

EJERCICIOS POR TEMA

miércoles, 27 de marzo de 2024

Tema 1

CONCEPTOS BASICOS SOBRE ESTRUCTURAS DE DATOS

Imprimir una cadena de entrada

```
# - - - Ingrese una cadena
cadena = input()
print(cadena)

# - - - Ingrese un número entero
numero_entero = int(input())
print(numero_entero)

# - - - Ingrese un número flotante
numero_flotante = float(input())
print(numero_flotante)

# - - - Ingrese un booleano (como "True" o "False")
booleano_str = input()
    # - - - Convertir la entrada a un booleano
booleano = booleano_str.lower() == "true"
print(booleano)

# - - - Ingrese su nombre y edad
nombre = input()
edad = int(input())
print(f"{nombre}, {edad}")
```

Declaración de Variables

```
#Forma: identificador = expresion
# Ejemplos
x = 5
x = 5 + 5
x = 5 + 5 * 5 # 30, primero se evalúa 5 * 5
x = (5 + 5) * 5 # 50, primero se evalúa 5 + 5
x = suma(5, 5) # invocación a función
x = suma(5, 5) - 1 # válido porque ambas son expresiones enteras
x = 'hola' # cadena, se puede cambiar el tipo
x = 'hola ' + 'mundo' # válido porque + se refiere a concatenación
x = 'hola' + 4 # error, incompatibilidad de tipos
y = 4
x += y # equivalente a x = x + y, hay variantes -=, *=, /=, %=
x = 4**2 # 4 elevado al cuadrado
x = 4 % 2 # operación de módulo
t = True or False
t = not t # intercambiar valor de verdad
b = 4 < 5 # b es True
```

```
b = 4 != 4 # b es False

# - - - Entero (int)
edad = 25

# - - - Flotante (float)
precio = 15.99

# - - - Cadena (str)
nombre = "Juan"

# - - - Booleano (bool)
activo = True

# - - - Lista (una colección ordenada y mutable de elementos):
numeros = [1, 2, 3, 4, 5]

# - - - Tupla (una colección ordenada e inmutable de elementos):
punto = (10, 20)

# - - - Diccionario (una colección no ordenada de pares clave-valor):
persona = {"nombre": "María", "edad": 30, "ciudad": "Madrid"}

# - - - Conjunto (una colección no ordenada de elementos únicos):
colores = {"rojo", "verde", "azul"}

# - - - Nulo (NoneType):
resultado = None
```

FASES DE CONTROL

```
If - - - Meses del año
num = int(input())
if num == 1:
    print('enero')
elif num == 2:
    print('febrero')
elif num == 3:
    print('marzo')
elif num == 4:
    print('abril')
elif num == 5:
    print('mayo')
elif num == 6:
    print('junio')
elif num == 7:
    print('julio')
elif num == 8:
    print('agosto')
elif num == 9:
    print('septiembre')
elif num == 10:
    print('octubre')
elif num == 11:
    print('noviembre')
elif num == 12:
    print('diciembre')
```

```

# - - - Calificación de estudiantes
calificacion = float(input())
if calificacion >= 6:
    print('Aprobaste')
else:
    print('Reprobaste')

# - - - Número par o impar
numero = int(input())
if numero % 2 == 0:
    print('Es número par')
else:
    print('Es un número impar')

# - - - Cálculo descuento
precio = float(input())
categoria = input()
a = 0.10
b = 0.20
c = 0.30
if categoria == 'a':
    resultado = precio - (precio * a)
elif categoria == 'b':
    resultado = precio - (precio * b)
elif categoria == 'c':
    resultado = precio - (precio * c)
print(resultado)

# - - - Triángulo según sus lados
lado1 = float(input())
lado2 = float(input())
lado3 = float(input())
if lado1 == lado2 and lado2 == lado3:
    print('Es triángulo equilátero')
elif lado1 == lado2 or lado1 == lado3 or lado2 == lado3:
    print('El triángulo es isósceles')
else:
    print('El triángulo es escaleno')

```

While -----

```

# - - - Contador regresivo (de x número al 1)
numero = int(input())
while numero >= 1:
    print(numero)
    numero -= 1

# - - - Tabla de multiplicar
numero = int(input())
hasta_numero = int(input())
i = 1
while i <= hasta_numero: #puede ser hasta el 10 como normalmente
    print(numero, 'x', i, '=', numero * i)
    i = i + 1

# - - - Calculo de potencia
base = int(input())
exponente = int(input())
resultado = 1
i = 1

