Sets - sets is an unordered collection with no duplicates elements

In [2]:
```
Names = {"Sree","Padma","John","Kavin","Padma","Padma","Kavin"}
print(Names)  ## duplicates have been removed
```

```
{'Padma', 'Kavin', 'Sree', 'John'}
```

In [5]:
```
##  membership testing
print("Padma" in Names)
"David" in Names
```

```
True
```

Out[5]:
```
False
```

Demonstrate set operations on unique letters from two words

In [22]:
```
a = set("aabbdfgrtterhgjyd")
b = set("hhlosetuuyhfsdfg")

print(a)
print(b)

print(a - b)        ## letters in a but not in b
print(a | b)        ## letters in a or b or both
print(a & b)        ## letters in both a and b
print(a ^ b)        ## letters in a or b but not both
```

```
{'e', 'g', 'j', 't', 'a', 'h', 'y', 'b', 'f', 'r', 'd'}
{'e', 'g', 'd', 'o', 't', 'h', 'y', 's', 'f', 'l', 'u'}
{'r', 'a', 'j', 'b'}
{'o', 'h', 'f', 'l', 'd', 'u', 'e', 'g', 'j', 't', 'y', 'b', 's', 'r', 'a'}
{'e', 'g', 't', 'h', 'y', 'f', 'd'}
{'j', 'o', 'a', 'b', 's', 'l', 'r', 'u'}
```

set comprehensions

In [23]:
```
a = {x for x in "aabbdfgrtterhgjyd" if x not in "bbdfgrtt"}
print(a)

names = {x for x in "David Mathew" if x not in "Mathew"}
print(names)
```

```
{'e', 'j', 'y', 'h', 'a'}
{'D', 'i', ' ', 'v', 'd'}
```

Modify set

In [38]:
```
Names = {"Sree","Padma","John","Kavin","Padma","Padma","Kavin"}
Names.add("Sree Priya")           ## add elements
print(Names)

Names.discard("John")   ## removes matching object from an existing set
print(Names)
```

```
Names.clear()    ## removes all the elements
Names
```

```
{'Kavin', 'Sree', 'Padma', 'Sree Priya', 'John'}
{'Kavin', 'Sree', 'Padma', 'Sree Priya'}
```

Out[38]: set()

In [40]:
```
Names = {"Sree","Padma","John","Kavin","Padma","Padma","Kavin"}
print(Names.pop())  ## will remove a random item
print(Names.pop())
print(Names.pop())
```

```
Padma
Kavin
Sree
```

Set operations

In [32]:
```
junior = {"R" , "Python" ,"Excel"}
senior = {"SQL","Java","Python","Power BI"}

union = junior.union(senior)
print(union)

union2 = senior.union(junior)
print(union2)

intersection = junior.intersection(senior)
print(intersection)

set_difference = junior.difference(senior)
print(set_difference)                    ##returns elements belonging to junior b

Symmetric_difference = junior.symmetric_difference(senior)
print(Symmetric_difference)              ## returns elements not common to both s
```

```
{'SQL', 'Power BI', 'R', 'Excel', 'Python', 'Java'}
{'SQL', 'R', 'Power BI', 'Python', 'Java', 'Excel'}
{'Python'}
{'Excel', 'R'}
{'Power BI', 'Java', 'SQL', 'R', 'Excel'}
```

In [36]:
```
#### Add Any Iterable
student = {"Kavin" , "David" , "Padma" }
students = ["John","Berlin"]
student.update(students)
print(student)

students.update(student)
print(students)
```

```
{'Kavin', 'Berlin', 'Padma', 'John', 'David'}
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[36], line 7
      4 student.update(students)
      5 print(student)
----> 7 students.update(student)
      8 print(students)

AttributeError: 'list' object has no attribute 'update'
```

Dictionaries

In [41]:
```
## creating dictionaries
dict = {"David":344, "Joey":455, "Chandler":766}
dict
```

Out[41]:
```
{'David': 344, 'Joey': 455, 'Chandler': 766}
```

In [43]:
```
## accesing
print(dict["Joey"])
print(dict["David"])
```

```
455
344
```

