Apuntes Completos de Python

📚 Índice

📌 Arrays (Listas)

Creación de Listas

python

```python
# Listas simples
numeros = [1, 2, 3, 4, 5]
nombres = ["Ana", "Luis", "María", "Carlos"]
mixta = [10, "Hola", 3.14, True]
vacia = []
```

Acceso a Elementos

python

```python
numeros = [1, 2, 3, 4, 5]

print(numeros[0])   # Primer elemento → 1
print(numeros[-1])  # Último elemento → 5
print(numeros[-2])  # Penúltimo elemento → 4
print(numeros[1:3]) # Slice [inicio:fin) → [2, 3]
print(numeros[:3])  # Desde inicio hasta índice 3 → [1, 2, 3]
print(numeros[2:])  # Desde índice 2 hasta el final → [3, 4, 5]
```

Modificación de Elementos

python

```python
numeros = [1, 2, 3, 4, 5]
numeros[2] = 99    # Cambia el tercer elemento
print(numeros)     # [1, 2, 99, 4, 5]
```

Métodos Principales de Listas

Agregar elementos

```python
lista = [1, 2, 3]

lista.append(4)       # Agregar al final → [1, 2, 3, 4]
lista.insert(1, 100)  # Insertar en índice 1 → [1, 100, 2, 3, 4]
lista.extend([5, 6])  # Extender con otra lista → [1, 100, 2, 3, 4, 5, 6]
```

Eliminar elementos

```python
lista = [1, 2, 3, 4, 5, 3]

lista.remove(3)       # Elimina primera ocurrencia de 3 → [1, 2, 4, 5, 3]
ultimo = lista.pop()  # Elimina y devuelve el último → [1, 2, 4, 5]
elemento = lista.pop(1) # Elimina y devuelve índice 1 → [1, 4, 5]
del lista[0]          # Elimina elemento en índice 0 → [4, 5]
lista.clear()         # Vacía completamente la lista → []
```

Ordenar y manipular

```python
numeros = [5, 2, 8, 1, 9]

numeros.sort()        # Ordena ascendente → [1, 2, 5, 8, 9]
numeros.sort(reverse=True)  # Ordena descendente → [9, 8, 5, 2, 1]
numeros.reverse()     # Invierte el orden actual
```

Búsqueda y conteo

```python
lista = [1, 2, 3, 2, 4, 2]

indice = lista.index(3)    # Devuelve índice del valor 3 → 2
cantidad = lista.count(2)  # Cuenta ocurrencias de 2 → 3
existe = 5 in lista        # Comprueba si existe → False
longitud = len(lista)      # Longitud de la lista → 6
```

Recorrer Listas

Con for

```python
numeros = [1, 2, 3, 4, 5]

# Recorrer valores
for numero in numeros:
    print(numero)

# Recorrer con índice
for i in range(len(numeros)):
    print(f"Índice {i}: {numeros[i]}")

# Recorrer con enumerate (mejor práctica)
for i, numero in enumerate(numeros):
    print(f"Índice {i}: {numero}")
```

Con while

```python
i = 0
while i < len(numeros):
    print(numeros[i])
    i += 1
```

Listas por Comprensión

```python
# Crear lista de cuadrados
cuadrados = [x**2 for x in range(1, 6)]
# Resultado: [1, 4, 9, 16, 25]

# Con condición
pares = [x for x in range(10) if x % 2 == 0]
# Resultado: [0, 2, 4, 6, 8]

# Transformación
palabras = ["hola", "mundo", "python"]
mayusculas = [p.upper() for p in palabras]
# Resultado: ['HOLA', 'MUNDO', 'PYTHON']
```

Copiar Listas

python

```python
original = [1, 2, 3]

# ⚠️ NO hacer esto (copia la referencia)
copia_mal = original
copia_mal[0] = 99  # ¡Modifica el original también!

# ✅ Formas correctas de copiar
copia1 = original.copy()
copia2 = original[:]
copia3 = list(original)
```

Arrays Bidimensionales (Matrices)

python

```python
# Crear matriz
matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Acceso a elementos
print(matriz[0][0])  # Fila 0, Columna 0 → 1
print(matriz[2][1])  # Fila 2, Columna 1 → 8

# Modificar elemento
matriz[1][2] = 99

# Recorrer matriz
for fila in matriz:
    for elemento in fila:
        print(elemento, end=" ")
    print()  # Nueva línea después de cada fila

# Recorrer con índices
for i in range(len(matriz)):
    for j in range(len(matriz[i])):
        print(f"[{i}][{j}] = {matriz[i][j]}")

# Crear matriz con comprensión
ceros = [[0 for _ in range(3)] for _ in range(3)]
# [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

# Agregar una fila
nueva_fila = [10, 11, 12]
matriz.append(nueva_fila)

# Agregar una columna
for i in range(len(matriz)):
    matriz[i].append(i + 100)
```

🔖 Tuplas

¿Qué es una Tupla?

Una tupla es una colección **ordenada e inmutable**. Una vez creada, no se pueden modificar sus elementos.

Creación de Tuplas

python

```python
# Tuplas básicas
tupla1 = (1, 2, 3, 4, 5)
tupla2 = ("Ana", "Luis", "María", "Carlos")
mixta = (10, "Hola", 3.14, True)

# Tupla de un solo elemento (necesita coma)
un_elemento = (5,)

# Sin paréntesis (también es válido)
tupla3 = 1, 2, 3
```

Acceso a Elementos

python

```python
tupla1 = (1, 2, 3, 4, 5)

print(tupla1[0])    # Primer elemento → 1
print(tupla2[-1])   # Último elemento → "Carlos"
print(mixta[1:3])    # Slice → ("Hola", 3.14)
```

Inmutabilidad

python

```python
tupla = (1, 2, 3, 4, 5)

# ❌ Esto da error - Las tuplas NO se pueden modificar
# tupla[0] = 99  # TypeError: 'tuple' object does not support item assignment

# ✅ Puedes crear una nueva tupla
nueva_tupla = (99,) + tupla[1:]  # (99, 2, 3, 4, 5)
```

Operaciones con Tuplas

python

```python
tupla = (1, 2, 3, 2, 4, 2)

# Longitud
print(len(tupla))        # 6

# Contar ocurrencias
print(tupla.count(2))    # 3

# Obtener índice
print(tupla.index(3))    # 2

# Concatenar tuplas
tupla_a = (1, 2, 3)
tupla_b = (4, 5, 6)
tupla_c = tupla_a + tupla_b  # (1, 2, 3, 4, 5, 6)

# Repetir tupla
repetida = (1, 2) * 3  # (1, 2, 1, 2, 1, 2)

# Verificar existencia
print(2 in tupla)      # True
print(10 not in tupla) # True
```

Recorrer una Tupla

python

```python
tupla = ("Ana", "Luis", "María")

# Con for
for elemento in tupla:
    print(elemento)

# Con enumerate
for i, elemento in enumerate(tupla):
    print(f"Índice {i}: {elemento}")
```

Desempaquetado de Tuplas

python

```python
# Desempaquetado básico
x, y, z = (1, 2, 3)
print(x, y, z)  # 1 2 3

# Desempaquetado con *
primero, *resto = (1, 2, 3, 4, 5)
print(primero)  # 1
print(resto)    # [2, 3, 4, 5]

# Intercambio de valores (sin variable temporal)
a = 5
b = 10
a, b = b, a  # Ahora a=10, b=5

# Función que retorna múltiples valores
def obtener_coordenadas():
    return (10, 20)

x, y = obtener_coordenadas()
```

Tuplas Anidadas

python

```python
anidada = (1, (2, 3), (4, 5))

print(anidada[0])     # 1
print(anidada[1])     # (2, 3)
print(anidada[1][0])  # 2
print(anidada[2][1])  # 5
```

Conversión entre Tuplas y Listas

python

```python
# Lista a tupla
lista = [1, 2, 3, 4]
tupla = tuple(lista)  # (1, 2, 3, 4)

# Tupla a lista
tupla = (5, 6, 7, 8)
lista = list(tupla)   # [5, 6, 7, 8]
```

¿Cuándo Usar Tuplas?

