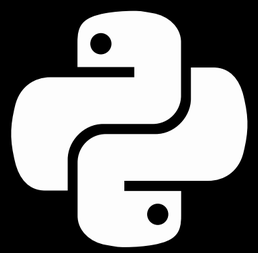

# Python
# Data structures

# Contents

- List
- Tuple
- Set
- Dictionary

# Lists

# Lists

```
x = ["apple", "orange", "banana"]
```

- A list is a collection of items enclosed in square brackets []

- Lists are ordered, and can be of any type (e.g. integers, strings, etc.)

- Lists are mutable, items can be modified after they are created.

# Accessing items in list

```
x = ["apple", "orange", "banana"]


print(x[0])
```

- Access  the items present in a list with the help of index number

# Access a range of index

```
x = ["apple", "orange", "banana", "kiwi", "strawberry"]


print(x[1 : 5])
```

# Change items present in list

```
x = ["apple", "orange", "banana"]


x[2] = "pineapple"


print(x)
```

# Insert

```
x = ["apple", "orange", "banana"]


x.insert(1, "pineapple")


print(x)
```

# Append

```
x = ["apple", "orange", "banana"]


x.append("pineapple")


print(x)
```

- This will add the specified item in the end of the list

# Extend

```
x = ["apple", "orange", "banana"]

y = ["kiwi", "pineapple", "strawberry"]


x.extend(y)


print(x)
```

- This will append elements from another list

# Remove

```
x = ["apple", "orange", "banana"]


x.remove("apple")


print(x)
```

- Remove specified string from the list

# pop

```
x = ["apple", "orange", "banana"]


x.pop(2)


print(x)
```

- Remove item present in the specified index

# Clear

```
x = ["apple", "orange", "banana"]


x.clear()


print(x)
```

- Clears the entire list

# Sorting list

```
x = ["orange", "watermelon", "banana", "apple"]


x.sort()


print(x)
```

- This will sort the list in alphabetical order, numbers will be sorted in increasing order

# Sort Descending

```
x = ["orange", "watermelon", "banana", "apple"]


x.sort(reverse = True)


print(x)
```

- This will sort the list in descending order

# Copy list

```
x = ["orange", "banana", "apple"]


y = x.copy()


print(y)
```

- This will create a new list by copying

# Tuple

# Tuple

```
x = ("apple", "orange", "banana")
```

- Tuple is a collection of items enclosed in parentheses ()
- Like lists, tuples are ordered and can be of any type.
- However, tuples are immutable, which means that their items cannot be modified after they are created.

# Accessing items in tuple

```
x = ("apple", "orange", "banana")


print(x[0])
```

- Access  the items present in a tuple with the help of index number

# Access a range of index

```
x = ("apple", "orange", "banana", "kiwi", "strawberry")


print(x[1 : 4])
```

# Tuple is immutable

```
x = ("apple", "orange", "banana")


y = list(x)


y[0] = "kiwi"


x = tuple(y)


print(y)
```

- Changing elements in tuple is not possible, instead,
- Convert tuple into list and then change the item
- Then again convert the changed list back into tuple

# Remove

```
x = ("apple", "orange", "banana")


y = list(x)


y.remove("banana")


x = tuple(y)


print(y)
```

- To remove specified string from the tuple, convert tuple into a list, then remove and convert back into tuple

# Combine two tuple

```
x = ("a", "b", "c")

y = (1, 2, 3)


z = x + y

print(z)
```

- Combine two tuple with the help of "+"

# Set

# Set

```
x = {"apple", "orange", "banana"}
```

- Set is a collection of unique items enclosed in curly braces {}
- The items in a set have no defined order, and can be of any type
- sets are mutable
- Duplicate is not allowed

# Duplicates will be ignored

```
x = {"apple", "orange", "banana", "orange"}


print(x)
```

# Add

```
x = {"apple", "orange", "banana"}


x.add("kiwi")


print(x)
```

- Use add method to add new items in the set

# Update

```
x = {"apple", "orange", "banana"}

y = {"kiwi", "mango"}

x.update(y)



print(x)
```

- Use update method to combine two set

# Remove

```
x = {"apple", "orange", "banana"}


x.remove("orange")


print(x)
```

- Use remove method to remove specified item from set

# Difference

```
x = {1, 2, 3, 4, 5, 6}


y = {5, 6, 7, 8, 9, 10}


print(x.difference(y))
```

- To find the values exist in x alone

# Difference

```
x = {1, 2, 3, 4, 5, 6}


y = {5, 6, 7, 8, 9, 10}


print(x.difference(y))
```

- To find the values exist in x alone

# Intersection

```
x = {1, 2, 3, 4, 5, 6}


y = {5, 6, 7, 8, 9, 10}


print(x.intersection(y))
```

- To find common values exist in both

# Symmetric difference

```
x = {1, 2, 3, 4, 5, 6}


y = {5, 6, 7, 8, 9, 10}


print(x.symmetric_difference(y))
```

- All the items which are not common will be in the output set

# Dictionary

# Dictionary

```
dict = {"name" : "john","age" : 30, "country" : "India"}
```

- Dictionary is a collection of key-value pairs enclosed in curly braces {}

- Dictionaries are also mutable, key-value pairs can be added, removed, or modified after they are created.

- Duplicates are not allowed

# Dictionary

```
x = {
  "name" : "john",
  "age" : 30,
  "country" : "India"
}
print(x)
```

# Duplicates are overwritten

```
x = {

   "Name" : "John",

   "Age"  : 20,

   "Country" : "India",

   "Age" : 25

}
```

# All data types are valid

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}
```

# Accessing dictionary

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]
```

```python
x = thisdict.get("model")
```

# Accessing dictionary

.KEYS()

```
x = thisdict.keys()
```

.VALUES()

```
x = thisdict.values()
```

.ITEMS()

```
x = thisdict.items()
```

# Changing values

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
```

# Update method

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})
```

# Pop

# Nested Dictionary

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```