



2A. EJERCICIO. FUNCIONES Y ESTRUCTURAS DE DATOS

Unidad 2. Funciones y estructuras de datos

Datos

Nombre: Keith Alexis Gutiérrez Ibarra

Docente: Silvia Guadalupe Victorio Aguilar

Fecha de elaboración: 26/09/2025

Introducción

En este trabajo se desarrollarán dos ejercicios: un sistema que calcula la edad según la fecha de nacimiento y un sistema de evaluación de calificaciones. El primero es un formulario para los agentes que permite calcular la edad a partir de la fecha de nacimiento, lo cual es información importante para cotizar una póliza, y el segundo es un sistema de evaluación de alumnos donde se puede cargar una lista de materias y, a partir de estas, generar un perfil de cada alumno que muestre varios datos del estudiante y sus calificaciones, con la condición de marcar si están aprobados o reprobados.

El propósito de estos ejercicios es desarrollar habilidades para la solución de problemas básicos mediante el uso de algoritmos, estructuras de datos y funciones en Python.

Desarrollo

Ejercicio 1

a) Análisis del problema: Para el primer ejercicio se nos piden las siguientes condiciones que debe tener el algoritmo

- La captura de la fecha de nacimiento debe ser en una sola cadena de texto con formato DD-MM-AAAA.
- Validar que sea una fecha existente, si supera los 31 días o menor a 1 día de un mes, se deberá marcar como inválido.
- El año de nacimiento debe ser mayor de 1900.
- Validar si el cliente ya cumplió años en la fecha de captura o aún no.
- Validar que no se puede dejar información en blanco o campos vacíos.

b) Diseño del algoritmo

- Se agrego la librería datetime para el manejo de fechas
- Se crea un diccionario llamado perfil para almacenar la información del cliente
- Se usa un sistema de bucles para solicitar nombre y revisar que este no tenga espacios vacíos

- Se crea una función llamada `calculo_edad` para la validación de fechas, calculo de la edad, cumplimiento de condiciones estipuladas en el problema y arroja un estado de cumpleaños
- Creación de un bucle para solicitar la fecha de nacimiento y hacer la llamada a la función `calculo_edad`
- Una vez correcta todas las condiciones se devuelven los datos del diccionario

c) Código en Python

```
Ejercicio 1.py X Ejercicio 2.py
Ejercicio 2A > Ejercicio 1.py > ...
1  # Esta libreria permite el uso de fechas
2  import datetime
3  # Se crea un diccionario para almacenar toda la informacion del cliente
4  perfil = {"nombre": "", "nacimiento": "", "edad": None, "estado_cumpleanos": ""}
5
6  # Se inicia el sistema
7  print("Formulario de Seguro")
8  print("\nCompleta los siguientes datos:")
9  # Bucle para evitar los campos vacios en la introduccion del nombre
10 while True:
11     # Se solicita el nombre
12     perfil["nombre"] = input("Nombre: ")
13     # Condicion que permite romper el bucle cuando se cumple
14     if perfil["nombre"].strip():
15         break
16     # Condicion cuando el espacio se queda vacio
17     else:
18         print("El campo no puede estar vacio")
19 # Se crea un función llamada calculo_edad que permite el calculo de la edad
20 # segun la fecha de nacimiento y valida la fecha
21 def calculo_edad():
22     try:
23         # Se llama a la variable global
24         global fecha_nacimiento
25         # Se convierte del tipo string a tipo date la fecha
26         fecha_nacimiento = datetime.datetime.strptime(fecha_nacimiento, "%d-%m-%Y").date()
27         # Condicion que comprueba que el año de nacimiento debe ser mayor de 1900.
28         if fecha_nacimiento.year < 1900:
29             #Se retorna inavildo como condicion para el while de fecha de nacimiento
30             return "invalido"
31         # Se obtiene la fecha actual
32         hoy = datetime.date.today()
```

```

33     # Se crean tupla para facilitar la comparacion de las fechas
34     nacimiento_tupla = (fecha_nacimiento.month, fecha_nacimiento.day)
35     hoy_tupla = (hoy.month, hoy.day)
36     # Operacion para calcular las fechas con ayuda de las tuplas
37     edad = hoy.year - fecha_nacimiento.year - ((hoy_tupla) < (nacimiento_tupla))
38     # Se usan varias condiciones comparando las tupla para verificar el estado
39     if hoy_tupla == nacimiento_tupla:
40         estado = "Hoy es el cumpleaños del cliente"
41     elif hoy_tupla > nacimiento_tupla:
42         estado = "El cliente ya cumplio años"
43     else:
44         estado= "El cliente aun no ha cumplido años"
45     # Se actualiza el diccionario con los datos calculados
46     perfil["edad"] = edad
47     perfil["estado_cumpleanos"] = estado
48     perfil["nacimiento"] = fecha_nacimiento
49     #Se retorna true como condicion para el while de fecha de nacimiento
50     return True
51
52     except ValueError:
53         print("Formato de fecha inválido o no existe")
54         #Se retorna none como condicion para el while de fecha de nacimiento
55         return None
56 # Bucle para solicitar y validar fecha
57 while True:
58     # Se solicita la fecha de nacimiento al usuario
59     fecha_nacimiento = input("Fecha de nacimiento (formato DD-MM-AAAA):")
60     # Se llama a la funcion calculo_edad
61     validacion = calculo_edad()
62     # Condiciones que evaluan el resultado de la funcion
63     if validacion == "invalido":
64
65         print("El año de nacimiento debe ser mayor de 1900")
66         break
67     elif validacion is not None:
68         print("Fecha de nacimiento:")
69         print(perfil["nacimiento"])
70         print("edad:")
71         print(perfil["edad"])
72         print("Estado del cumpleaños:")
73         print(perfil["estado_cumpleanos"])
74         break
75     else:
76         # Si la funcion devolvio None el bucle continuara, pidiendo la fecha de nuevo.
77         print("Por favor, intentalo de nuevo")

