migo_7_dict&others

June 20, 2024

Dictionaries

collection of key-value pairs. Each key is unique(string or int) and is used to store and retrieve corresponding values.

Unordered,Mutable,Indexed (by keys),Dynamic

Creating a Dictionary

[1]:
```
my_dict = {}
st_details = {"id":"003456",'name': 'Lala', 'age': 25, 'city': 'Eilat'}
st_dict={"details":st_details,"grades":{"Math":[90,85,79],"Python":
  ↪[99,100,100,85]}}
print(my_dict)
print(st_details)
print(st_dict)
```

```
{}
{'id': '003456', 'name': 'Lala', 'age': 25, 'city': 'Eilat'}
{'details': {'id': '003456', 'name': 'Lala', 'age': 25, 'city': 'Eilat'},
'grades': {'Math': [90, 85, 79], 'Python': [99, 100, 100, 85]}}
```

Accessing and Modifying Values:

[2]:
```
st_details["name"]="lolo"
print(st_details["name"])
print(st_dict["grades"]["Math"])
print(st_dict["grades"]["Python"][-1])
```

```
lolo
[90, 85, 79]
85
```

Adding Items:

[3]:
```
st_details["subject"]="Programming"
print(st_dict)
```

```
{'details': {'id': '003456', 'name': 'lolo', 'age': 25, 'city': 'Eilat',
'subject': 'Programming'}, 'grades': {'Math': [90, 85, 79], 'Python': [99, 100,
100, 85]}}
```

Removing Items

[4]:
```
del(st_dict["details"])
del(st_dict["grades"]["Math"])
print(st_dict)
print(st_details.pop("age"))
print(st_details)
st_dict["details"]=st_details
st_dict["grades"]["Math"]=[99,99,99]
```

```
{'grades': {'Python': [99, 100, 100, 85]}}
25
{'id': '003456', 'name': 'lolo', 'city': 'Eilat', 'subject': 'Programming'}
```

Iterating through a Dictionary

[5]:
```
# Iterating through keys
for key in st_dict:
    print(key)

# Iterating through values
for value in st_dict.values():
    print(value)

# Iterating through key-value pairs
for key, value in st_dict.items():
    print(key, value)
```

```
grades
details
{'Python': [99, 100, 100, 85], 'Math': [99, 99, 99]}
{'id': '003456', 'name': 'lolo', 'city': 'Eilat', 'subject': 'Programming'}
grades {'Python': [99, 100, 100, 85], 'Math': [99, 99, 99]}
details {'id': '003456', 'name': 'lolo', 'city': 'Eilat', 'subject':
'Programming'}
```

Dictionary Methods: keys(),values(),items(),get(key, default)

[11]:
```
print(st_details.keys())
print(st_details.values())
print(st_details.items())
print(st_details.get("id"))
print(st_details.get("firstName","default"))
```

```
dict_keys(['id', 'name', 'city', 'subject'])
dict_values(['003456', 'lolo', 'Eilat', 'Programming'])
dict_items([('id', '003456'), ('name', 'lolo'), ('city', 'Eilat'), ('subject',
'Programming')])
003456
default
```

Tuples

collection of items

Ordered,Immutable,Allows Duplicates,Heterogeneous-can contain different data types.

Creating tuples

[12]:
```
my_tuple = ()
my_tuple = (1, 2, 3, 'apple')
# Tuple without parentheses (not recommended)
my_tuple = 1, 2, 3, 'apple'
single_element_tuple = (1,)
```

Accessing Elements

[13]:
```
print(my_tuple[1])
```

```
2
```

Unpacking & packing Tuples

[17]:
```
a,b,c,d=my_tuple
print(a,b,c,d)
tp=a,b,c,d
print(tp)
```

```
1 2 3 apple
(1, 2, 3, 'apple')
```

Methods

[18]:
```
tp=(1,2,3,2,2,4)
print(tp.count(2))
print(tp.index(2))
```

```
3
1
```

Sets

Unordered,Mutable,No Duplicates,No Indexing,Heterogeneous: Sets can contain different data types.

Creating set

[28]:
```
my_set = set()
# Set with initial values
my_set = {1, 2, 3, 'apple'}
```

add and remove

[29]:
```
my_set.add("banana")
my_set.remove(2)
```

```
print(my_set)
```

```
{1, 3, 'apple', 'banana'}
```

Operations: Union,Intersection,Difference,Symmetric Difference

[31]:
```
set1 = {1, 2, 3}
set2 = {3, 4, 5}

# Union
print(set1 | set2)  # Output: {1, 2, 3, 4, 5}

# Intersection
print(set1 & set2)  # Output: {3}

# Difference
print(set1 - set2)  # Output: {1, 2}

# Symmetric Difference
print(set1 ^ set2)  # Output: {1, 2, 4, 5}
```

```
{1, 2, 3, 4, 5}
{3}
{1, 2}
{1, 2, 4, 5}
```

bytearray & memoryview

[43]:
```
my_bytes = b"Hello"
my_bytearray = bytearray(b"Hello world!!!")
mv = memoryview(my_bytearray)

print(my_bytes)              # Output: b'Hello'
print(my_bytearray)          # Output: bytearray(b'Hello world!!!')
print(mv)                    # Output: <memory at 0x...> (memoryview
 ↪representation)

# Ensuring the slice length matches the replacement bytes length
mv[0:5] = b"lala "           # Replacing the first 5 bytes with 'lala!'
print(my_bytearray)
```

```
b'Hello'
bytearray(b'Hello world!!!')
<memory at 0x0000020B9D782440>
bytearray(b'lala  world!!!')
```

[ ]: