El futuro digital es de todos

MinTIC

CRUD y Conversión de colecciones de datos

OPERADO POR:

Universidad Tecnológica de Pereira

Mision TIC 2022

RUTA DE APRENDIZAJE 1

En la clase anterior, revisamos como crear una aplicación CRUD básica, trabajando con toda la información desde consola.

Recordemos que para lograr que en consola la aplicación se visualice correctamente debemos realizar la impresión de cada línea con mucho cuidado:

```
datosActualizados={'titulo':'', 'descripcion':'', 'estado':'', 'fecha inicio':'', 'fecha finalizacion':''}
while True:
    print('Indique la accion que desea realizar: ')
    print('Consultar: 1')
    print('Actualizar: 2')
    print('Crear nueva tarea: 3')
    print('Borrar: 4')
```

De igual manera revisamos como imprimir las acciones que se generan cuando se realiza la consulta. Si solo muestra las tareas en espera, en ejecución por aprobar o todas las tareas.

Definimos un valor numérico diferente para lograr hacer la consulta de forma lógica y así generar la menor cantidad de confusión para el usuario al momento de realizar la consulta, así el usuario solo debe leer la información, y escribir el numero que esta relacionado a la acción que necesitamos.

Esta será nuestra **acción 1.**

```
accion = input('Escriba la opcion:')
if not(accion=='1') and not(accion=='2') and not(accion=='3') and not(accion=='4'):
    print('Comando invalido por favor eliga una opcion valida')
elif accion=='1':
     opc_consulta=''
     print('Indique la tarea que desea consultar: ')
     print('Todas las tareas: 1')
     print('En espera: 2')
     print('En ejecucion: 3')
     print('Por aprobar: 4')
     print('Finalizada: 5')
     opc_consulta = input('Escriba la tarea que dese consultar:')
     if opc_consulta=='1':
         print()
         print()
         print('** Consultando todas las tareas **')
         leer(rut,'todo')
     elif opc_consulta=='2':
         print()
         print()
         print('** Consultando tareas en espera **')
         leer(rut,'En espera')
     elif opc_consulta=='3':
         print()
         print()
         print('** Consultando tareas en ejecucion **')
         leer(rut,'En ejecucion')
     elif opc_consulta=='4':
         print()
         print()
         print('** Consultando tareas por aprobar **')
         leer(rut,'Por aprobar')

     elif opc_consulta=='5':
         print()
         print()
         print('** Consultando tareas finalizadas **')
         leer(rut,'Finalizada')
```

Ahora vamos a generar las instruccion es para imprimir en consola para actualizar la información del estado de las tareas.

Definimos también para el usuario la opción de continuar si no desea actualizar, y en caso contrario definir la opción de cambio que desea.

Esta será nuestra **acción 2.**

```
elif accion=='2':
    datosActualizados={'titulo':'', 'descripcion':'', 'estado':'', 'fecha inicio':'', 'fecha finalizac
    print('** Actualizar Tarea **')
    print()
    id_Actualizar=int(input('Indique el Id de la tarea que desea actualizar: '))
    print()
    print('** Nuevo titulo **')
    print('**Nota: si no desea actualizar el titulo solo oprima ENTER')
    datosActualizados['titulo']=input('Indique el nuevo titulo de la tarea : ')
    print()
    print('** Nueva descripcion **')
    print('**Nota: si no desea actualizar la descripcion solo oprima ENTER')
    datosActualizados['descripcion']= input('Indique la nueva descripcion de la tarea : ')
    print()
    print('** Nueva estado **')
    print('En espera: 2')
    print('En ejecucion: 3')
    print('Por aprobar 4')
    print('Finalizada: 5')

    print('**Nota: si no desea actualizar el estado solo oprima ENTER')
    estadoNuevo= input('Indique el nuevo estado de la tarea : ')
    if estadoNuevo=='2':
        datosActualizados['estado']='En espera'
    elif estadoNuevo=='3':
        datosActualizados['estado']='En ejecucion'
    elif estadoNuevo=='4':
        datosActualizados['estado']='Por aprobar'
    elif estadoNuevo=='5':
        now = datetime.now()
        datosActualizados['estado']='Finalizada'
        datosActualizados['fecha finalizacion']=str(now.day) +'/'+ str(now.month) +'/'+str(now.year)

    now = datetime.now()
    datosActualizados['fecha inicio']=str(now.day) +'/'+ str(now.month) +'/'+str(now.year)
    actualizar(rut,id_Actualizar, datosActualizados)
    print()
```

