



INSTITUTO POLITÉCNICO  
NACIONAL



UNIDAD PROFESIONAL INTERDISCIPLINARIA EN  
INGENIERÍA Y TECNOLOGÍAS AVANZADAS

PROGRAMACIÓN AVANZADA 2MV7

---

## Practica 2

---

*Autor:*

Barrios Mendez Jose Alberto

Boleta: 2022640111

*Profesor:*

Cruz Mora Jose Luis

**Ing. Mecatrónica**

6 de marzo de 2024

# Índice

<b>1. Objetivo.</b>	<b>3</b>
<b>2. Introduccion.</b>	<b>3</b>
<b>3. Desarrollo.</b>	<b>3</b>
3.1. Programa 1. . . . .	3
3.2. Programa 2. . . . .	6
<b>4. Ejecucion.</b>	<b>9</b>
4.1. Problema 1. . . . .	9
4.2. Programa 2 . . . . .	10
<b>5. Conclusiones.</b>	<b>12</b>

## 1. Objetivo.

Determinar las clases que conforman un problema y crearlas en un lenguaje de programación

## 2. Introduccion.

En la presente practica llevaremos a cabo el analisis de la programacion orientada a objetos en python, veremos algunos conceptos como lo es la Herencia la cual es fundamental en este tipo de programas.

Realizaremos 2 programas con el uso de clases, el primer programa tiene como finalidad pedir al usuario ciertos datos personales para despues imprimirlos en forma de un string que concatene todos los datos pedidos.

Para el segundo ejercicio vamos a realizar una lista de alumnos de igual forma esta se imprime al final con el mismo metodo de crear un string que concatene todos los datos requeridos

## 3. Desarrollo.

### 3.1. Programa 1.

Para empezar con el desarrollo de esta primera practica es necesario analizar el problema, el cual de forma concreta es:

*.Escriba un programa que pida al usuario el nombre, la edad, la estatura y el numero telefónico de tres personas y lo guarde en una lista de tipo Persona. Para realizarlo, debera definir la clase Persona con las propiedades Nombre, Edad, Estatura y Telefono. Al finalizar su programa, este deber a mostrar en pantalla los datos de cada persona capturada. Para esto deber a crear el metodo to string().*

Bien, para empezar con el desarrollo de este problema podemos iniciar declarando las clases correspondientes, recordando el concepto de clases

**Clase:** se refiere a una plantilla en la cual nosotros vamos a definir las propiedades de un objeto asimilado a la realidad, asi como el comportamiento o tambien llamados metodos

Lo primero que podemos realizar es declara la clase "persona" dicha clase tendra como declaramos la clase persona con los siguientes atributos: Nombre, Edad, Telefono, Estatura. Esto con la finalidad de satisfacer las necesidades que el problema requiere.

A parte de agregar atributos tambien le vamos a asignar un metodo o accion, el cual va a ser lo equivalente a presentarse en la vida real, debido a que imprimira en pantalla todos los atributos que se agreguen una vez la instancia sea creada. Se puede visualizar mejor en la parte inferior

```
class Persona:
    def __init__(self,nombre,edad,estatura,telefono) -> None:
        self.nombre=nombre
        self.edad=edad
        self.estatura=estatura
        self.telefono=telefono

    def __str__(self) -> str:

        return f"!Hola!, Mi nombre es {self.nombre}, tengo {self.edad} años, naci en {-self.edad+2024}\nmido" \
        f"{self.estatura} metros y ni numero de telefono es {self.telefono} !Saludos!"
```

Figura 1: Creacion de la clase persona con sus respectivos atributos

En la figura 1 se puede observar la creacion de la clase persona con sus respectivos atributos y su metodo llamado str el cual nos devolvera el texto similar al que es requerido en el problema

En este problema se requiere que el usuario es el que ingrese los datos de las 3 personas. Esto supone un reto para el programa, debido a que los datos que ingrese deben estar dentro de los rangos de una persona normal, para evitar que ingrese datos incorrectos o diferentes a los solicitados se generaron 4 funciones con el fin de verificar que el usuario al menos ingrese datos en campos validos, a continuacion presentamos las 4 funciones generadas

1. Funcion pedir nombre Esta funcion se encarga de recibir un nombre como argumento y analizar si dicho nombre contiene todos sus caracteres de tipo alfabetico, con ayuda de la funcion `.isalpha()`, la cual se encarga de analizar la cadena y devolver True o False en cada caso, esto esta dentro de un ciclo while, el cual se encarga de ejecutarse hasta que el usuario ingrese un nombre aceptable para el programa

```
22 def pedir_nombre():
23     while True:
24         nombre=(input("Ingresa el nombre: "))
25
26         if (nombre.isalpha()):
27             break
28         else:
29             print("Ingresa un nombre valido!!")
30     return nombre
```

Figura 2: Funcion para ingresar un nombre valido

2. Funcion pedir edad Para esta funcion vamos a considerar un rango de edad que este entre los 0 y 100 años, en caso de que el usuario ingrese una edad que no este en ese rango el programa volvera a pedir al usuario que ingrese la edad

```

34 def pedir_edad():
35     while True:
36         try:
37             edad=int(input("ingrese su edad: "))
38             if (edad<=0 or edad>100):
39                 print("Ingrese una edad valida: ")
40             else:
41                 break
42         except ValueError:
43             print("Ingrese una edad valida ")
44
45     return edad

```

Figura 3: Funcion para ingresar una edad valido

3. Funcion pedir estatura Para esta funcion lo que hacemos es etablecer un limite que este entre 0 y 2.15 metros, alguno otro valor fuera de este rango volvera a preguntar la estatura

```

46
47 def pedir_estatura():
48     while True:
49         try:
50             estatura=float(input("Ingrese su estatura en metros : "))
51             if (estatura<=0 or estatura >2.15):
52                 print("Estatura no valida: ")
53             else:
54                 break
55         except ValueError:
56             print("Estatura invalida: ")
57
58     return estatura

```

Figura 4: Funcion para una estatura aceptable

4. Funcion pedir telefono Para esta funcion vamos a considerar un numero valido al menos en mexico con 10 digitos, con el uso de la funcion len, la cual le pasamos como argumento una cadena donde esta guardado el numero ingresado por el usuario, esta funcion analiza el numero de caracteres, en el caso donde se ingrese un numero con diferentes características la sentencia se va a seguir ejecutando hasta el ingreso de uno valido

Listing 1: Funcion para pedir un numero de telefono

```

1 def pedir_telefono():
2     while True:
3         try:
4             numero=int(input("Ingrese su numero de telefono (10 digitos):

```

```

5         if (numero<0 or (len(str(numero))<10)):
6             print("Ingresa un numero valido")
7         else :
8             break
9     except ValueError:
10        print("Ingrese un numero valido: ")
11    return numero

```

Ahora que hemos terminado de hacer las funciones que nos ayudaran a pedir los datos requeridos vamos con la parte de carga de codigo e impresion de valores:

Primero iniciamos con la creacion de una tupla llamada personas, despues iteramos en un ciclo for de 1 hasta 3, donde conforme avanza i, en pantalla se mostrara el numero de persona en la que estaremos ingresando los datos, recalquemos que la variable persona es un objeto de tipo Persona, y aqui es donde añadimos las funciones que definimos anteriormente, las funciones iran en los argumentos del objeto persona. posteriormente la variable persona es añadida a la tupla.