```

```

while i <= exponente:
    resultado *= base
    i = i + 1
print(base, 'elevado a', exponente, 'es:', resultado)

# --- Secuencia de Fibonacci
n = int(input())
a, b = 0, 1
contador = 0
while contador < n:
    print(a, end = ' ')
    a, b = b, a + b
    contador = contador + 1

# --- Contador de dígitos que tiene un x numero
numero = int(input())
contador = 0
while numero != 0:#while se ejecuta mientras número sea diferente de
0
    numero /= 10
    contador = contador + 1
print(contador, 'dígitos')

```

While - utilizar

break-----

```

# --- Búsqueda de un número
def encontrar_numero(numero_a_buscar):
    numero = 1
    while True:
        if numero == numero_a_buscar:
            print(f"Número {numero_a_buscar} encontrado!")
            break
        print("Buscando...")
        numero += 1
    if __name__ == '__main__':
        numero_buscado = int(input())
        encontrar_numero(numero_buscado)

```

While - utilizar

continue-----

```

# --- Impresión de números pares
def imprimir_numeros_pares():
    inicio = int(input())
    fin = int(input())
    print("Números pares en el rango:")
    numero_actual = inicio
    while numero_actual <= fin:
        if numero_actual % 2 != 0: #Si número impar,
omitirlo
            numero_actual += 1
            continue
        print(numero_actual)
        numero_actual += 2 # Saltar al siguiente número
par
    if __name__ == '__main__':
        imprimir_numeros_pares()

```

FUNCIONES

```
# --- Sumar dos números
```

```

def suma(a, b):
    resultado = a + b
    return resultado
if __name__ == '__main__':
    a = int(input())
    b = int(input())
    # Llamando a la función y almacenando resultado en variable
    resultado_suma = suma(a, b)
    print(resultado_suma)

# - - - Saludo personalizado
def saludar(nombre):
    mensaje = 'Hola, ' + nombre + ', ¿Cómo estás?'
    return mensaje
if __name__ == '__main__':
    nombre_usuario = input() #solicitar nombre
    #Llamando a la función y mostrando mensaje
    print(saludar(nombre_usuario))

```

Tema 2

EJECUCIÓN Y MANEJO DE PROGRAMAS

Comandos para el examen: Aquí :)

...

Tema 3

CADENAS DE TEXTO

ESTÁNDARES DE TEXTO

ASCII

Unicódigo - - - - -

- En Python, la función `ord` devuelve el punto de código correspondiente a un carácter
- Mientras que la función `chr` hace la operación inversa

	Resultados en consola
<code>print(ord('a'))</code>	97
<code>print(ord('A'))</code>	65
<code>print(chr(97))</code>	a
<code>print(chr(65))</code>	A
<code># un carácter en japones</code>	12371
<code>print(ord('🍣'))</code>	128512
<code># un emoji</code>	
<code>print(ord('😊'))</code>	

MANEJO DE CADENAS

Tipo Str - - - - -

```

# - - - Ejemplo 1: Cadena de longitud 1
s2 = 'a'
print(s2[0]) # Imprime el único carácter de la cadena, que es 'a'.

```

```

print(s2[-1]) # Imprime el único carácter de la cadena, que es 'a'.
print(s2[len(s2)-1]) # Imprime también el único carácter de la cadena,
que es 'a'.

# - - - Ejemplo 2: Cadena con varios caracteres
s3 = 'python'
print(s3[0]) # Imprime el primer carácter de la cadena, que es 'p'.
print(s3[-1]) # Imprime el último carácter de la cadena, que es 'n'.
print(s3[len(s3)-1]) # Imprime también el último carácter de la cadena,
que es 'n'.

```

Creación de cadenas - - - - -

Dos formas validas

- 'Ejemplo de cadena'
- "Ejemplo de cadena"

```

print("My father's house")
print('My father\'s house') # se escapa la '
print('Primera línea\nSegunda línea')
print('\tCon sangría')

```

Resultados en consola
 My father's house
 My father's house
 Primera línea
 Segunda línea
 Con sangría

Cadenas multilínea - - - - -

```

s = '''Esta es una
cadena de varias líneas
útil para crear formatos de texto
o documentar funciones y módulos
'''

print(s)

```

Recorrido de cadenas - - - - -

```

# - - - Recorrido básico de la cadena e impresión de cada carácter
en una línea separada.
cadena = "Python"
for carácter in cadena:
    print(carácter)

```

Resultado:

```
P
y
t
h
o
n
```

```

# - - - Recorrido inverso de la cadena e impresión de cada
carácter en una línea separada.
cadena = "Hola mundo"
for i in range(len(cadena)-1, -1, -1):
    print(cadena[i])

```

Resultado:

```
o
d
n
u
m
a
l
o
```

H

```
# - - - Recorrido de la cadena con salto de dos caracteres e  
impresión de cada carácter en una Línea separada.  
cadena = "ABCDEFGHIJ"  
for i in range(0, len(cadena), 2):  
    print(cadena[i])
```

Resultado:

A
C
E
G
I

SUBCADENAS

```
# Ejemplo 1: Copia completa de la cadena 'buenos dias'  
s = 'buenos dias'  
print(s[0:11])  
  
# Ejemplo 2: Mismo resultado que el anterior usando el tamaño de  
la cadena  
print(s[:len(s)])  
  
# Ejemplo 3: Otra forma de obtener la cadena completa  
print(s[:])  
  
# Ejemplo 4: Todos los caracteres excepto el primero ('uenos  
dias')  
print(s[1:])  
  
# Ejemplo 5: Todos los caracteres excepto el último ('buenos dia')  
print(s[:-1])  
  
# Ejemplo 6: Las dos letras entre el espacio ('os di')  
print(s[4:9])  
  
# Ejemplo 1: Obtener una subcadena invertida  
cadena = "Python"  
print(cadena[::-1]) # Cadena invertida: "nohtyP"  
  
# Ejemplo 2: Obtener una subcadena con inicio y fin específicos  
cadena = "Mi nombre es Grissel"  
print(cadena[3:12]) # Desde el índice 3 hasta el 12: "nombre es"
```

Operaciones sobre cadenas - - - - -

Len es una función genérica de Python que permite determinar el número de elementos en varias estructuras de datos

```
# - - - Obtener la longitud de una cadena de texto.  
cadena = "Hola mundo"  
longitud = len(cadena)  
print("La longitud de la cadena es:", longitud) # Imprime:  
La Longitud de la cadena es: 10  
  
# - - - Obtener la longitud de una lista.  
lista = [1, 2, 3, 4, 5, 6]  
longitud = len(lista)  
print("La longitud de la lista es:", longitud) # Imprime: La  
Longitud de la lista es: 6
```

- **Concatenación**

```
s = 'hola ' + 'mundo'  
print(s)
```

- **Strip**

Quita caracteres de espaciadores a izquierda y derecha

```
s = ' hola mundo \n\n\n'.strip()  
print(s) # hola mundo
```

- - - Eliminar espacios en blanco al principio y al final de una cadena.

```
cadena = " Hola mundo "  
limpia = cadena.strip()  
print(limpia) # Imprime: "Hola mundo"
```

- - - Eliminar caracteres específicos al principio y al final de una cadena.

```
cadena = "---Hola mundo---"  
limpia = cadena.strip("-")  
print(limpia) # Imprime: "Hola mundo"
```

- - - Eliminar solo los espacios en blanco al principio de una cadena.

```
cadena = " Hola mundo "  
limpia = cadena.lstrip()  
print(limpia) # Imprime: "Hola mundo "
```

- - - Eliminar solo los espacios en blanco al final de una cadena.

```
cadena = " Hola mundo "  
limpia = cadena.rstrip()  
print(limpia) # Imprime: " Hola mundo"
```

- **Split**

Dada una cadena separadora, genera una lista de cadenas considerando el separador

```
s = 'nombre,edad,carrera,matricula'  
partes = s.split(',')  
print(partes)
```

```
s = 'hola-->mando'  
print(s.split('-->')) # ['nombre', 'edad', 'carrera', 'matricula']
```

- - - Dividir una cadena en palabras usando espacios como delimitador.

```
cadena = "Hola mundo feliz"  
palabras = cadena.split()  
print(palabras) # Imprime: ['Hola', 'mundo', 'feliz']
```