```

d) Ejecución y pruebas

Ejemplos del funcionamiento del código en diversos escenarios:

- Nombre vacío:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Formulario de Seguro

Completa los siguientes datos:
Nombre:
El campo no puede estar vacío
Nombre: 
```

- Fecha con formato incorrecto:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Formulario de Seguro

Completa los siguientes datos:
Nombre:
El campo no puede estar vacío
Nombre: Alex
Fecha de nacimiento (formato DD-MM-AAAA):01/11/2001
Formato de fecha inválido o no existe
Por favor, intentalo de nuevo
Fecha de nacimiento (formato DD-MM-AAAA): 
```

- Fecha vacía:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Formulario de Seguro

Completa los siguientes datos:
Nombre:
El campo no puede estar vacío
Nombre: Alex
Fecha de nacimiento (formato DD-MM-AAAA):01/11/2001
Formato de fecha inválido o no existe
Por favor, intentalo de nuevo
Fecha de nacimiento (formato DD-MM-AAAA):
Formato de fecha inválido o no existe
Por favor, intentalo de nuevo
Fecha de nacimiento (formato DD-MM-AAAA): 
```

- Año menor a 1900:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Formulario de Seguro

Completa los siguientes datos:
Nombre:
El campo no puede estar vacío
Nombre: Alex
Fecha de nacimiento (formato DD-MM-AAAA):01/11/2001
Formato de fecha inválido o no existe
Por favor, intentalo de nuevo
Fecha de nacimiento (formato DD-MM-AAAA):
Formato de fecha inválido o no existe
Por favor, intentalo de nuevo
Fecha de nacimiento (formato DD-MM-AAAA):01-11-1899
El año de nacimiento debe ser mayor de 1900
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>

```

- Estado de cumpleaños posterior al día actual

```

Formulario de Seguro

Completa los siguientes datos:
Nombre: Alex
Fecha de nacimiento (formato DD-MM-AAAA):01-11-2001
Fecha de nacimiento:
2001-11-01
edad:
23
Estado del cumpleaños:
El cliente aun no ha cumplido años
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>

```

- Estado de cumpleaños al día actual

```

Formulario de Seguro

Completa los siguientes datos:
Nombre: Alex
Fecha de nacimiento (formato DD-MM-AAAA):28-09-2001
Fecha de nacimiento:
2001-09-28
edad:
24
Estado del cumpleaños:
Hoy es el cumpleaños del cliente

```

- Estado de cumpleaños anterior al día actual

```
Formulario de Seguro

Completa los siguientes datos:
Nombre: Alex
Fecha de nacimiento (formato DD-MM-AAAA):01-01-2001
Fecha de nacimiento:
2001-01-01
edad:
24
Estado del cumpleaños:
El cliente ya cumple años
```

Ejercicio 2

a) Análisis del problema: Para el primer ejercicio se nos piden las siguientes condiciones que debe tener el algoritmo

