Set

A set in Python is an unordered, mutable, and unindexed collection of unique elements. It does not allow duplicate values.

Set Characteristics

- Unordered: Elements are not stored in a specific order.
- Unindexed: You cannot access elements using an index.
- Unique Elements: Duplicates are automatically removed.

Set Methods

1. add(): Adds an element to the set
2. update(): Updates the set with elements from another set
3. remove(): Removes a specific element (raises error if not found)
4. discard(): Removes a specific element (no error if not found)
5. pop(): Removes and returns a random element
6. copy(): Returns a shallow copy of the set
7. clear(): Removes all elements
8. union(): Returns the union of two sets
9. intersection(): Returns the intersection of two sets
10. difference(): Returns the difference of two sets
11. symmetric_difference(): Returns elements in either set, but not both
12. issubset(): Checks if set is a subset of another
13. issuperset(): Checks if set is a superset of another
14. isdisjoint(): Checks if two sets have no elements in common

In [ ]:
```
# Creating a Set
my_set = {1, 2, 3, 4, 5}
print(my_set)
```

{1, 2, 3, 4, 5}

In [ ]:
```
new_set = {10, 20, 30, 40, 50}
print(new_set)
```

{50, 20, 40, 10, 30}

In [ ]:
```
sample_set = {1, 2, 2, 3, 4, 4, 5}
print(sample_set  # Output: {1, 2, 3, 4, 5} (duplicates removed)
```

{1, 2, 3, 4, 5}

Set Methods

In [ ]:
```
# 1. add() - Adds an element to the set
s = {1, 2, 3}
s.add(4)
print(s)  # Output: {1, 2, 3, 4}
```

{1, 2, 3, 4}

In [ ]:
```
# 2. update() - Adds multiple elements from another iterable
s = {1, 2}
s.update([3, 4], {5, 6})
print(s)  # Output: {1, 2, 3, 4, 5, 6}
```

{1, 2, 3, 4, 5, 6}

In [ ]:
```
# 3. remove() - Removes a specific element (raises error if not found)
s = {1, 2, 3}
s.remove(2)
print(s)  # Output: {1, 3}
# s.remove(5) - This would raise KeyError if 5 is not in the set
```

{1, 3}

In [ ]:
```
# 4. discard() - Removes a specific element (no error if not found)
s = {1, 2, 3}
s.discard(2)
print(s)  # Output: {1, 3}
s.discard(5)  # No error
```

{1, 3}

In [ ]:
```
# 5. pop() - Removes and returns a random element
s = {10, 20, 30}
removed = s.pop()
print("Removed:", removed)
print("Remaining:", s)
```

Removed: 10
Remaining: {20, 30}

In [ ]:
```
# 6. copy() - Returns a shallow copy of the set
s1 = {1, 2, 3}
s2 = s1.copy()
print(s2)  # Output: {1, 2, 3}
```

{1, 2, 3}

In [ ]:
```
# 7. clear() - Removes all elements from the set
s = {1, 2, 3}
s.clear()
print(s)  # Output: set()
```

set()

In [ ]:
```
# 8. union() - Combines elements from both sets (no duplicates)
a = {1, 2, 3}
b = {3, 4, 5}
result = a.union(b)
print(result)  # Output: {1, 2, 3, 4, 5}
```

{1, 2, 3, 4, 5}

In [ ]:
```
# 9. intersection() - Returns common elements from both sets
a = {1, 2, 3}
b = {2, 3, 4}
result = a.intersection(b)
print(result)  # Output: {2, 3}
```

{2, 3}

In [ ]:
```
# 10. difference() - Elements in one set but not in the other
a = {1, 2, 3}
b = {2, 3, 4}
result = a.difference(b)
print(result)  # Output: {1}
```

{1}

In [ ]:
```
# 11. symmetric_difference() - Elements in either set but not both
a = {1, 2, 3}
b = {3, 4, 5}
result = a.symmetric_difference(b)
print(result)  # Output: {1, 2, 4, 5}
```

{1, 2, 4, 5}

In [ ]:
```
# 12. issubset() - Checks if all elements in one set are in another
a = {1, 2}
b = {1, 2, 3}
print(a.issubset(b))  # Output: True
```

True

In [ ]:
```
# 13. issuperset() - Checks if a set contains all elements of another set
a = {1, 2, 3, 4}
b = {1, 2}
print(a.issuperset(b))  # Output: True
```

True

In [ ]:
```
# 14. isdisjoint() - Checks if sets have no elements in common
a = {1, 2}
b = {3, 4}
print(a.isdisjoint(b))  # Output: True
```

True

Dictionary

A dictionary is a collection that is ordered, changeable, and does not allow duplicates.

In [ ]:
```
thisdict = {
  "model": "Mercedes",
  "color": "black",
  "year": 1964
}
print(thisdict)
# Accessing Items
print(thisdict["model"])
print(thisdict["color"])
print(thisdict["year"])
```

{'model': 'Mercedes', 'color': 'black', 'year': 1964}
Mercedes
black
1964

In [33]:
```
# Adding New Items
laptop = {
  "brand": "Dell",
  "model": "XPS 13",
  "year": 2022
}
laptop["processor"] = "Intel i7"
print(laptop)
```

{'brand': 'Dell', 'model': 'XPS 13', 'year': 2022, 'processor': 'Intel i7'}

In [ ]:
```
# Getting All Keys
newDict = {
  "Student_Id": 21,
  "Student_Name": "John",
  "Student_Department": "Software"
}
print(newdict.keys())

# Getting All Values
print(newdict.values())

# Getting All Items
print(newdict.items())

dict_keys(['Student_Id', 'Student_Name', 'Student_Department'])
dict_values([21, 'John', 'Software'])
dict_items([('Student_Id', 21), ('Student_Name', 'John'), ('Student_Department', 'Software')])
```