- - - Dividir una cadena en palabras usando una coma como delimitador.

```
cadena = "manzana,naranja,plátano"  
frutas = cadena.split(",")  
print(frutas) # Imprime: ['manzana', 'naranja', 'plátano']
```

- - - Dividir una cadena en caracteres individuales.

```
cadena = "Python"  
caracteres = list(cadena)
```

```

print(caracteres) # Imprime: ['P', 'y', 't', 'h', 'o', 'n']

# - - - Dividir una cadena en Líneas utilizando el carácter
de nueva línea como delimitador.
cadena = "Primera línea\nSegunda línea\nTercera línea"
lineas = cadena.split("\n")
print(lineas)
# Imprime: ['Primera línea', 'Segunda Línea', 'Tercera
Línea']

```

- **Startswith**

Regresa verdadero si una cadena empieza con una subcadena

```
print('hola mundo'.endswith('mundo')) # True
```

```

# - - - Verificar si una cadena comienza con un prefijo
específico
cadena = "Hola mundo"
if cadena.startswith("Hola"):
    print (True)
else:
    print(False) # salida: True

# - - - Filtrar una lista de cadenas que comienzan con un
prefijo específico:
lista = ["manzana", "banana", "pera", "malón", "naranja"]
resultado = [fruta for fruta in lista if
fruta.startswith("ma")]
print(resultado) # Output: ['manzana', 'malón']
# - - - Contar cuántas cadenas en una lista comienzan con un
prefijo específico
lista = ["gato", "perro", "gallina", "caballo", "cabra"]
contador = sum(1 for palabra in lista if
palabra.startswith("g"))
print("Cantidad de palabras que comienzan con 'g':",
contador) # Output: 2

```

- **Endswith**

Regresa verdadero si una cadena termina con cierta subcadena

```
print('hola mundo'.endswith('mundo')) # True
```

```

# - - - Verificar si una cadena termina con un sufijo
específico
cadena = "Hola mundo"
if cadena.endswith("mundo"):
    print("La cadena termina con 'mundo'")
else:
    print("La cadena no termina con 'mundo'") # Salida: La
cadena termina con 'mundo'

# - - - Filtrar una lista de nombres de archivos que terminan
con una extensión específica
archivos = ["documento.txt", "imagen.jpg", "script.py",
"datos.csv", "presentacion.pptx"]
archivos_txt = [archivo for archivo in archivos if
archivo.endswith(".txt")]
print(archivos_txt) # Salida: ['documento.txt']

# - - - Contar cuántas cadenas en una lista terminan con un

```

```

sufijo específico
palabras = ["gato", "perro", "gallina", "caballo", "cabra"]
contador = sum(1 for palabra in palabras if
    palabra.endswith("o"))
print("Cantidad de palabras que terminan con 'o':",
    contador) # Salida: 3

• Join
Concatena cadenas en una lista de cadenas, usando la cadena
actual como separador
print('.join(['hola', 'mundo', 'mundial']))

# - - - Unir una lista de cadenas en una sola cadena usando
un separador
lista = ["Hola", "mundo", "Python"]
cadena = " - ".join(lista)
print(cadena) # Salida: Hola - mundo - Python

# - - - Convertir una lista de números en una cadena de texto
separada por comas
numeros = [1, 2, 3, 4, 5]
cadena_numeros = ", ".join(str(num) for num in numeros)
print(cadena_numeros) # Salida: 1, 2, 3, 4, 5

# - - - Unir las claves y valores de dos listas en una cadena
de texto formateada
claves = ["nombre", "edad", "ciudad"]
valores = ["Juan", 30, "Madrid"]
cadena_lista = ", ".join(f"{claves[i]}: {valores[i]}" for i
in range(len(claves)))
print(cadena_lista) # Salida: nombre: Juan, edad: 30,
ciudad: Madrid

```

PLANTILLAS DE TEXTO

Son una forma conveniente de crear cadenas con espacios que se van a llenar después

Operador % -----

```

# - - - Calcular si un año es bisiesto
anio = int(input("Ingrese un año: ")) # 2024
if (anio % 4 == 0 and anio % 100 != 0) or (anio % 400 == 0):
    print("El año es bisiesto.")
else:
    print("El año no es bisiesto.") # El año es bisiesto
# - - - Convertir segundos a minutos y segundos:
segundos_totales = int(input("Ingrese la cantidad de segundos: "))
# 180
minutos = segundos_totales // 60
segundos_restantes = segundos_totales % 60
print(f"{segundos_totales} segundos son {minutos} minutos y
{segundos_restantes} segundos.") # 180 segundos son 3 minutos y 0
segundos.
# Sustituciones
# - - - Sustitución de valores en una cadena de texto
nombre = input() # Grissel
edad = int(input()) # 19
mensaje = "Hola, %s. Tienes %d años." % (nombre, edad)

