LECTURE 13

DOHYUNG KIM

WHAT IS DISCUSSED IN THE LAST CLASS

- Set

TODAY, WE WILL LEARN ABOUT

- Dictionary

DICTIONARY

- An unordered collection of data
- Used to store data like a map
  - Other data types hold a single value as an element
  - Dictionary holds **key:value** pair
- You can access **value** of a dictionary by referring to its **key** name, inside [ ]

```
carDic = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(carDic)
print("carDic[model]:", carDic["model"])
print("carDic[year]:", carDic["year"])
```

CREATING DICTIONARY

- Creating an empty dictionary

```
d1 = dict()
print(d1)

d2 = {}
print(d2)
```

- Creating a dictionary from a list of (key, value) pairs

```
scores = [("math", 100), ("physics", 99), ("history", 70)]
d = dict(scores)
print(d)  #unordered
```

- Statically-allocate a dictionary

```
scoreDic = {"math":100, "physics":99, "history":70}
print(scoreDic)
```

ADDING DATA TO DICTIONARY

- **d [key] = value**
  - Dictionaries map keys to values

```
employee = dict()
key = "Albert"
info = 20200001, "male"
employee[key] = info

key = "Cathy"
info = 20200002, "female"
employee[key] = info

print(employee)
```

ACCESSING DATA IN DICTIONARY

- **d [key]**

```
employee = dict()
key = "Albert"
info = 20200001, "male"
employee[key] = info

key = "Cathy"
info = 20200002, "female"
employee[key] = info

print("Albert's info :", employee["Albert"])
print("Albert's id :", employee["Albert"][0])
print("Cathy's gender :", employee["Cathy"][1])
```

KEYS IN DICTIONARY

- Keys are unique

```
d = dict()
d["Albert"] = 3
d["Albert"] = 2
d["Albert"] = 1
print(d)
```

- Keys must be immutable

```
d = dict()
a = [1]   # what if a list becomes a key
d[a] = 42 # Error: unhashable type: 'list'
```

KEYS IN DICTIONARY

- **d.keys()**
  - dict_keys object is return
  - iterable

- cf) **d.items() / d.values()**

```
d = {"Albert": 100, "Cathy":90, "Brown":80}
print("d :", d)
print()
print("keys in d :", d.keys())
print()

print("keys : ", end="")
for i in d.keys():
  print(i, end=" ")
print("\n")

print("items : ", end="")
for i in d.items():
  print(i, end=" ")
print("\n")

print("values : ", end="")
for i in d.values():
  print(i, end=" ")
```

KEYS IN DICTIONARY

- Keys in list or tuple

```
d = {"Albert": 100, "Cathy":90, "Brown":80}
print("keys in d :", d.keys())
print("\n * keys in list")
keys_list1 = [*d]
print(keys_list1)
keys_list2 = list(d)
print(keys_list2)
keys_list3 = list(d.keys())
print(keys_list3)

print("\n * keys in tuple")
keys_tuple1 = tuple(d)
print(keys_tuple1)
keys_tuple2 = tuple(d.keys())
print(keys_tuple1)
```

**** Notes!!
in the expression [*d], * unpacks the container

LOOP OVER DICTIONARY

- Using keys() or items()

```
d = {"Albert": 100, "Cathy":90, "Brown":80}
print("d :", d)
print("keys in d :", d.keys())

for val in d.keys():
  print("d[%s]: %d " % (val, d[val]))

print()
for val in d.items():
  print("d[%s]: %d " % (val[0], val[1]))
```

- You may see the order of keys : Albert, Cathy, Brown..
- How can we print items in the alphabetical order of keys?

SOLUTION

- Using sorted() function learned in previous class

```
d = {"Albert": 100, "Cathy":90, "Brown":80}
print("d :", d)

for key in d.keys():
  print("d[%s]: %d " %(key, d[key]))

print()
for key in sorted(list(d)):
  print("d[%s]: %d " %(key, d[key]))

print()
for key in sorted(list(d), reverse = True):
  print("d[%s]: %d " %(key, d[key]))
```



```
for key in list(d).sort:
```

- Can we use list(d).sort() instead of sorted(list(d))?

OPERATIONS ON DICTIONARY

- **len()**

```
d = { 1:[1,2,3,4,5], 2:"abcd" }
print(len(d))
```

- **d.copy()**

```
d1 = { 1:"Albert" }
d2 = d1.copy()
d1[2] = "Cathy"
print(d1)
print(d2)
```

- **d.clear()**

```
d = { 1:"Albert", 2:"Cathy" }
d.clear()
print(d, len(d))
```

OPERATIONS ON DICTIONARY

- key **in** d / key **not in** d

```
d = { 1:"Albert", 2:"Cathy" }
print(0 in d)
print(1 in d)
print("Albert" in d)

print(0 not in d)
print(1 not in d)
print("Albert" not in d)
```

- **d.get**(key [, default])

```
d = { 1:"Albert", 2:"Cathy" }
print(d.get(1))
print(d.get(1) == d[1])
print(d.get(0))
print(d)
```

OPERATIONS ON DICTIONARY

- **del d[key]**

```
d = { 1:"Albert", 2:"Cathy" }
print(1 in d)
del d[1]
print(1 in d)
print("d[2]:", d[2])
print("d[1]:", d[1])
```

- **d1.update(d2)**

```
d1 = { 1:"Albert", 2:"Cathy" }
d2 = { 2:"John", 3:"Simon" }
d1.update(d2)
print("d1 =", d1)
print("d2 =", d2)
```

OPERATIONS ON DICTIONARY

- You can refer to the following link for more operations on a dictionary
  - https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

QUESTION?