- ✅ Datos que no deben cambiar (coordenadas, fechas, configuraciones)
- ✅ Claves de diccionarios (las listas no pueden ser claves)
- ✅ Retorno múltiple de funciones
- ✅ Ligeramente más rápidas que las listas
- ✅ Protegen datos contra modificaciones accidentales

Comparación Tupla vs Lista

Característica	Tupla	Lista
Sintaxis	`(1, 2, 3)`	`[1, 2, 3]`
Mutable	❌ No	✅ Sí
Velocidad	Más rápida	Más lenta
Uso de memoria	Menor	Mayor
Métodos	Solo 2 (`count`, `index`)	Muchos (`append`, `remove`, etc.)
Como clave de dict	✅ Sí	❌ No

🎲 Sets (Conjuntos)

¿Qué es un Set?

Un set es una colección **no ordenada** y **sin elementos duplicados**. Es útil para eliminar duplicados y realizar operaciones matemáticas de conjuntos.

Creación de Sets

```python
# Set básico
mi_set = {1, 2, 3, 4}
print(mi_set)  # {1, 2, 3, 4}

# Set vacío (¡OJO! {} crea un diccionario vacío)
vacio = set()

# Set desde lista (elimina duplicados automáticamente)
lista = [1, 2, 2, 3, 4, 4, 5]
set_sin_duplicados = set(lista)
print(set_sin_duplicados)  # {1, 2, 3, 4, 5}

# Set desde string
letras = set("abracadabra")
print(letras)  # {'a', 'b', 'r', 'c', 'd'}
```

Métodos de Sets

Agregar elementos

```python
mi_set = {1, 2, 3}

mi_set.add(4)       # Agregar un elemento → {1, 2, 3, 4}
mi_set.add(2)       # No agrega duplicados → {1, 2, 3, 4}

# Agregar múltiples elementos
mi_set.update([5, 6, 7])  # {1, 2, 3, 4, 5, 6, 7}
```

Eliminar elementos

```python
mi_set = {1, 2, 3, 4, 5}

# remove() - Da error si no existe
mi_set.remove(2)    # {1, 3, 4, 5}
# mi_set.remove(99)  # KeyError

# discard() - NO da error si no existe
mi_set.discard(99)  # No pasa nada
mi_set.discard(3)   # {1, 4, 5}

# pop() - Elimina un elemento "aleatorio" y lo devuelve
ultimo = mi_set.pop()
print(f"Elemento eliminado: {ultimo}")

# clear() - Vacía el set
mi_set.clear()      # set()
```

Operaciones de Conjuntos

Unión (todos los elementos)

```python
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}

# Método union()
print(a.union(b))    # {1, 2, 3, 4, 5, 6}

# Operador |
print(a | b)        # {1, 2, 3, 4, 5, 6}
```

Intersección (elementos comunes)

```python
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}

# Método intersection()
print(a.intersection(b))  # {3, 4}

# Operador &
print(a & b)              # {3, 4}
```

Diferencia (en A pero no en B)

```python
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}

# Método difference()
print(a.difference(b))    # {1, 2}

# Operador -
print(a - b)              # {1, 2}
print(b - a)              # {5, 6}
```

Diferencia Simétrica (en A o B, pero no en ambos)

```python
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}

# Método symmetric_difference()
print(a.symmetric_difference(b))  # {1, 2, 5, 6}

# Operador ^
print(a ^ b)                      # {1, 2, 5, 6}
```