```

```

print(mensaje) # Hola, Grissel. Tienes 19 años.
# - - - Sustitución de valores en una cadena de formato
temperatura = 25.5
humedad = 60
informacion = "La temperatura es %.1f°C y la humedad es %d%." %
(temperatura, humedad)
print(informacion) # La temperatura es 25.5°C y la humedad es 60%.

```

Format -----

```

# - - - Formateo básico de cadenas
nombre = input()
edad = int(input())
mensaje = "Hola, {}. Tienes {} años.".format(nombre, edad)
print(mensaje) # Hola, Grissel. Tienes 19 años.

# - - - Especificación de índices de lugar y formatos
precio = float(input())
descuento = float(input())
precio_final = precio * (1 - descuento)
informacion = "El precio original es ${0:.2f}, con un descuento
del {1:.0%}, el precio final es ${2:.2f}.".format(precio,
descuento, precio_final)
print(informacion) # El precio original es $450.65, con un
descuento del 20%, el precio final es $360.52.

# - - - Formateo de alineación y relleno
nombre = "Juan"
apellido = "Pérez"
telefono = "123456789"
informacion = "Nombre: {:<10} Apellido: {:>10} Teléfono: {:^
10}.".format(nombre, apellido, telefono)
print(informacion) # Nombre: Juan           Apellido:      Pérez
Teléfono: 123456789

```

Cadenas f -----

```

# - - - Interpolación de variables en una cadena
nombre = "María"
edad = 25
mensaje = f"Hola, {nombre}. Tienes {edad} años."
print(mensaje)

# - - - Cálculo dentro de una cadena f
radio = float(input())
area = f"El área de un círculo con radio {radio} es {3.14 * radio
** 2}." 
print(area) # El área de un círculo con radio 30.0 es 2826.0.

# - - - Usando expresiones más complejas en una cadena f
precio_unitario = float(input()) # 567.45
cantidad = int(input()) # 3
total = f"El total a pagar es: {precio_unitario * cantidad * (1 -
0.1)}." 
print(total) # El total a pagar es: 1532.1150000000002.

```

Tema 4

OBJETOS

Clases

- Son plantillas para crear objetos
- Una clase define un tipo de dato

```
# - - - Clase Persona para representar a una persona con nombre y edad
class Persona():
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def presentarse(self):
        return f"¡Hola! Mi nombre es {self.nombre} y tengo {self.edad} años."

# Crear objetos de la clase Persona
persona1 = Persona("Juan", 30)
persona2 = Persona("María", 25)

# Utilizar métodos de la clase Persona
print(persona1.presentarse()) #¡Hola! Mi nombre es Juan y tengo 30 años.
print(persona2.presentarse())#¡Hola! Mi nombre es María y tengo 25 años.

# - - - Clase Rectángulo para representar un rectángulo con ancho y altura
class Rectangulo():
    def __init__(self, ancho, altura):
        self.ancho = ancho
        self.altura = altura

    def calcular_area(self):
        return self.ancho * self.altura

# Crear objetos de la clase Rectángulo
rectangulo1 = Rectangulo(5, 10)
rectangulo2 = Rectangulo(3, 7)

# Utilizar métodos de la clase Rectángulo
print("Área del rectángulo 1:", rectangulo1.calcular_area()) #Área del rectángulo 1: 50
print("Área del rectángulo 2:", rectangulo2.calcular_area()) #Área del rectángulo 2: 21

# - - - Clase Libro para representar un libro con título y autor
class Libro():
    def __init__(self, titulo, autor):
        self.titulo = titulo
        self.autor = autor

    def informacion(self):
        return f"El libro '{self.titulo}' fue escrito por {self.autor}."

# Crear objetos de la clase Libro
libro1 = Libro("Harry Potter", "J.K. Rowling")
libro2 = Libro("Cien años de soledad", "Gabriel García Márquez")

# Utilizar métodos de la clase Libro
print(libro1.informacion())#El libro 'Harry Potter' fue escrito por J.K. Rowling.
print(libro2.informacion())#El libro 'Cien años de soledad' fue escrito por Gabriel García Márquez.
```

