=False, blank=False, on_delete=models.CASCADE)
    curso = models.ForeignKey(Curso, null=False, blank=False, on_delete=models.CASCADE)
    fechaMatricula = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        txt = "{0} matricula{1} en el curso {2} / Fecha: {3}"
        if self.estudiante.sexo == "F":
            letraSexo = "a"
        else:
            letraSexo = "o"
        fecMat = self.fechaMatricula.strftime("%A %d/%m/%Y %H:%M:%S")
        return txt.format(self.estudiante.nombreCompleto(), letraSexo, self.curso, fecMat)
```

Figura 4: Parte 2 del archivo models.py

```
def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            login(request, user)
            return redirect('index') # Redirige a la lista de tareas después de registrarse
    else:
        form = UserCreationForm()
    return render(request, 'register.html', {'form': form})

def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('index') # Redirige a la lista de tareas después de iniciar sesión
    else:
        form = AuthenticationForm()
    return render(request, 'index.html', {'form': form})

def logout_view(request):
    logout(request)
    return redirect('index')

def formularioContacto(request):
    return render(request, "formularioContacto.html")

def contactar(request):
    if request.method=="POST":
        asunto = request.POST["txtAsunto"]
        mensaje= request.POST["txtMensaje"]+" / Email" + request.POST["txtEmail"]
        email_desde= settings.EMAIL_HOST_USER
        email_para = ["isaac.martel.b@uni.pe"]
        send_mail(asunto,mensaje,email_desde,email_para,fail_silently=False)
        return render(request, "contactoExitoso.html")
    return render(request, "formularioContacto.html")

def index(request):
    if request.user.is_authenticated:
```

Figura 5: Código del archivo views1.py

```
    else:
        return login_view(request)

@login_required
def task_list(request):
    tasks = Task.objects.filter(student=request.user)
    if request.method == 'POST':
        form = TaskForm(request.POST)
        if form.is_valid():
            task = form.save(commit=False)
            task.student = request.user
            task.save()
            return redirect('index')
    else:
        form = TaskForm()
    return render(request, 'index.html', {'tasks': tasks, 'form': form})

@login_required
def update_task(request, task_id):
    task = get_object_or_404(Task, id=task_id, student=request.user)
    if request.method == 'POST':
        form = UpdateTaskForm(request.POST, instance=task)
        if form.is_valid():
            form.save()
            return redirect('index')
    else:
        form = UpdateTaskForm(instance=task)
    # return render(request, 'update_task.html', {'form': form, 'task': task})

@login_required
def edit_task(request, task_id):
    task = get_object_or_404(Task, id=task_id, student=request.user)
    if request.method == 'POST':
        new_description = request.POST.get('new_description')
        if new_description:
            task.description = new_description
            task.save()
            return redirect('index')
    else:
        return redirect('index')

@login_required
def delete_task(request, task_id):
    task = get_object_or_404(Task, id=task_id, student=request.user)
    if request.method == 'POST':
        task.delete()
        return redirect('index')
    # return render(request, 'delete_task.html', {'task': task})
```

Figura 6: Código del archivo views2.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index, name='index'),
6     path('contacto/', views.contactar, name='contactar'),
7     path('admin/', views.index, name='admin'),
8     path('login/', views.login_view, name='login'),
9     path('logout/', views.logout_view, name='logout'),
10    path('tasks/', views.task_list, name='task_list'),
11    path('tasks/update/<int:task_id>', views.update_task, name='update_task'),
12    path('tasks/delete/<int:task_id>', views.delete_task, name='delete_task'),
13    path('register/', views.register, name='register'),
14    path('tasks/edit/<int:task_id>', views.edit_task, name='edit_task'),
15 ]
```

Figura 7: Código del archivo urls.py

My ToDo List

Inicio de Sesión

Nombre de usuario:

Contraseña:

[No tienes una cuenta? Registrate aqui](#)

-
- Contacto
-
- Administradores

Figura 8: Captura de la interfaz web

My ToDo List

Agregar una tarea

Description:

Lista de Tareas

- ☐ rgerg
- ☐ marcos
- [Cerrar Sesión](#)

-
- Contacto
-
- Administradores

Figura 9: Captura 2 de la interfaz web