Verificación y Comparación

python

```python
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}
c = {1, 2}

# Verificar si un elemento existe
print(3 in a)       # True
print(10 not in b)  # True

# Subconjunto (todos los elementos de c están en a)
print(c.issubset(a))  # True
print(c <= a)       # True

# Superconjunto (a contiene todos los elementos de c)
print(a.issuperset(c))  # True
print(a >= c)         # True

# Disyuntos (no tienen elementos en común)
x = {1, 2, 3}
y = {4, 5, 6}
print(x.isdisjoint(y))  # True
```

Recorrer un Set

python

```python
mi_set = {1, 2, 3, 4, 5}

# Con for (orden no garantizado)
for valor in mi_set:
    print(valor)

# No se puede acceder por índice
# print(mi_set[0])  # ❌ TypeError
```

Operaciones con Múltiples Sets

```python
a = {1, 2, 3}
b = {2, 3, 4}
c = {3, 4, 5}

# Unión de múltiples sets
union_total = a.union(b, c)  # {1, 2, 3, 4, 5}
union_total = a | b | c      # {1, 2, 3, 4, 5}

# Intersección de múltiples sets
inter_total = a.intersection(b, c)  # {3}
inter_total = a & b & c             # {3}
```

Casos de Uso Prácticos

Eliminar duplicados de una lista

```python
lista = [1, 2, 2, 3, 4, 4, 5, 1]
lista_sin_duplicados = list(set(lista))
# Resultado: [1, 2, 3, 4, 5]
```

Encontrar elementos únicos entre dos listas

```python
lista1 = [1, 2, 3, 4, 5]
lista2 = [4, 5, 6, 7, 8]

comunes = list(set(lista1) & set(lista2))     # [4, 5]
solo_lista1 = list(set(lista1) - set(lista2)) # [1, 2, 3]
todos = list(set(lista1) | set(lista2))       # [1, 2, 3, 4, 5, 6, 7, 8]
```

Verificar si todos los elementos son únicos

```python
lista = [1, 2, 3, 4, 5]
todos_unicos = len(lista) == len(set(lista))  # True

lista_con_duplicados = [1, 2, 2, 3]
todos_unicos = len(lista_con_duplicados) == len(set(lista_con_duplicados))  # False
```

Set Comprehension

```python
# Crear set de cuadrados
cuadrados = {x**2 for x in range(1, 6)}
# {1, 4, 9, 16, 25}

# Con condición
pares = {x for x in range(10) if x % 2 == 0}
# {0, 2, 4, 6, 8}
```

Frozenset (Set Inmutable)

```python
# Frozenset no se puede modificar
fs = frozenset([1, 2, 3, 4])

# No tiene métodos add(), remove(), etc.
# Puede ser clave de diccionario
diccionario = {fs: "valor"}

# Útil para sets de sets
set_de_sets = {frozenset([1, 2]), frozenset([3, 4])}
```

Resumen de Operaciones de Sets

Operación	Método	Operador	Resultado
Unión	`a.union(b)`	`a | b`	Todos los elementos
Intersección	`a.intersection(b)`	`a & b`	Elementos comunes
Diferencia	`a.difference(b)`	`a - b`	En a pero no en b
Diferencia simétrica	`a.symmetric_difference(b)`	`a ^ b`	En a o b, no ambos
Subconjunto	`a.issubset(b)`	`a <= b`	a está contenido en b
Superconjunto	`a.issuperset(b)`	`a >= b`	a contiene a b
Disyuntos	`a.isdisjoint(b)`	-	Sin elementos comunes

📝 Strings (Cadenas)

Creación de Strings

python

```python
# Comillas simples o dobles
cadena1 = "Hola mundo"
cadena2 = 'Python es genial'

# Comillas triples (multilínea)
cadena3 = """Este es un string
con varias líneas
de texto"""

# String de números
cadena4 = "12345"

# String vacío
vacio = ""
```