Objetos

- Son instancias de una clase, administradas por el proceso
- Esencialmente son contenedores de valores (atributos/propiedades) y pueden tener un comportamiento (métodos)

```
class Ejemplo:
```

```
    # se necesita un método especial __init__  
    def __init__(self):  
        pass
```

```
objeto1 = Ejemplo() # instanciación de clase  
objeto2 = Ejemplo() # otro objeto de la misma clase  
print(type(objeto1))
```

Propiedades / Atributos

- Son similares a las variables, pero sólo existen dentro del objeto
- Se pueden crear desde el método especial __init__

```
class Persona:
```

```
    def __init__(self, nombre, edad):  
        self.nombre = nombre # Atributo de instancia  
        self.edad = edad     # Atributo de instancia
```

Métodos

- Son similares a las funciones, pero en el contexto de un objeto
- Esto es, tienen acceso directo a las propiedades (a través de self)
- En un método, el primer atributo debe ser la referencia especial self, incluso si el método no recibe parámetros se debe poner

```
# - - - Método para calcular el área de un rectángulo en una  
# clase Rectangulo  
class Rectangulo:  
    def __init__(self, ancho, altura):  
        self.ancho = ancho  
        self.altura = altura  
  
    def calcular_area(self): # este es un método  
        return self.ancho * self.altura  
  
# Crear objeto de la clase Rectangulo  
rectangulo = Rectangulo(5, 10)  
# Llamar al método para calcular el área del rectángulo  
area = rectangulo.calcular_area()  
print("Área del rectángulo:", area)
```

```
# - - - Método para saludar en una clase Persona
```

```
class Persona:  
    def __init__(self, nombre):  
        self.nombre = nombre  
  
    def saludar(self): # este es un método  
        print(f"Hola, soy {self.nombre}. ¡Mucho gusto!")  
  
# Crear objeto de la clase Persona  
persona = Persona("Juan")  
# Llamar al método para que la persona salude  
persona.saludar()
```

```

# - - - Método para verificar si un número es par en una
# clase Número
class Número:
    def __init__(self, valor):
        self.valor = valor

    def es_par(self): # este es un método
        if self.valor % 2 == 0:
            return True
        else:
            return False
# Crear objeto de la clase Número
número = Número(7)
# Llamar al método para verificar si el número es par
if número.es_par():
    print("El número es par.")
else:
    print("El número es impar.")

```

Igualdad en objetos

```

# - - - Comparación de atributos en una clase Persona
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def __eq__(self, other):
        return self.nombre == other.nombre and self.edad ==
other.edad
# Crear objetos de la clase Persona
persona1 = Persona("Juan", 30)
persona2 = Persona("Juan", 30)
persona3 = Persona("María", 25)
# Comparar objetos de la clase Persona
print("¿persona1 es igual a persona2?", persona1 == persona2)
print("¿persona1 es igual a persona3?", persona1 == persona3)

```

```

# - - - ejemplo que utiliza la función isinstance() para
determinar si un objeto es una instancia de una clase
específica
class Animal:
    def __init__(self, nombre):
        self.nombre = nombre
class Perro(Animal):
    def ladrar(self):
        print("¡Guau!")
class Gato(Animal):
    def maullar(self):
        print("¡Miau!")
# Crear instancias de las clases
mi_perro = Perro("Buddy")
mi_gato = Gato("Whiskers")
# Verificar si las instancias pertenecen a la clase Animal
print("¿mi_perro es una instancia de Animal?", 
isinstance(mi_perro, Animal))
print("¿mi_gato es una instancia de Animal?", 
isinstance(mi_gato, Animal))
# Verificar si las instancias pertenecen a las clases Perro y
Gato

```

```

print("¿mi_perro es una instancia de Perro?",  

      isinstance(mi_perro, Perro))  

print("¿mi_gato es una instancia de Gato?",  

      isinstance(mi_gato, Gato))

```

Tema 5

LISTAS

- Estructuras de datos lineales, dinámicas y mutables
- En general el acceso a elementos es secuencial y no aleatorio (como el de los arreglos), lo cual es ineficiente