In [44]:
```
list(dict)
```

Out[44]:
```
['David', 'Joey', 'Chandler']
```

In [46]:
```
sorted(dict)
```

Out[46]:
```
['Chandler', 'David', 'Joey']
```

In [47]:
```
"Padma" in dict
```

Out[47]:
```
False
```

dict comprehensions

In [50]:
```
a = {x:x**2 for x in (3,7,8)}
print(a)

a = {x:x**3 for x in (3,7,8)}
print(a)
```

```
{3: 9, 7: 49, 8: 64}
{3: 27, 7: 343, 8: 512}
```

Accessing components of dictionary

In [2]:
```
my_dict = {"brand": "Ford", "model": "Mustang", "year": 1964}

print(my_dict["brand"])
print(my_dict.keys())
```

```
print(my_dict.values())
print(my_dict.items())
```

```
Ford
dict_keys(['brand', 'model', 'year'])
dict_values(['Ford', 'Mustang', 1964])
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

Modifying a dictionary

In [3]:
```
my_dict["colour"]="red"
print(my_dict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'colour': 'red'}
```

In [12]:
```
my_dict = {"brand": "Ford", "model": "Mustang", "year": 1964}
my_dict.update({"colour":"red"})
my_dict
```

Out[12]:
```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'colour': 'red'}
```

In [13]:
```
my_dict["colour"] = "blue"
my_dict
```

Out[13]:
```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'colour': 'blue'}
```

In [14]:
```
del my_dict["colour"]
print(my_dict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

In [16]:
```
my_dict.clear()
my_dict
```

Out[16]:
```
{}
```

Nested Dictionaries

In [23]:
```
students  = {"class1" : {"name":"Padma" , "subjects":"mathematics" , "marks":87},
            "class2" : {"name":"Sree" , "subjects":"Physiscs" , "marks":76},
            "class3" : {"name":"Kavin" , "subjects":"chemistry" , "marks":67}}
students
```

Out[23]:
```
{'class1': {'name': 'Padma', 'subjects': 'mathematics', 'marks': 87},
 'class2': {'name': 'Sree', 'subjects': 'Physiscs', 'marks': 76},
 'class3': {'name': 'Kavin', 'subjects': 'chemistry', 'marks': 67}}
```

In [28]:
```
class1 = {
  "name" : "Mathew",
  "year" : 2004
}
class2 = {
  "name" : "David",
  "year" : 2007
}
```

```
class3 = {
  "name" : "Joey",
  "year" : 2011
}

mystudents = {
  "class1" : class1,
  "class2" : class2,
  "class3" : class3
}
mystudents
```

Out[28]:
```
{'class1': {'name': 'Mathew', 'year': 2004},
 'class2': {'name': 'David', 'year': 2007},
 'class3': {'name': 'Joey', 'year': 2011}}
```

Access Items in Nested Dictionaries

In [29]:
```
mystudents["class3"]["year"]
```

Out[29]:
```
2011
```

Coding Exercises

Exploring Python Dictionaries

(a) Create and display a dictionary with key-value pairs.

In [72]:
```
dict = [{"name":"Mathew" , "class":10 , "syllabus":"state" , "percentage":88},
        {"name":"David" , "class":9 , "syllabus":"ICSE" , "percentage":68},
        {"name":"Kavin" , "class":10 , "syllabus":"state" , "percentage":78}]
dict
```

Out[72]:
```
[{'name': 'Mathew', 'class': 10, 'syllabus': 'state', 'percentage': 88},
 {'name': 'David', 'class': 9, 'syllabus': 'ICSE', 'percentage': 68},
 {'name': 'Kavin', 'class': 10, 'syllabus': 'state', 'percentage': 78}]
```

(b) Access and display dictionary keys, values, and key-value pairs using the `keys()`, `values()`, and `items()` methods.

In [73]:
```
for student in dict:
    print(student.keys())
    print(student.values())
```

```
dict_keys(['name', 'class', 'syllabus', 'percentage'])
dict_values(['Mathew', 10, 'state', 88])
dict_keys(['name', 'class', 'syllabus', 'percentage'])
dict_values(['David', 9, 'ICSE', 68])
dict_keys(['name', 'class', 'syllabus', 'percentage'])
dict_values(['Kavin', 10, 'state', 78])
```

In [74]:
```
dict = {"name":"Mathew" , "class":10 , "syllabus":"state" , "percentage":88}

print(dict.keys())
```

```
print(dict.values())
print(dict.items())
```

```
dict_keys(['name', 'class', 'syllabus', 'percentage'])
dict_values(['Mathew', 10, 'state', 88])
dict_items([('name', 'Mathew'), ('class', 10), ('syllabus', 'state'), ('percentage',
88)])
```

(c) Update existing entries and add new key-value pairs to a dictionary.

In [75]:
```
dict.update({"state" :"kerala"})
print(dict)

dict["state"] = "karnataka"
dict
```

```
{'name': 'Mathew', 'class': 10, 'syllabus': 'state', 'percentage': 88, 'state': 'ker
ala'}
```

Out[75]:
```
{'name': 'Mathew',
 'class': 10,
 'syllabus': 'state',
 'percentage': 88,
 'state': 'karnataka'}
```

(d) Remove dictionary elements using `del`, `pop()`, or `popitem()` methods.

In [61]:
```
del dict["percentage"]
dict
```

Out[61]:
```
{'name': 'Mathew', 'class': 10, 'syllabus': 'state', 'state': 'karnataka'}
```

In [62]:
```
print(dict.pop("class"))
print(dict.pop("state"))
print(dict)
```

```
10
karnataka
{'name': 'Mathew', 'syllabus': 'state'}
```

In [63]:
```
print(dict.popitem())
```

```
('syllabus', 'state')
```

(e) Iterate through a dictionary to access keys, values, and items.

In [65]:
```
student = {'name': 'Mathew', 'class': 10, 'syllabus': 'state', 'percentage': 88, 's

for key,value in student.items():
    print(f" Key: {key}, Value: {value}")


dict = [{"name":"Mathew" , "class":10 , "syllabus":"state" , "percentage":88},
        {"name":"David" , "class":9 , "syllabus":"ICSE" , "percentage":68},
        {"name":"Kavin" , "class":10 , "syllabus":"state" , "percentage":78}]

for key,value in dict.items():
```

```
    print(f" Key: {key}, Value: {value}")
```

```
Key: name, Value: Mathew
Key: class, Value: 10
Key: syllabus, Value: state
Key: percentage, Value: 88
Key: state, Value: kerala
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[65], line 11
      4     print(f" Key: {key}, Value: {value}")
      7 dict = [{"name":"Mathew" , "class":10 , "syllabus":"state" , "percentage":8
8},
      8         {"name":"David" , "class":9 , "syllabus":"ICSE" , "percentage":68},
      9         {"name":"Kavin" , "class":10 , "syllabus":"state" , "percentage":7
8}]
---> 11 for key,value in dict.items():
     12     print(f" Key: {key}, Value: {value}")

AttributeError: 'list' object has no attribute 'items'
```

In [68]:
```
dict = [{"name":"Mathew" , "class":10 , "syllabus":"state" , "percentage":88},
        {"name":"David" , "class":9 , "syllabus":"ICSE" , "percentage":68},
        {"name":"Kavin" , "class":10 , "syllabus":"state" , "percentage":78}]

for student in dict:
    for key,value in student.items():
        print(f" Key: {key}, Value: {value}")
```

```
Key: name, Value: Mathew
Key: class, Value: 10
Key: syllabus, Value: state
Key: percentage, Value: 88
Key: name, Value: David
Key: class, Value: 9
Key: syllabus, Value: ICSE
Key: percentage, Value: 68
Key: name, Value: Kavin
Key: class, Value: 10
Key: syllabus, Value: state
Key: percentage, Value: 78
```

(f) Demonstrate the use of dictionary methods such as `update()`, `len()`, `sorted()`, and `clear()`.

In [4]:
```
dict = {"name":"Mathew" , "class":10 , "syllabus":"state" , "percentage":88}
dict.update({"state" :"kerala"})
print(dict)

print(len(dict))
print(sorted(dict))
```

```
dict.clear()
dict
```

```
{'name': 'Mathew', 'class': 10, 'syllabus': 'state', 'percentage': 88, 'state': 'ker
ala'}
5
['class', 'name', 'percentage', 'state', 'syllabus']
```

Out[4]: {}