Ahora generamos una actualización mas discriminada, la cual le permita al usuario actualizar el titulo, la descripción de la tarea o las fechas de finalización.

Esta será nuestra **acción 3**.

```
elif accion=='3':
    datosActualizados={'tarea':'', 'descripcion':'', 'estado':'', 'fecha inicio':'', 'fecha finalizaci
    print('** Crear nueva Tarea **')

    print()
    print('** titulo **')
    print()
    datosActualizados['titulo']=input('Indique el titulo de la tarea : ')
    print()
    print('** descripcion **')
    datosActualizados['descripcion']= input('Indique la descripcion de la tarea : ')
    print()
    datosActualizados['estado']='En espera'
    now = datetime.now()
    datosActualizados['fecha inicio']=str(now.day) +'/'+ str(now.month) +'/'+str(now.year)
    datosActualizados['fecha finalizacion']=''
    agregar(rut,datosActualizados)
```

Con la ultima acción podremos darle la opción al usuario de poder eliminar la tarea que desea.

Esta será nuestra **acción 4**.

```
elif accion=='4':
    print('')
    print('** Eliminar Tarea **')
    iden=int(input('Indique el Id de la tarea que desea eliminar: '))
    borrar(rut,iden)
```

¡Ahora solo debemos ejecutar el código y comenzar a consultar y editar nuestra base de datos desde la consola!

Una vez ejecutamos el código, nos aparecen las opciones que definimos anteriormente, para cada acción que generamos nos indica el valor numérico que debemos escribir en consola, si por ejemplo escribimos la **opción 1.**

Al oprimir **enter** nos aparecen las nuevas opciones que nos preguntan, al haber seleccionado la opción 1 ahora nos aparecen opciones diferentes que nos indican que tipo de tareas queremos ver en consola.

Por lo tanto podremos revisar o consultar la información de las tareas que cumplen con la condición que seleccionamos.

```
Indique la accion que desea realizar:
Consultar: 1
Actualizar: 2
Crear nueva tarea: 3
Borrar: 4

Escriba la opcion:1
Indique la tarea que desea consultar:
Todas las tareas: 1
En espera: 2
En ejecucion: 3
Por aprobar: 4
Finalizada: 5

Escriba la tarea que dese consultar:3
```

Cadenas con formato:

De igual manera, podemos formatear cadenas con distintos tipos de datos utilizando una plantilla que con tenga marcas que puedan ser reemplazadas por el contenido que queremos introducir.

Para definir la plantilla la definimos con el símbolo de porcentaje (**%**), si queremos introducir una cadena podemos utilizar entonces **%s**.

```
nombre = "Liliana"

cadena_formateada = "¡Hola %s!" % nombre

print (cadena_formateada)
```

En el ejemplo anterior, el símbolo de % seguido de la letra s dentro del texto es una especia de marca o marcador, que indica que en se lugar o posición se introducirá un valor que será formateado como una cadena. Luego después del texto y con símbolo de % ponemos el valor que queremos colocar.

Si el contenido no es una cadena, Python lo convertirá al introducirlo, de modo que no debemos preocuparnos por transformar el valor.

```
print ("Esto es una letra M: %s" % "M")

print ("Esto es una numero 8: %s" % 8)

print ("Esta es una lista: %s" % [10, 30, 100])
```

Podemos colocar en la platilla tantos valores como queramos, colocando los valores entre paréntesis. Solo debemos recordar muy bien que el numero de marcadores dentro de debe ser igual al número de valores que escribiremos mas adelante.

```
nombre1 = "Liliana"

nombre2 = "Catalina"


print ("¡Hola %s y %s bienvenidas!" % (nombre1, nombre2))
```

También podemos fijar el ancho mínimo que ocupara el valor introducido en la plantilla indicándolo entre el signo de % y la letra s.