- Si uno de los campos (usuario y contraseña) está vacío deberá mostrar un error de autenticación. Deberá pedir el número de alumnos y materias, pedir el nombre del alumno, y su matrícula.
- El sistema debe evaluar, como condición de evaluación, que si es una calificación mayor a 6, es aprobado y de lo contrario, reprobado.
- Validar que no se puede dejar información en blanco o campos vacíos.

b) Diseño del algoritmo

- Se crea una lista llamada lista estudiantes para almacenar a los estudiantes
- Se crea un diccionario llamado materias para almacenar materias y calificaciones
- Iniciamos el sistema con un bucle para obtener el nombre y revisar que este campo no se encuentre vacío
- Desarrollamos una función llamada calculo_edad que realiza el calculo de la edad a partir de la fecha de nacimiento, valida la fecha, comprueba el año de nacimiento y arroja un estado de la condición de cumpleaños
- Se crea un bucle para solicitar la fecha de nacimiento y revisar que el campo no se encuentra vacío
- Dentro del bucle se encuentra una serie de condiciones que devuelven un resultado según el retorno que de la función

c) Implementación en Python

```
Ejercicio 2A > Ejercicio 2.py > ...
1  # Se crea una lista para almacenar todos los estudiantes
2  lista_estudiantes = []
3  # Se crea un diccionario para almacenar materias y calificaciones
4  materias = {}
5  # Se crea un función llamada promedio que permite calcular el promedio y darle un estado
6  def promedio(estudiante):
7      # Se obtiene las calificaciones
8      calificaciones = estudiante["calificaciones"].values()
9      # Calculo de promedio
10     promedio_final = sum(calificaciones) / len(calificaciones)
11     # Condicion de aprobado o reprobado
12     if promedio_final >= 6:
13         estado = "Aprobado"
14     else:
15         estado = "Reprobado"
16     return promedio_final, estado
17
18 # Se inicia el sistema
19 print("Sistema de evaluacion")
20 print("\nRegistro de Materias:")
21 # Bucle para registro de materias
22 while True:
23     # Se solicita el nombre de la materia
24     print("Para dejar de cargar materias introduce exit")
25     materia= input("Nombre de la materia: ")
26     # Condicion para salir del bucle
27     if materia.lower() == 'exit':
28         if not materias:
29             print("Debe registrar al menos una materia antes de salir.")
30             continue
31         else:
32             break
```

```
Ejercicio 2A > Ejercicio 2.py > ...
33     # Condicion cuando el espacio se queda vacio
34     if not materia.strip():
35         print("No puede estar vacion el campo")
36         continue
37     # Se agrega la materia al diccionario
38     materias[materia] = None
39     print("Se registro correctamente")
40 # Registro de alumnos
41 print("\nRegistro de alumnos")
42 # Cantidad de alumnos a registrar
43 no_alumnos = int(input("Numero de alumnos: "))
44 # Bucle para registro de alumnos segun la condicion anterior
45 for i in range(no_alumnos):
46     print(f"\nRegistro del alumno {i+1} de {no_alumnos}")
47     # Se solicita el nombre y se verifica que no este vacia
48     while True:
49         nombre= input("Nombre del alumno: ")
50         if nombre.strip():
51             break
52         print("No puede estar vacion el campo")
53     # Se solicita la matricula y se verifica que no este vacia
54     while True:
55         matricula= input("Matricula del alumno: ")
56         if matricula.strip():
57             break
58         print("No puede estar vacion el campo")
59     # Se crea un diccionario para los datos del estudiante
60     estudiante = {
61         "nombre": nombre,
62         "matricula": matricula,
63         "calificaciones": materias.copy()
64     }
```

```

65     # Calificaciones
66     print(f"Introduce las calificaciones para {nombre}")
67     # Bucle para insertar las calificaciones de cada materia
68     for materia in estudiante["calificaciones"]:
69         while True:
70             calificacion = input(f"Calificacion para {materia}: ")
71             # Condicion cuando el espacio no se queda vacio
72             if calificacion.strip():
73                 # Convierte el tipo de dato de string a int
74                 calificacion = int(calificacion)
75                 # Guarda la calificacion en el diccionario del alumno
76                 estudiante["calificaciones"][materia] = calificacion
77                 break
78             # Condicion cuando el espacio se queda vacio
79             else:
80                 print("No puede estar vacio el campo")
81     # Se agrega el estudiante a la lista general
82     lista_estudiantes.append(estudiante)
83     print("Se registro correctamente")
84 # Resultados
85 print("\nREGISTRO FINAL DEL SISTEMA")
86 # condicion que verifica si se registró al menos un estudiante
87 if not lista_estudiantes:
88     print("No se registró ningún alumno.")
89 else:
90     # Muestra la indormacion de cada estudiante
91     for estudiante in lista_estudiantes:
92         print(f"\nAlumno: {estudiante['nombre']} | Matrícula: {estudiante['matricula']}")
93         print("Materias y Calificaciones:")
94         for materia, calificacion in estudiante['calificaciones'].items():
95             print(f"  {materia}: {calificacion}")
96         # Calcula el promedio y da el estado
97
98         promedio_final, estado = promedio(estudiante)
99         print(f"Promedio: {promedio_final:.2f} | Estado: {estado}")

```

d) Ejecución y pruebas

Capturas de pantalla donde se muestre:

- Salir sin materias registradas:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Sistema de evaluacion

Registro de Materias:
Para dejar de cargar materias introduce exit
Nombre de la materia: exit
Debe registrar al menos una materia antes de salir.
Para dejar de cargar materias introduce exit
Nombre de la materia:

```

- Materia vacía:

```
Se registro correctamente
Para dejar de cargar materias introduce exit
Nombre de la materia:
No puede estar vacio el campo
Para dejar de cargar materias introduce exit
Nombre de la materia: 
```

- Registro exitoso de materias:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Sistema de evaluacion

Registro de Materias:
Para dejar de cargar materias introduce exit
Nombre de la materia: Español
Se registro correctamente
Para dejar de cargar materias introduce exit
Nombre de la materia: 
```

- Numero de estudiante igual a 0

```
Registro de alumnos
Numero de alumnos: 0

REGISTRO FINAL DEL SISTEMA
No se registró ningún alumno.
PS C:\Users\keith\OneDrive\Documentos\Pyrhon> 
```

- Nombre vacío:

```
Registro de alumnos
Numero de alumnos: 2

Registro del alumno 1 de 2
Nombre del alumno:
No puede estar vacio el campo
Nombre del alumno: 
```

- Matricula vacía:

```
Registro del alumno 1 de 2
Nombre del alumno:
No puede estar vacio el campo
Nombre del alumno: Juan
Matricula del alumno:
No puede estar vacio el campo
Matricula del alumno: 
```

- Calificaciones vacías:

```
Nombre del alumno: Juan
Matricula del alumno:
No puede estar vacio el campo
Matricula del alumno: 1234
Introduce las calificaciones para Juan
Calificacion para Español:
No puede estar vacio el campo
Calificacion para Español: █
```

- Registro exitoso de estudiante:

```
Registro del alumno 1 de 2
Nombre del alumno:
No puede estar vacio el campo
Nombre del alumno: Juan
Matricula del alumno:
No puede estar vacio el campo
Matricula del alumno: 1234
Introduce las calificaciones para Juan
Calificacion para Español:
No puede estar vacio el campo
Calificacion para Español: 7
Calificacion para Ingles: 8
Calificacion para Matematicas: 9
Se registro correctamente
```

- Promedio mayor o igual a 6:

```
REGISTRO FINAL DEL SISTEMA

Alumno: Alex | Matrícula: 1234
Materias y Calificaciones:
  Ingles: 7
  Español: 9
  Historia: 9
Promedio: 8.33 | Estado: Aprobado
```

- Promedio menor a 6:

```
Alumno: Juan | Matrícula: 3
Materias y Calificaciones:
  Ingles: 3
  Español: 4
  Historia: 4
Promedio: 3.67 | Estado: Reprobado
PS C:\Users\keith\OneDrive\Documentos\Python> █
```

Conclusiones

En este trabajo se desarrollaron dos sistemas en Python un formulario para calcular la edad a partir de la fecha de nacimiento y un sistema de evaluación de calificaciones, el primero permitió aplicar estructuras de control, validaciones y funciones para garantizar que los datos ingresados fueran correctos, además de calcular la edad y determinar el estado del cumpleaños, el segundo permite registrar materias, alumnos y calificaciones, calculando así los promedios y establecer un estado de aprobado o reprobado, esto aplicando estructuras de datos como listas y diccionarios junto con funciones para organizar la información.

Estos ejercicios permiten ver la importancia de las estructuras de datos como listas, diccionarios y tuplas para mejorar el orden y funcionamiento del código, también con el uso de funciones que ayudan a evitar la replicación de código varias veces, optimizando el código de una mejor manera, además estos ejercicios ayudan a consolidar conceptos que vimos en la unidad acerca de la estructura de datos y funciones