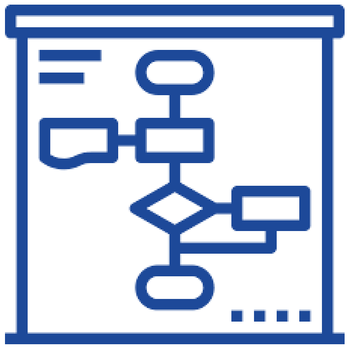
INTRODUCCIÓN A LA PROGRAMACIÓN



OTOÑO, 2021

UNIVERSIDAD TECNOLÓGICA DE CHILE
INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA
nacap

UNIDAD III

Arreglos, tuplas y diccionarios

{*Diccionarios*}

Conceptos Generales

Diccionarios

- Un diccionario es un tipo de datos que sirve para asociar **pares de objetos**.
- Un diccionario puede ser visto como una colección de **claves**, cada una de las cuales tiene asociada un **valor**.
- Los diccionarios están ordenados y no hay claves repetidas (duplicadas).
- Los diccionarios se pueden cambiar, lo que significa que podemos cambiar, agregar o eliminar elementos después de que se haya creados.
- La única manera de acceder a un valor es a través de su clave.
- Desde la perspectiva de Python, los diccionarios se definen como objetos con el tipo de datos 'diccionario': <class 'dict'>.

```
laboratorios = {
  "lab1": 204,
  "lab2": 205,
  "lab3": 208
}
```

Nota: Un diccionario vacío puede ser creado usando {} o con la función **dict()**:

```
d = {}
d = dict()
```

También es posible crear listas de claves o valores:

```
x=list(telefonos)
x=list(telefonos.values())
```

Crear un diccionario

Los diccionarios literales se crean usando llaves { y }. La clave y el valor van separados por dos puntos:

```
telefonos = {'Pancho': 95552437, 'Javier': 95551428, 'Sandra': 95550012}
print(telefonos)
#salida {'Pancho': 95552437, 'Javier': 95551428, 'Sandra': 95550012}
```

En este ejemplo, las claves son 'Pancho', 'Javier' y 'Sandra', y los valores asociados a ellas son, respectivamente, 95552437, 95551428 y 95550012.

Acceder a elementos del diccionario

Puede acceder a los elementos de un diccionario haciendo referencia a su nombre clave, entre corchetes; de este modo, el valor asociado a la clave k en el diccionario d se puede obtener mediante **d[k]**:

```
laboratorios = {
    "lab1": 204,
    "lab2": 205,
    "lab3": 208
}
x=laboratorios[‘lab2']
print(x)
#salida 205
```

También hay un método llamado **get()** que le dará el mismo resultado:

```
x=laboratorios.get('lab2')
```

```
telefonos = {'Pancho': 95552437, 'Javier': 95551428, 'Sandra': 95550012}
print(telefonos[‘Pancho’])
#salida 95552437
```

Nota: A diferencia de los índices de las listas, las claves de los diccionarios no necesitan ser números enteros.
Si la clave no está presente en el diccionario, ocurre un error de llave (`KeyError`):

Operaciones con diccionarios

Agregar elementos

Se puede agregar una clave nueva simplemente asignándole un valor:

```
telefonos = {'Pancho': 95552437, 'Javier': 95551428, 'Sandra': 95550012}
telefonos['Claudia']=95577001
print(teléfonos)
#salida {'Pancho': 95552437, 'Javier': 95551428, 'Sandra': 95550012, 'Claudia': 95577001}
```

Si se asigna un valor a una llave que ya estaba en el diccionario, el valor anterior se **sobrescribe**. Recuerde que un diccionario no puede tener claves repetidas:

```
telefonos = {'Pancho': 95552437, 'Javier': 95551428, 'Sandra': 95550012}
telefonos[‘Pancho’]=95550012
print(teléfonos)
#salida {'Pancho': 95550012, 'Javier': 95551428, 'Sandra': 95550012}
```

Operaciones con diccionarios

Obtener claves

El método **keys()** devolverá una lista de todas las claves del diccionario.

La lista de claves es una vista del diccionario, lo que significa que cualquier cambio realizado en el diccionario se reflejará en la lista de claves.
Si agrega un nuevo elemento al diccionario original, la lista de claves también se actualiza.

```
telefonos = {'Pancho': 95552437, 'Javier': 95551428, 'Sandra': 95550012}
print(telefonos.keys())
#salida dict_keys(['Pancho', 'Javier', 'Sandra'])
```

Nota: El método **items()** devolverá cada elemento en un diccionario, como tuplas en una lista.

```
print(telefonos.items())
```

Operaciones con diccionarios

Actualizar diccionario

El método **update()** actualizará el diccionario con los elementos del argumento dado. El argumento debe ser un diccionario o un objeto iterable con pares clave: valor.

```
laboratorios = {
    'lab1': 204,
    'lab2': 205,
    'lab3': 208
}
laboratorios.uptade({‘lab2’: 210})
```

Borrar claves

Para borrar una clave, se puede usar la sentencia **del**:

```
del telefonos['Pancho’]
print(telefonos)
#salida {'Javier': 95551428, 'Sandra': 95550012}
```

Eliminar elementos

El método **pop()** elimina el elemento con el nombre de clave especificado:

```
laboratorios.pop('lab2')
```

Vaciar el diccionario

El método **clear()** vacía el diccionario:

```
laboratorios.clear()
```

Recorrer un diccionario

Puede recorrer un diccionario utilizando un bucle **for**. Al recorrer un diccionario, el valor de retorno son las claves del diccionario, pero también existen métodos para devolver los valores, por tanto, los diccionarios son iterables.

Encontrar un elemento

k in d permite saber si la llave k está en el diccionario d:

```
ruedas={'auto': 4, 'bicicleta': 2, 'triciclo': 3}
x='auto' in ruedas
print(x)
#salida True
```

Iterar sobre un diccionario

Al iterar sobre un diccionario se obtienen las claves:

```
for x in ruedas:
    print(x)
#salida
auto
bicicleta
triciclo
```

```
for x in ruedas.keys():
    print(x)
```

Iterar sobre las claves

Al iterar sobre las claves se obtienen los valores:

```
for x in ruedas.values():
    print(x)
#salida
4
3
2
```

```
for x in ruedas:
    print(ruedas[x])
```

Para iterar sobre las llaves y los valores simultáneamente, se usa el método d.items()

```
for x,y in ruedas.items():
    print("Ruedas",x,"=",y)
```

Ejercicios

1. Escriba la función contar_letras(oracion) que retorne un diccionario asociando a cada letra la cantidad de veces que aparece en la oración:

```
contar_letras('El elefante avanza hacia Asia')
{'a': 8, 'c': 1, 'e': 4, 'f': 1, 'h': 1, 'i': 2, 'l': 2, 'n': 2, 's': 1, 't': 1, 'v': 1, 'z': 1}
```

Cada valor del diccionario debe considerar tanto las apariciones en mayúscula como en minúscula de la letra correspondiente. Los espacios deben ser eliminados.

2. Escriba una función promedio(num) que entregue el promedio de los valores de todas las claves:

```
n = {2:2, 8:7, 4:2, 6:4}
El promedio de las claves es:  5.0
```

3. Escriba una función maximo_par(j) que entregue el valor máximo de la suma de una clave y su valor en el diccionario j:

```
j = {5:1, 4:7, 9:0, 2:2}
Resultado: 11
```

Nota complementaria: Para obtener la lista de palabras de la oración, puede usar el método split de los strings:

```
s = 'el gato y el pato'
s.split()
['el', 'gato', 'y', 'el', 'pato']
```

inacap.cl