Si el texto es mas corto, se completara con espacios en blanco, si ponemos números negativos los espacios aparecerán detrás del texto:

```
Primeros_meses= ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio"]

for i in Primeros_meses:

    print("* %10s * %-10s *" % (i,i))
```

```
*      Enero * Enero      *
*    Febrero * Febrero    *
*      Marzo * Marzo      *
*      Abril * Abril      *
*       Mayo * Mayo       *
*      Junio * Junio      *
```

Si necesitamos formatear números esteros, podemos hacerlo con %d:

```
print("El lenguaje de programación %s nacio en el año %d" % ("Python",1991))
```

```
El lenguaje de programación Python nacio en el año 1991
```

Si el numero introducido no es un entero, se convertirá antes de ser añadido a la cadena de caracteres.

```
hexadecimal = 0XA0
decimal = 10.5

print("El numero hexadecimal es %d y el decimal es %d" %(hexadecimal, decimal))
```

```
El numero hexadecimal es 160 y el decimal es 10
```

Del mismo modo usando %x o %X podemos representar números en notación hexadecimal, con caracter es alpabeticos en mayúsculas o minúsculas, respectivamente. También s posible utilizar los códigos %e y %E para usar la notación exponencial y %o para mostrar el numero octal.

```
numero = 1000

print("%d en hexadecimal (minusculas) es %x" %(numero, numero))

print("%d en hexadecimal (minusculas) es %X" %(numero, numero))

print("%d en hexadecimal (minusculas) es %e" %(numero, numero))

print("%d en hexadecimal (minusculas) es %E" %(numero, numero))
```

```
1000 en hexadecimal (minusculas) es 3e8
1000 en hexadecimal (minusculas) es 3E8
1000 en hexadecimal (minusculas) es 1.000000e+03
1000 en hexadecimal (minusculas) es 1.000000E+03
```

Para poner numero con decimales se usa %f:

```
tercio = 1.0 / 3

print("Un tercio (sin decimales) es %d" % tercio)

print("Un tercio es %f" % tercio)
```

```
Un tercio (sin decimales) es 0
Un tercio es 0.333333
```

Si tras el signo de porcentaje añadimos un punto seguido de un numero y la letra “f”, indica cual es el numero máximo de numero decimales o dígitos decimales que deseamos mostrar en consola:

```
tercio = 1.0 / 3

print("Un tercio es %f" % tercio)

print("Un tercio es %.2f" % tercio)
```

```
Un tercio es 0.333333
Un tercio es 0.33
```

Interacción directa con el usuario:

Hemos desarrollado ejemplos donde el usuario puede participar dentro del calculo o el desarrollo de nuestro problema y hacerlo mas interactivo con el usuario.

Python tiene de igual manera una función definida para detener el proceso, y pedir la información necesaria al usuario.

Para este proceso tenemos la función **input()** la cual muestra un texto al usuario y detiene la ejecución del programa mientras espera una respuesta del teclado o por parte del usuario.

La función se encargara de recolectar todo lo que se escriba en el teclado hasta que se pulse la tecla ENTER. EL texto que recibe la función será retornado como una cadena de texto que podemos utilizar como necesitemos.

```python
nombre = input("Escriba su nombre, por favor:")

print("Hola" + nombre )
```

```
Escriba su nombre, por favor: Laura
Hola Laura
```

Ejercicio de clase:

¡Con nuestro código **CRUD** funcionado correctamente para realizar consultas y modificaciones, realice modificaciones adicionales con estas funciones!

El futuro digital es de todos

MinTIC

GRACIAS

OPERADO POR:

Universidad Tecnológica de Pereira

MisiónTIC2022