Acceso a Caracteres

python

```python
texto = "Hola mundo"

print(texto[0])     # Primer carácter → "H"
print(texto[-1])    # Último carácter → "o"
print(texto[0:4])   # Slice → "Hola"
print(texto[:4])    # Desde inicio → "Hola"
print(texto[5:])    # Hasta el final → "mundo"
print(texto[-5:])   # Últimos 5 → "mundo"
```

Inmutabilidad de Strings

python

```python
texto = "Hola"

# ❌ No se pueden modificar caracteres directamente
# texto[0] = "h"  # TypeError

# ✅ Crear nuevo string
texto = "h" + texto[1:]  # "hola"
```

Concatenación y Repetición

```python
# Concatenación con +
saludo = "Hola " + "Python"
print(saludo)  # "Hola Python"

# Repetición con *
eco = "Hey! " * 3
print(eco)  # "Hey! Hey! Hey! "

# Concatenación múltiple
nombre = "Ana"
mensaje = "Hola, " + nombre + "!"  # "Hola, Ana!"
```

Métodos de Transformación

Mayúsculas y minúsculas

```python
texto = "Bienvenido a Python"

print(texto.upper())      # "BIENVENIDO A PYTHON"
print(texto.lower())      # "bienvenido a python"
print(texto.title())      # "Bienvenido A Python"
print(texto.capitalize()) # "Bienvenido a python"
print(texto.swapcase())   # "bIENVENIDO A pYTHON"
```

Limpieza de espacios

```python
texto = "  Bienvenido a Python  "

print(texto.strip())  # "Bienvenido a Python" (ambos lados)
print(texto.lstrip()) # "Bienvenido a Python  " (izquierda)
print(texto.rstrip()) # "  Bienvenido a Python" (derecha)

# Eliminar caracteres específicos
texto2 = "***Hola***"
print(texto2.strip("*")) # "Hola"
```

Reemplazo

```python
texto = "Bienvenido a Python"

print(texto.replace("Python", "Java"))  # "Bienvenido a Java"
print(texto.replace("e", "3"))          # "Bi3nv3nido a Python"
print(texto.replace("e", "3", 1))       # "Bi3nvenido a Python" (solo 1 vez)
```

Búsqueda en Strings

Encontrar subcadenas

```python
texto = "Bienvenido a Python"

# find() - Devuelve índice o -1 si no existe
print(texto.find("Python"))    # 13
print(texto.find("Java"))      # -1

# index() - Devuelve índice o lanza error
print(texto.index("Python"))   # 13
# print(texto.index("Java"))   # ValueError

# rfind() / rindex() - Busca desde la derecha
print(texto.rfind("e"))        # 7
```

Verificar contenido

```python
texto = "Bienvenido a Python"

# in / not in
print("Python" in texto)       # True
print("Java" not in texto)     # True

# startswith() / endswith()
print(texto.startswith("Bien")) # True
print(texto.endswith("thon"))  # True
print(texto.endswith("Java"))  # False
```

Contar ocurrencias

```python
texto = "Bienvenido a Python"

print(texto.count("e"))       # 2
print(texto.count("Python"))  # 1
print(texto.count("x"))       # 0
```

División y Unión de Strings

split() - Dividir string en lista

```python
frase = "uno,dos,tres,cuatro"

# Dividir por coma
lista = frase.split(",")
print(lista)  # ['uno', 'dos', 'tres', 'cuatro']

# Dividir por espacios (por defecto)
texto = "Hola mundo Python"
palabras = texto.split()
print(palabras)  # ['Hola', 'mundo', 'Python']

# Limitar divisiones
resultado = frase.split(",", 2)
print(resultado)  # ['uno', 'dos', 'tres,cuatro']
```

join() - Unir lista en string

```python
lista = ["uno", "dos", "tres", "cuatro"]

# Unir con guion
resultado = "-".join(lista)
print(resultado)  # "uno-dos-tres-cuatro"

# Unir con espacio
resultado = " ".join(lista)
print(resultado)  # "uno dos tres cuatro"

# Unir sin separador
resultado = "".join(lista)
print(resultado)  # "unodostrescuatro"
```