Por ultimo usamos un ciclo for para mostrar en pantalla las presentaciones de cada persona en consola

Listing 2: Carga de valores e impresion de resultados

```

1 personas=[]
2
3 for i in range(3):
4     print(f"-----PERSONA {i+1}")
5     persona=Persona(pedir_nombre(),pedir_edad(),pedir_estatura(), pedir_telefono())
6     personas.append(persona)
7     os.system("cls")
8
9 os.system("cls")
10 for i in range(3):
11     print(personas[i])

```

### 3.2. Programa 2.

Ahora analisemos el programa numero 2, el cual consiste en solicitar al usuario El nombre y datos del profesor, asi como de n alumnos, para posteriormente imprimirlos en una tipo lista. La practica sugiere usar 3 clases las cuales son Alumno, Profesor, Lista, para esta practica yo lo desrrolle de otra forma ya que se me hizo mas factible desarrollarlo con las siguientes clases

Listing 3: Clases utilizadas en esta practica

```

1 class Persona:
2     def __init__(self,nombre,ap,am,fecha_n) -> None:
3         self.nombre=nombre
4         self.ap=ap
5         self.am=am
6         self.fecha_n=fecha_n

```

```

7
8 class Alumno(Persona):
9     def __init__(self, nombre,ap,am,fecha_n,boleta,grupo,carrera,correo) -> None:
10         super().__init__(nombre,ap,am,fecha_n)
11
12         self.boleta=boleta
13         self.grupo=grupo
14         self.carrera=carrera
15         self.correo=correo
16
17     def __str__(self) -> str:
18         return f"{self.ap} {self.am},{self.nombre}--{self.boleta}--{self.fecha_n}"
19         f"{self.carrera}--{self.grupo}--{self.correo}"
20
21 class Profesor(Persona):
22     def __init__(self,nombre,no_employado) -> None:
23         super().__init__(nombre,ap=None,am=None,fecha_n=None)
24         self.no_employado=no_employado
25
26     def __str__(self) -> str:
27         return f"\n\nProfesor:  {self.nombre}\nNo. Empleados: {self.no_employado}"

```

En la listeng 3, observamos las clases usadas en este programa se basan en una clase principal la cual es Persona, esta clase tiene los atributos de nombre,apellidos, y fecha de nacimiento las cuales son heredadas a las clases hijas de Alumno y Profesor, ambas clases agregan sus propios atributos, como profesor agrega su numero de empleado, y alumnos agrega los datos de carrera y algunos de contacto, tambien agregan el metodo str, con la finalidad de devolver una cadena tipo string que muestre todos los atributos proporcionados por el usuario

Ademas agregamos funciones similares a las del problema 1, solo las mencionaremos y daremos una breve explicacion de cada funcion para no alargar mucho el texto con elementos muy similares

```
33
34 def pedirnombre(): ...
43
44 def pedir_app_p(): ...
53
54 def pedir_app_m(): ...
63
64 def fecha_nacimiento(): ...
73
74 def pedir_boleta(): ...
82
83 def pedir_grupo(): ...
86
87 def pedir_carrera(): ...
90
91 def pedir_correo(): ...
99
100 def pedir_no_employado(): ...
108
109
```

Figura 5: Funciones usadas en el programa 2

1. Pedir nombre es similar al anterior, ahora puede admitir 2 nombres y de igual forma analizarlos para comprobar que tenga caracteres validos
2. pedir apellido paterno: analiza la entrada y pide un apellido valido, de igual forma para el apellido materno
3. fecha de nacimiento: con ayuda de la libreria datetime, podemos verificar si la entrada de la fecha es valida o invalida y volverla a solicitar
4. Boleta: admite solo caracteres enteros
5. grupo: para esta funcion puede ser una mezcla de caracteres alfanumericos, buscando una respueste a grupos similares en UPIITA
6. Carrera: Permite el ingreso de tipo string
7. correo: para validar esta entrada nos respaldamos en el caracter @ para darla como valida
8. Numero de empleado: unicamente admite entrada de tipo numerico

Para llevar a cabo el almacenamiento de datos, vamos a necesitar una tupla, y como el ejemplo anterior con un ciclo while vamos a salir del bucle cuando ingresemos n, podemos observarlo en la siguiente figura



```

118 while True:
119     os.system("cls")
120     print(f"-----ALUMNO {i+1}-----")
121     alumno = Alumno(pedir_nombre(), pedir_app_p(), pedir_app_m(), fecha_nacimiento(), pedir_boleta(), pedir_grupo(), pedir_carrera(), pedir_correo())
122     lista.append(alumno)
123     respuesta = input("¿Desea agregar otro alumno a la lista? (s/n)")
124     i += 1
125     if respuesta.lower() == 'n':
126         break
127     elif respuesta.lower() == 's':
128         continue
129     else:
130         print("Respuesta no válida. Por favor, responda 's' para sí o 'n' para no.")

```

Figura 6: Ciclo para guardar los objetos en una tupla

Finalmente se imprimen la lista con un ciclo for y con ayuda del metodo str que se asigna en cada clase (alumno y profesor)

Listing 4: Imprime la lista en consola

```

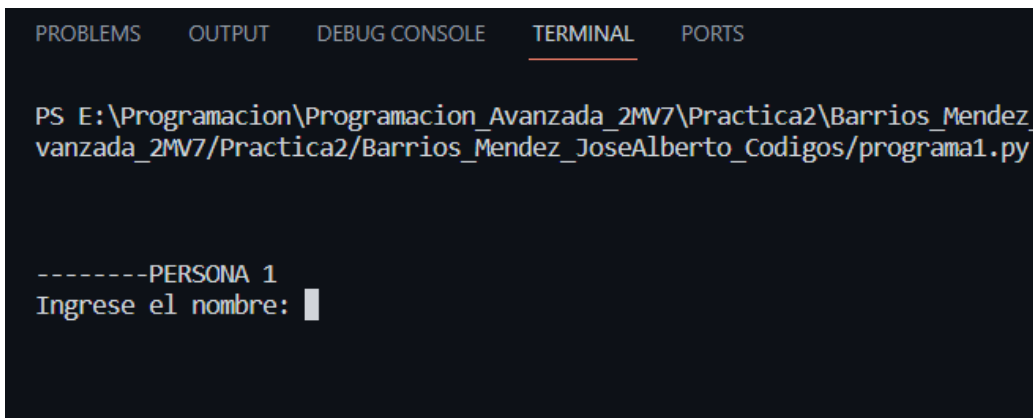
1 for i in range(len(lista)):
2     print(lista[i])
3     print("-"*50)
4
5 print(f"Total de alumnos inscritos: {len(lista)}")

```

## 4. Ejecucion.

### 4.1. Problema 1.

Ahora vamos a ejecutar el primer programa y veamos que nos imprime en pantalla:



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Programacion\Programacion_Avanzada_2MV7\Practica2\Barrios_Mendez_vanzada_2MV7\Practica2\Barrios_Mendez_JoseAlberto_Codigos\programa1.py

-----PERSONA 1
Ingrese el nombre: 

```

Figura 7: Ejecucion del programa 1

vamos a intentar ingresar algunos nombres con otros caracteres para ver como responde el programa ante estas entradas

```
PS E:\Programacion\Programacion_Avanzada_2MV7\Practica2\Barrios_Mendez_vanzada_2MV7\Practica2\Barrios_Mendez_JoseAlberto_Codigos\programa1.py

-----PERSONA 1
Ingrese el nombre: da3d
Ingresa un nombre valido!!
Ingrese el nombre: █
```

Figura 8: Ejecucion del programa 1

En la figura 7, podemos observar que al ingresar un nombre valido, nos pregunta la edad, de la misma forma testeamos algunos valores que no estan dentro del rango anteriormente definidos, de igual forma realizamos lo mismo para la altura y para el numero de telefono, al ingresar al final el numero de telefono correcto nos arrojava a la segunda persona, y despues a la tercera, llenando los datos de forma aleatoria observemos el resultado:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

!Hola¡, Mi nombre es Alberto, tengo 20 años, naci en 2004
mido1.7 metros y ni numero de telefono es 5560930373 !Saludos¡
!Hola¡, Mi nombre es Juan, tengo 23 años, naci en 2001
mido2.0 metros y ni numero de telefono es 5560830839 !Saludos¡
!Hola¡, Mi nombre es Luis, tengo 21 años, naci en 2003
mido1.62 metros y ni numero de telefono es 5560920382 !Saludos¡
PS E:\Programacion\Programacion_Avanzada_2MV7\Practica2\Barrios_Mendez_JoseAlberto_Codigos> █
```

