PREGRADO

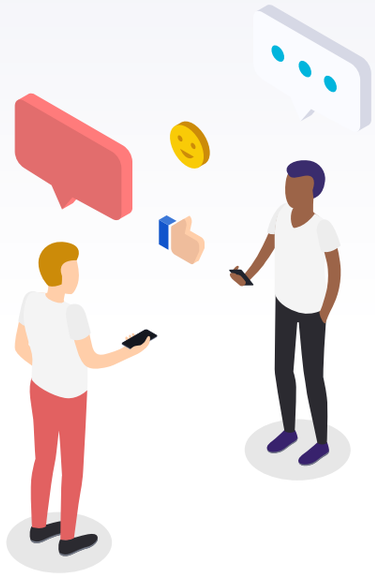
UNIDAD 1 | SEMANA 4

Diccionarios

1ACC0201| Programación Orientada a Objetos

Logro

Al final de la sesión el alumno entiende como usar diccionarios en Python



Diccionarios

- Los **diccionarios** son la colección de datos más poderosa de Python
- Un diccionario es una colección que está ordenada, es cambiable y no permite llaves duplicadas.
- Los **diccionarios** se utilizan para almacenar valores de datos en pares **clave: valor**.
- A partir de la versión 3.7 de Python, los diccionarios están ordenados.
- Que los **diccionarios** están ordenados, significa que **los elementos tienen un orden definido**, y ese orden no cambiará.
- Los diccionarios se escriben usando **llaves {},** y tienen **claves y valores**.
- Los **diccionarios** nos permiten realizar operaciones rápidas
- Las **claves** pueden ser cualquier tipo de datos inmutable: **strings, enteros, flotantes, tuplas**

Creando Diccionarios

- Se puede crear un diccionario como muestra el ejemplo

```
# Definiendo un diccionario
auto = {"marca": "Ford", "modelo": "Mustang", "año": 1964}
print(auto)
```

```
{'marca': 'Ford', 'modelo': 'Mustang', 'año': 1964}
```

- Se puede acceder a un valor del diccionario a través de su llave

```
# Accesando un valor por su clave
auto = {"marca": "Ford", "modelo": "Mustang", "año": 1964}
print(auto["marca"])
```

```
Ford
```

- La función len() devuelve la cantidad de pares clave: valor de un diccionario

```
# Longitud de un diccionario
auto = {"marca": "Ford", "modelo": "Mustang", "año": 1964}
print(len(auto))
```

```
3
```

Hay tres pares clave: valor

Elementos de un Diccionario – Tipos de datos

- Veamos el siguiente ejemplo

```
auto = {
    "marca": "Ford",
    "modelo": "Mustang",
    "año": 1964,
    "colores": ["rojo", "azul", "negro"],
}

print(auto["colores"][::-1])
```

```
['negro', 'azul', 'rojo']
```

- Analice y discute el resultado del código

Adicionando datos a un Diccionario

- Veamos el siguiente ejemplo

```
# Adiciones a un diccionario
auto = {
    "marca": "Ford",
    "modelo": "Mustang",
    "año": 1964,
    "colores": ["rojo", "azul", "negro"],
}
auto["cilindrada"] = 3000
print(auto)
auto["cilindrada"] = 2000
print(auto)
auto["colores"][1] = "azul perlado"
print(auto)
```

- Si la clave no existe, se crea la entrada al diccionario
- Si ya existe la clave, se cambia el valor correspondiente a la clave
- Analice y discuta la línea donde aparece el “azul perlado”

```
{'marca': 'Ford', 'modelo': 'Mustang', 'año': 1964, 'colores': ['rojo', 'azul', 'negro'], 'cilindrada': 3000}
{'marca': 'Ford', 'modelo': 'Mustang', 'año': 1964, 'colores': ['rojo', 'azul', 'negro'], 'cilindrada': 2000}
{'marca': 'Ford', 'modelo': 'Mustang', 'año': 1964, 'colores': ['rojo', 'azul perlado', 'negro'], 'cilindrada': 2000}
```

Obteniendo valores y llaves de los diccionarios

- El método *values()*

```
# Obteniendo los valores de un diccionario
auto = {
    "marca": "Ford",
    "modelo": "Mustang",
    "año": 1964,
    "colores": ["rojo", "azul", "negro"],
}

lista1 = list(auto.values())
print(lista1)
```

```
['Ford', 'Mustang', 1964, ['rojo', 'azul', 'negro']]
```

- Este método devuelve una lista con todos los valores del diccionario

Obteniendo valores y llaves de los diccionarios

- El método ***keys()***

```
# Obteniendo las llaves de un diccionario
auto = {
    "marca": "Ford",
    "modelo": "Mustang",
    "año": 1964,
    "colores": ["rojo", "azul", "negro"],
}

lista1 = list(auto.keys())
print(lista1)
```

```
['marca', 'modelo', 'año', 'colores']
```

- Este método devuelve una lista con todas las llaves del diccionario

Obteniendo valores y llaves de los diccionarios

- El método ***items()***

```
# Obteniendo los llaves y valores de un
diccionario
auto = {
    "marca": "Ford",
    "modelo": "Mustang",
    "año": 1964,
    "colores": ["rojo", "azul", "negro"],
}

lista1 = list(auto.items())
print(lista1)
```

- Este método devuelve una lista con tuplas formadas por las llaves y sus valores

```
[('marca', 'Ford'), ('modelo', 'Mustang'), ('año', 1964), ('colores', ['rojo', 'azul', 'negro'])]
```

Recorriendo Diccionarios

- Se pueden por tanto recorrer tanto las llaves como los valores de un diccionario con ***in, keys(), values(), items()***

```
auto = {
    "marca": "Ford",
    "modelo": "Mustang",
    "año": 1964,
    "colores": ["rojo", "azul", "negro"],
}
for llaves in auto:
    print(f"{llaves}")

print()
for llaves in auto.keys():
    print(f"{llaves}")

print()
for valores in auto.values():
    print(f"{valores}")

print()
for llaves, valores in auto.items():
    print(f"{llaves} {valores}")
```

```
marca
modelo
año
colores

marca
modelo
año
colores

Ford
Mustang
1964
['rojo', 'azul', 'negro']

marca Ford
modelo Mustang
año 1964
colores ['rojo', 'azul', 'negro']
```

Analice y discuta los resultados

La función dict()

- Se pueden diccionarios usando la función ***dict()***

```
auto = dict(marca="Ford",
            modelo="Mustang",
            año=1964,
            colores=["rojo", "azul", "negro"])

print(auto)
```

```
{'marca': 'Ford', 'modelo': 'Mustang', 'año': 1964, 'colores': ['rojo', 'azul', 'negro']}
```

- ***dict()*** permite la creación de diccionarios vacíos

```
cuentas = dict()
```

El método get()

- Devuelve el valor para una clave específica, permitiendo especificar un valor alternativo si la llave no existe.
- Ejemplo: tenemos una lista con nombres y se quiere crear un diccionario teniendo como llave es esos nombres y como valor las veces que aparecen. Veamos y analicemos el código

```
recuentos = dict()
nombres = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for nombre in nombres:
    recuentos[nombre] = recuentos.get(nombre, 0) + 1
print(recuentos)
```

```
{'csev': 2, 'cwen': 2, 'zqian': 1}
```

- Se crea un diccionario vacio
- Se recorre la lista nombres
- Cada nombre se usa como llave del diccionario y se le asigna su valor incrementado en 1
- El método get(), si no encuentra la llave, retorna 0 porque así se ha especificado en el get() y le adiciona 1 y graba ese resultado como el valor
- Si la llave se encuentra, get() devuelve el valor y le suma 1 y con ese resultado actualiza el valor correspondiente a la llave

Usando diccionarios para contar el número de veces que una palabra aparece en un texto

- Usando el ejemplo anterior, se desarrolla un programa para que cuente cuantas veces aparece cada palabra en el texto

```
texto = """
el payaso corrió detrás del coche y el coche se metió en la tienda y
la tienda cayó sobre el payaso y el coche
"""
recuento = dict()
palabras = texto.split()  # la función split() crea una lista con las palabras
del texto

print("Palabras:", palabras)

print(“\nContando...”)
for palabra in palabras:
    recuento[palabra] = recuento.get(palabra, 0) + 1
print('Recuento', recuento)
```

Palabras: ['el', 'payaso', 'corrió', 'detrás', 'del', 'coche', 'y', 'el', 'coche', 'se', 'metió', 'en', 'la', 'tienda', 'y', 'la', 'tienda', 'cayó', 'sobre', 'el', 'payaso', 'y', 'el', 'coche']

Contando...
Recuento {'el': 4, 'payaso': 2, 'corrió': 1, 'detrás': 1, 'del': 1, 'coche': 3, 'y': 3, 'se': 1, 'metió': 1, 'en': 1, 'la': 2, 'tienda': 2, 'cayó': 1, 'sobre': 1}

Diccionarios dentro de diccionarios

- Los valores de un diccionario, pueden ser diccionarios

```
alumnos = {
    "alumno1": {"Fundamentos en programación": 13, "OOP": 18},
    "alumno2": {"Fundamentos en programación": 15, "OOP": 15},
    "alumno3": {"Fundamentos en programación": 16, "OOP": 19},
}
print(alumnos)
```

```
{'alumno1': {'Fundamentos en programación': 13, 'OOP': 18}, 'alumno2': {'Fundamentos en programación': 15, 'OOP': 15}, 'alumno3': {'Fundamentos en programación': 16, 'OOP': 19}}
```

Eliminando elementos de un diccionario

- El método ***pop()***

```
alumnos = {
    "alumno1": {"Fundamentos en programación": 13, "OOP": 18},
    "alumno2": {"Fundamentos en programación": 15, "OOP": 15},
    "alumno3": {"Fundamentos en programación": 16, "OOP": 19},
}
print(alumnos)
alumnos.pop("alumno2")
print()
print(alumnos)
```

{'alumno1': {'Fundamentos en programación': 13, 'OOP': 18}, 'alumno2': {'Fundamentos en programación': 15, 'OOP': 15}, 'alumno3': {'Fundamentos en programación': 16, 'OOP': 19}}

{'alumno1': {'Fundamentos en programación': 13, 'OOP': 18}, 'alumno3': {'Fundamentos en programación': 16, 'OOP': 19}}

Juntando diccionarios

- El método ***update()***

```
alumnos = {
    "alumno1": {"Fundamentos en programación": 13, "OOP": 18},
    "alumno2": {"Fundamentos en programación": 15, "OOP": 15},
    "alumno3": {"Fundamentos en programación": 16, "OOP": 19},
}
print(alumnos)
alumnos.pop("alumno2")
print()
print(alumnos)
print()
reg_auxi = {
    "alumno4": {"Fundamentos en programación": 14, "OOP": 17},
    "alumno5": {"Fundamentos en programación": 17, "OOP": 13.5}
}
alumnos.update(reg_auxi)
print(alumnos)
```

{'alumno1': {'Fundamentos en programación': 13, 'OOP': 18}, 'alumno2': {'Fundamentos en programación': 15, 'OOP': 15}, 'alumno3': {'Fundamentos en programación': 16, 'OOP': 19}}

{'alumno1': {'Fundamentos en programación': 13, 'OOP': 18}, 'alumno3': {'Fundamentos en programación': 16, 'OOP': 19}}

{'alumno1': {'Fundamentos en programación': 13, 'OOP': 18}, 'alumno3': {'Fundamentos en programación': 16, 'OOP': 19}, 'alumno4': {'Fundamentos en programación': 14, 'OOP': 17}, 'alumno5': {'Fundamentos en programación': 17, 'OOP': 13.5}}

PREGRADO

Ingeniería de Sistemas de Información

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC

Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco

Lima 33 - Perú

T 511 313 3333

https://www.upc.edu.pe

exígete, innova

LAUREATE
INTERNATIONAL
UNIVERSITIES