Formateo de Strings

f-strings (Python 3.6+) - Recomendado

```python
nombre = "Ana"
edad = 25
altura = 1.65

# Básico
print(f"Me llamo {nombre} y tengo {edad} años")

# Con expresiones
print(f"El año que viene tendré {edad + 1} años")

# Formateo de números
print(f"Altura: {altura:.2f} m")  # 2 decimales
print(f"Edad: {edad:03d}")        # Rellenar con ceros (025)

# Alineación
print(f"{nombre:>10}")  # Derecha (    Ana)
print(f"{nombre:<10}")  # Izquierda (Ana    )
print(f"{nombre:^10}")  # Centro (  Ana   )
```

format() - Método clásico

```python
nombre = "Ana"
edad = 25

print("Me llamo {} y tengo {} años".format(nombre, edad))
print("Me llamo {0} y tengo {1} años".format(nombre, edad))
print("Me llamo {n} y tengo {e} años".format(n=nombre, e=edad))
```

% formatting - Estilo antiguo

```python
nombre = "Ana"
edad = 25

print("Me llamo %s y tengo %d años" % (nombre, edad))
```

Validación de Strings

Verificar tipo de contenido

python

```python
# isdigit() - Solo dígitos
print("12345".isdigit())     # True
print("123a5".isdigit())     # False

# isalpha() - Solo letras
print("Python".isalpha())    # True
print("Python3".isalpha())   # False

# isalnum() - Letras y números
print("Python3".isalnum())   # True
print("Python 3".isalnum())  # False

# isspace() - Solo espacios
print("   ".isspace())       # True
print("  a  ".isspace())     # False

# isupper() / islower()
print("PYTHON".isupper())    # True
print("python".islower())    # True

# istitle()
print("Hola Mundo".istitle()) # True
print("Hola mundo".istitle()) # False
```

Substrings y Slicing Avanzado

python

```python
mensaje = "Aprendiendo Python paso a paso"

# Extraer palabras
print(mensaje[0:11])   # "Aprendiendo"
print(mensaje[12:18])  # "Python"
print(mensaje[-4:])    # "paso"

# Slicing con paso
print(mensaje[::2])    # Cada 2 caracteres
print(mensaje[::-1])   # Invertir string

# Invertir palabras
palabras = mensaje.split()
invertido = " ".join(palabras[::-1])
print(invertido)  # "paso a paso Python Aprendiendo"
```

Recorrer un String

Con for

```python
texto = "Python"

# Recorrer caracteres
for letra in texto:
    print(letra)

# Con enumerate
for i, letra in enumerate(texto):
    print(f"Índice {i}: {letra}")
```

Con while

```python
texto = "Python"
i = 0

while i < len(texto):
    print(texto[i])
    i += 1
```

Propiedades Útiles

```python
texto = "Hola Mundo"

# Longitud
print(len(texto))  # 10

# Máximo y mínimo (orden ASCII)
print(max(texto))  # "u"
print(min(texto))  # " " (espacio)

# Contar longitud sin espacios
sin_espacios = texto.replace(" ", "")
print(len(sin_espacios))  # 9
```

Casos de Uso Prácticos

Validar contraseña

python

```python
password = "MiPass123"

tiene_mayuscula = any(c.isupper() for c in password)
tiene_minuscula = any(c.islower() for c in password)
tiene_numero = any(c.isdigit() for c in password)
longitud_valida = len(password) >= 8

valida = all([tiene_mayuscula, tiene_minuscula, tiene_numero, longitud_valida])
print(f"Contraseña válida: {valida}")
```

Censu