Figura 9: Resultado final del programa 1

## 4.2. Programa 2

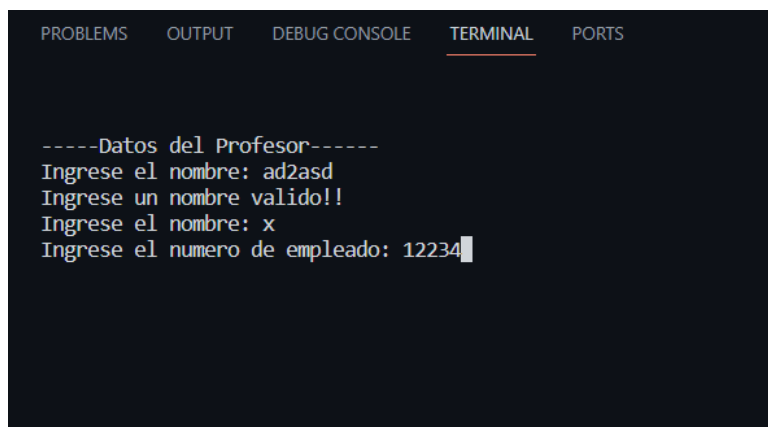
Veamos el resultado al ejecutar el programa

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

-----Datos del Profesor-----
Ingrese el nombre: █
```

Figura 10: Ejecucion del programa 2

Veamos que nos aparece la instruccion para ingresar el nombre del profesor, posteriormente tambien pedira el numero de empleado



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

-----Datos del Profesor-----
Ingrese el nombre: ad2asd
Ingrese un nombre valido!!
Ingrese el nombre: x
Ingrese el numero de empleado: 12234
```

Figura 11: Ingreso de datos del profesor

Una vez que ingresemos los datos del profesor nos mandara a ingresar los datos de los alumnos, a continuacion presentamos un caso, ademas de testear las respuestas para probar el codigo:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Ingrese el nombre: Alb12e
Ingrese un nombre valido!!
Ingrese el nombre: Alberto
Ingrese el Apellido PATERNO: 3r
Ingrese un apellido valido!!
Ingrese el Apellido PATERNO: Barrios
Ingrese el Apellido MATERNO: ,
Ingrese un apellido valido!!
Ingrese el Apellido MATERNO: Mendez
Ingrese la fecha de nacimiento (DD/MM/AAAA): 122/23/2021
Formato de fecha incorrecto. Intente de nuevo.
Ingrese la fecha de nacimiento (DD/MM/AAAA): 15/11/2003
Ingrese el numero de boleta: 2121322132
Ingrese su grupo: 2MV7
Ingrese su carrera: Mecatronica
Ingrese su correo electronico: alsda
Ingrese un correo electronico valido
Ingrese su correo electronico: alberto@gmail.com
```

Figura 12: Ingreso de datos de un estudiante, variando las respuestas

Para mostrar el resultado final, consideramos 2 alumnos ingresados par poder verificar que el programa funciona correctamente.

```
Profesor: x
No. Empleado: 12234
-----
Lista de alumnos
-----
Barrios Mendez,Alberto --2121322132--2003-11-15 00:00:00
Mecatronica--2M7--alberto@gmail.com
-----
hernandez gonzales,Juan--233123--2002-02-11 00:00:00
MECATRONICA--2M7--juan@gmail.com
-----
Total de alumnos inscritos: 2
PS E:\Programacion\Programacion_Avanzada_2M7\Practica2\Barrios_Mendez_JoseAlberto_Codigos> █
```

Figura 13: resultado final del programa 2

## 5. Conclusiones.

Al concluir esta practica donde tratamos algunos temas sobre Programacion orientada a objetos me doy cuenta que es muy eficiente este tipo de programacion ya que nos permite crear bastantes objetos a partir de varias clases con ayuda de la herencia, ademas de su facil manejo del lenguaje lo que lo convierte en una herramienta muy util para nuestro perfil como futuros ingenieros mecatronicos