Tipos principales

```

# - - - Declaración de una Lista
numeros = [1, 2, 3, 4, 5]

# - - - Acceder a elementos de la lista
print(numeros[0]) #Imprime el primer elemento de la lista (índice 0)
print(numeros[2]) #Imprime el tercer elemento de la lista (índice 2)

# - - - Modificar elementos de la Lista
numeros[0] = 10 #Modifica el primer elemento de la Lista
print(numeros)   #Imprime la lista modificada: [10, 2, 3, 4, 5]

# - - - Agregar elementos al final de la lista:
numeros.append(6) # Agrega el número 6 al final de la lista
print(numeros)    # Imprime la lista actualizada: [10, 2, 3, 4, 5, 6]

# - - - Eliminar elementos de la lista por valor:
numeros.remove(3) # Elimina el elemento 3 de la lista
print(numeros)    # Imprime la lista actualizada: [10, 2, 4, 5, 6]

# - - - Eliminar elementos de la lista por índice:
del numeros[0] # Elimina el primer elemento de la lista
print(numeros) # Imprime la lista actualizada: [2, 4, 5, 6]

# - - - Longitud de La lista
print(len(numeros)) # Imprime La Longitud de La lista

# - - - Iteración sobre una Lista
for numero in numeros:
    print(numero) # Imprime cada elemento de la lista en una Línea separada

# - - - Reversión de una Lista:
numeros.reverse() # Invierte el orden de los elementos en la lista
print(numeros)    # Imprime la lista invertida

# - - - Ordenamiento de una Lista:
numeros.sort() # Ordena los elementos de la lista de forma ascendente
print(numeros) # Imprime la lista ordenada

```

Algunos métodos útiles son:

- `insert(i,x)` inserta el elemento x en la posición i (después de la inserción, x ocupara la posición i).

```

# - - - Insertar un elemento en una posición específica usando
insert(i, x)
lista = [1, 2, 3, 4, 5]

```

```
lista.insert(2, 10)
print(lista) # Output: [1, 2, 10, 3, 4, 5]
```

- **append(x)** añade el elemento x al final de la lista (a.append(x) es equivalente a a.insert(len(a),x)).

```
# - - - Agregar un elemento al final de la Lista usando append(x)
lista = [1, 2, 3, 4, 5]
lista.append(6)
print(lista) # Output: [1, 2, 3, 4, 5, 6]
```

- **index(x)** devuelve el índice del primer elemento de la lista igual a x.

```
# - - - Encontrar el índice de un elemento usando index(x)
lista = [1, 2, 3, 4, 5]
indice = lista.index(3)
print(indice) # Output: 2
```

- **remove(x)** elimina el primer elemento de la lista igual a x.

```
# - - - Eliminar un elemento usando remove(x)
lista = [1, 2, 3, 4, 5]
lista.remove(3)
print(lista) # Output: [1, 2, 4, 5]
```

- **sort()** ordena los elementos de la lista.

```
# - - - Ordenar La Lista usando sort()
lista = [3, 1, 4, 2, 5]
lista.sort()
print(lista) # Output: [1, 2, 3, 4, 5]
```

- **reverse()** invierte el orden de los elementos de la lista.

```
# - - - Invertir el orden de la Lista usando reverse()
lista = [1, 2, 3, 4, 5]
lista.reverse()
print(lista) # Output: [5, 4, 3, 2, 1]
```

- **count(x)** cuenta el número de veces que x aparece en la lista.

```
# - - - Contar la cantidad de veces que aparece un elemento usando
count(x)
lista = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
cantidad = lista.count(3)
print(cantidad) # Output: 3
```

- Para eliminar elementos de la lista dando su posición en lugar de su valor, se puede utilizar del:

```
>>> a=[1,2,3,4,5]
>>> del a[2]
>>> a
[1, 2, 4, 5]
```

```
# - - - Eliminar un elemento en una posición específica usando del
lista = [1, 2, 3, 4, 5]
del lista[2]
```

```
print(lista) # Output: [1, 2, 4, 5]

# - - - Eliminar todos los elementos de una lista usando del con la lista entera
lista = [1, 2, 3, 4, 5]
del lista[:]
print(lista) # Output: []

# - - - Eliminar el último elemento de una lista usando del y slicing
lista = [1, 2, 3, 4, 5]
del lista[-1]
print(lista) # Output: [1, 2, 3, 4]
```