II. Variables y tipos de datos: tipos numéricos, cadenas, booleanos (2 horas)

A. Variables y tipos de datos

1. Variables y asignación
    - Ejemplo:

```
x = 5
nombre = "Juan"
_edad = 30
```

2. Tipos de datos en Python
    - Ejemplo:

```
entero = 42
flotante = 3.14
cadena = "Hola, mundo!"
booleano = True
```

3. Conversión de tipos
    - Ejemplo:

```
entero_a_float = float(42)
float_a_str = str(3.14)
str_a_bool = bool("True")
```

B. Tipos numéricos

1. Enteros
    - Ejemplo:

```
decimal = 42
binario = 0b101010
octal = 0o52
hexadecimal = 0x2A
```

2. Flotantes
    - Ejemplo:

```
flotante_decimal = 3.14
flotante_exponencial = 3.14e-2
```

3. Operaciones numéricas
    - Ejemplo:

```
suma = 5 + 3
resta = 7 - 2
multiplicacion = 4 * 6
division = 10 / 3
exponente = 2 ** 3
modulo = 9 % 2
```

C. Cadenas

1. Creación de cadenas
    - Ejemplo:

```
cadena_simple = 'Hola, mundo!'
cadena_doble = "Hola, mundo!"
```

2. Operaciones con cadenas
    - Ejemplo:

```
concatenacion = "Hola, " + "mundo!"
repeticion = "Hola! " * 3
primer_caracter = "Python"[0]
```

3. Métodos de cadenas
    - Ejemplo:

```
minusculas = "HOLA, MUNDO!".lower()
mayusculas = "hola, mundo!".upper()
sin_espacios = " Hola, mundo! ".strip()
lista_de_palabras = "Hola, mundo!".split()
cadena_unida = ", ".join(["Hola", "mundo"])
```

D. Booleanos

1. Definición de booleanos
    - Ejemplo:

```
verdadero = True
falso = False
resultado = 3 > 1
```

2. Operaciones lógicas
    - Ejemplo:

```
and_result = True and False
or_result = True or False
not_result = not True
```

3. Condiciones y estructuras de control de flujo
    - Ejemplo:

```
if 5 > 3:
    print("5 es mayor que 3")
elif 3 > 5:
    print("3 es mayor que 5")
else:
    print("5 y 3 son iguales")

for i in range(3):
    print(i)

contador = 0
while contador < 3:
    print(contador)
    contador += 1
```

Siguiendo con las notas para la clase, abordaremos más temas relacionados con estructuras de datos y funciones en Python:

E. Listas y tuplas

1. Creación de listas y tuplas
    - Ejemplo:

```
lista = [1, 2, 3, 4]
tupla = (1, 2, 3, 4)
```

2. Operaciones con listas y tuplas
    - Ejemplo:

```
lista.append(5)
lista.extend([6, 7, 8])
lista.pop()
lista.remove(2)
lista.insert(0, 0)
lista.sort()
lista.reverse()
elemento_tupla = tupla[1]
```

F. Diccionarios

1. Creación de diccionarios
    - Ejemplo:

```
diccionario = {"clave": "valor", "nombre": "Juan", "edad": 30}
```

2. Operaciones con diccionarios
    - Ejemplo:

```
valor = diccionario["clave"]
diccionario["nueva_clave"] = "nuevo_valor"
diccionario.update({"nombre": "Pedro", "altura": 175})
del diccionario["clave"]
diccionario_keys = diccionario.keys()
diccionario_values = diccionario.values()
diccionario_items = diccionario.items()
```

G. Conjuntos

1. Creación de conjuntos
    - Ejemplo:

```
conjunto = {1, 2, 3, 4}
```

2. Operaciones con conjuntos
    - Ejemplo:

```
conjunto.add(5)
conjunto.remove(2)
union = conjunto | {4, 5, 6, 7}
interseccion = conjunto & {4, 5, 6, 7}
diferencia = conjunto - {4, 5, 6, 7}
```

H. Funciones

1. Definición de funciones
    - Ejemplo:

```
def suma(a, b):
    return a + b
```

2. Llamada a funciones
    - Ejemplo:

```
resultado = suma(3, 4)
print(resultado)
```

3. Argumentos por defecto y argumentos con nombre
    - Ejemplo:

```
def saludo(nombre, saludo="Hola"):
    return f"{saludo}, {nombre}!"

mensaje = saludo("Juan", saludo="Buenos días")
print(mensaje)
```

Estas notas adicionales cubren temas como listas, tuplas, diccionarios, conjuntos y funciones en Python. Estas estructuras de datos y funciones son fundamentales para desarrollar programas más complejos y eficientes. Además, se incluyen ejemplos de código para ilustrar cómo crear, manipular y utilizar estos elementos en Python.