**AI Day 10 Notes**

**Syed Mansoor ul Hassan Bukhari**

**Python Dictionaries: Detailed Notes with Examples**

1. **Definition**: Dictionaries in Python are used to store data values in key:value pairs. They are collections that are ordered (as of Python 3.7), changeable, and do not allow duplicates. They are written with curly brackets, and have keys and values.

thisdict = {

"brand": "Suzuki",

"model": "Skoda",

"year": 2021

}

print(thisdict)

1. **Dictionary Items**: Dictionary items are ordered, changeable, and do not allow duplicates. They are presented in key:value pairs and can be referred to by using the key name.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print(thisdict["brand"])

1. **Order**: As of Python version 3.7, dictionaries are ordered, meaning the items have a defined order that will not change. In Python 3.6 and earlier, dictionaries are unordered, meaning the items do not have a defined order and you cannot refer to an item by using an index.
2. **Changeability**: Dictionaries are mutable, meaning that we can change, add, or remove items after the dictionary has been created.
3. **Duplication**: Dictionaries cannot have two items with the same key. Duplicate values will overwrite existing values.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964,

"year": 2020

}

print(thisdict)

1. **Dictionary Length**: The len() function is used to determine how many items a dictionary has.

print(len(thisdict))

1. **Data Types**: The values in dictionary items can be of any data type.

thisdict = {

"brand": "Ford",

"electric": False,

"year": 1964,

"colors": ["red", "white", "blue"]

}

print(thisdict)

1. **The dict() Constructor**: It is also possible to use the dict() constructor to make a dictionary.

thisdict = dict(name = "John", age = 36, country = "Norway")

print(thisdict)

1. **Accessing Items**: Items of a dictionary can be accessed by referring to its key name, inside square brackets, or by using the get() method.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

x = thisdict["model"]

x = thisdict.get("model")

1. **Checking if Key Exists**: To determine if a specified key is present in a dictionary, use the in keyword.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

if "model" in thisdict:

print("Yes, 'model' is one of the keys in the thisdict dictionary")

1. **Updating Dictionary**: You can change the value of a specific item by referring to its key name.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict["year"] = 2020

1. **Adding Items**: Adding an item to the dictionary is done by using a new index key and assigning a value to it.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict["color"] = "red"

1. **Removing Items**: The pop() method removes the item with the specified key name.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.pop("model")

1. **Looping Through a Dictionary**: You can loop through a dictionary by using a for loop. The for loop returns the keys of the dictionary, but there are methods to return the values as well.

for x in thisdict:

print(x) # print keys

print(thisdict[x]) # print values

1. **Nested Dictionaries**: A dictionary can contain dictionaries, this is called nested dictionaries.

myfamily = {

"child1" : {

"name" : "Emil",

"year" : 2004

},

"child2" : {

"name" : "Tobias",

"year" : 2007

},

"child3" : {

"name" : "Linus",

"year" : 2011

}

}

1. **Clearing Dictionary**: The clear() keyword empties the dictionary.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.clear()

1. **Deleting Dictionary**: The del keyword removes the dictionary completely.

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

del thisdict