In [ ]:
```
# Checking if a Key Exists
thisdict = {
  "Jan": 31,
  "Feb": 28,
  "Mar": 31,
  "Apr": 30
}

if "Feb" in thisdict:
  print("Yes, 'Feb' is one of the keys in the thisdict1 dictionary")
```

Yes, 'Feb' is one of the keys in the thisdict1 dictionary

In [ ]:
```
# Changing Values
country_capital = {
  "Pakistan": "Karachi",
  "France": "Paris",
  "Japan": "Tokyo",
  "Brazil": "Brasilia",
  "Canada": "Ottawa"
}
country_capital["Pakistan"] = "Islamabad"
print(country_capital)
```

{'Pakistan': 'Islamabad', 'France': 'Paris', 'Japan': 'Tokyo', 'Brazil': 'Brasilia', 'Canada': 'Ottawa'}

In [ ]:
```
# Removing an Item (del)
book_info = {
  "title": "Harry Potter and the Sorcerer's Stone",
  "author": "J.K. Rowling",
  "year": 1997
}
del book_info["year"]
print(book_info)
```

{'title': 'Harry Potter and the Sorcerer's Stone', "author": "J.K. Rowling"}

In [ ]:
```
# Looping Through Keys
laptop_info = {
  "brand": "Dell",
  "model": "XPS 13",
  "year": 2023
}

for x in laptop_info:
  print(x)
```

brand
model
year

In [ ]:
```
# Looping Through Values
laptop_info = {
  "brand": "Dell",
  "model": "XPS 13",
  "year": 2023
}

for x in laptop_info.values():
  print(x)
```

Dell
XPS 13
2023

In [ ]:
```
# Looping Through Keys and Values
laptop_info = {
  "brand": "Dell",
  "model": "XPS 13",
  "year": 2023
}

for x, y in laptop_info.items():
  print(x, y)
```

brand Dell
model XPS 13
year 2023

In [ ]:
```
# Accessing Items in a Nested Dictionary
family = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
print(family["child1"]["year"])
print(family["child2"]["name"])
print(family["child3"]["name"])
```

2004
Tobias
Linus

Dictionary Methods

1. clear()
2. copy()
3. fromkeys()
4. get()
5. items()
6. keys()
7. values()
8. pop()
9. popitem()
10. setdefault()
11. update()

In [ ]:
```
# 1. clear(): Removes all items from the dictionary.
my_dict = {'a': 1, 'b': 2}
my_dict.clear()
print(my_dict)  # Output: {}
```

{}

In [ ]:
```
# 2. copy(): Returns a shallow copy of the dictionary.
original = {'a': 10, 'b': 20}
copied = original.copy()
print(copied)  # Output: {'a': 10, 'b': 20}
```

{'a': 10, 'b': 20}

In [ ]:
```
# 3. fromkeys(): Creates a new dictionary from the given sequence of keys and a default value.
keys = ("name", "age")
new_dict = dict.fromkeys(keys, "Unknown")
print(new_dict)  # Output: {'name': 'Unknown', 'age': 'Unknown'}
```

{'name': 'Unknown', 'age': 'Unknown'}

In [ ]:
```
# 4. get(): Returns the value for the specified key. If the key is not found, returns None or a default value.
person = {"name": "Alice", "age": 25}
print(person.get("name"))  # Output: Alice
print(person.get("city", "N/A"))  # Output: N/A
```

Alice
N/A

In [ ]:
```
# 5. items(): Returns a view object of key-value pairs.
data = {"a": 1, "b": 2}
print(data.items())  # Output: dict_items([('a', 1), ('b', 2)])

dict_items([('a', 1), ('b', 2)])
```

In [ ]:
```
# 6. keys(): Returns a view object of all keys.
info = {"id": 101, "status": "active"}
print(info.keys())  # Output: dict_keys(['id', 'status'])

dict_keys(['id', 'status'])
```

In [ ]:
```
# 7. pop(): Removes the item with the specified key and returns its value.
student = {"name": "John", "grade": "A"}
grade = student.pop("grade")
print(student)  # Output: {'name': 'John'}
print(grade)  # Output: A

{'name': 'John'}
A
{'name': 'John'}
```

In [ ]:
```
# 8. popitem(): Removes and returns the last inserted key-value pair.
record = {"id": 1, "name": "Sara"}
item = record.popitem()
print(item)  # Output: ('name', 'Sara')
print(record)  # Output: {'id': 1}

('name', 'Sara')
{'id': 1}
```

In [ ]:
```
# 9. setdefault(): Returns the value of a key. If the key does not exist, it inserts the key with the specified default
# value.
profile = {"name": "Mike"}
city = profile.setdefault("city", "New York")
print(profile)  # Output: {'name': 'Mike', 'city': 'New York'}

{'name': 'Mike', 'city': 'New York'}
```

In [ ]:
```
# 10. update(): Updates the dictionary with key-value pairs from another dictionary or iterable.
info = {"name": "Tom", "age": 31}
info.update({"age": 32, "city": "London"})
print(info)  # Output: {'name': 'Tom', 'age': 32, 'city': 'London'}

{'name': 'Tom', 'age': 31, 'city': 'London'}
```

In [ ]:
```
# 11. values(): Returns a view object of all the values.
car = {"brand": "Ford", "model": "Fiesta", "year": 2019}
print(car.values())  # Output: dict_values(['Ford', 'Fiesta', 2019])

dict_values(['Ford', 'Fiesta', 2019])
```

In [ ]: