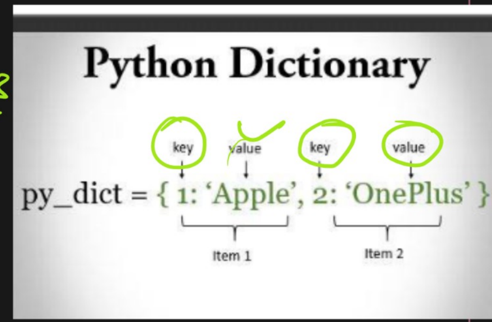
Eng

dictionary in python

Day-4 10-08-2023



1. {} ← empty
2. {pair1, pair2, pair3, ----- }
3. {key1 : value1, key2 : value2, --- } Unordered
   (key1 : value1 = pair1)
4. dict & Its Objects does not Support Indexing/Slicing.
5. dict & Its Objects are Iterable (Loop it)
6. dict Object Mutable C U R D

E-commerce

Clientside

Serverside

{"1" : "Hello"}

image, video, text, comment, ...

loaded from database

No Sql — MongodB

Key — value

Sql

Stores an Items

```
#d={}
d={"name":"Nishant","age":89,"dept":"IT",
    "name":["Suresh","Naresh","Mahesh"],98675.56:"hbzcjsad#$^%^@6825###"}
print(d)
print(id(d),type(d),len(d))
```

a=5
id(a)
a=6
id(b)

String

Number (int, float)

Key's Immutable

Update / Adde

Key — as an Array.

Key [list / tuple]

Immutable

dict

C U R D

* {} → duplicate key *
* Key should be Immutable → str, int, float, tuple

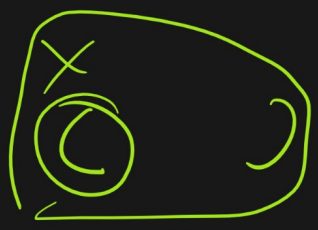
#dict() : Key should be Identifier
* Duplicate key Not allowed.

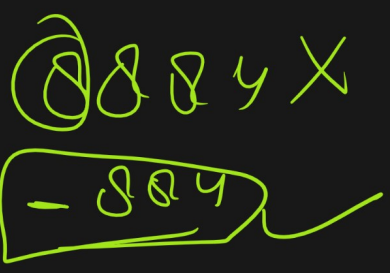
* d[Key]=value
* d[NewKey]=Newvalue
* d.update(d1)
* d.clear()
* d.copy()

* d[Key]
* d.get(key,default)
* d.values()
* d.keys()
* d.items()
* for i in d: print(i) keys

# del d[Key]
valy = d.pop(key,default) *

[A-Za-z0-9_=]





* () → Tuple
* [] → List
* {pair1, pair2} → dict
* {key1, key2, key3} → set

List

del listobject[Index]

Keyword

↓

deleted the value from list

* listobj.pop(Index)

value

method

Dict

del dictobject[key]

deleted the whole pair (key-value) from dict

dictobj.pop(key)

value

Prob-1 (I) Create A Dict. of Five (5) family Member Age & Name.

Note

Age → value

Name → key.

* Two or more Names belongs to same Age.

Print → Data Created

Flow

Enter the 1. Name = Jhon.

Enter the Jhon Age = 34

Enter the 2 Name = Greeta.

Enter the Greeta Name = 34

II ∞ while loop → Enter the Age to get Name (Press Q to exit):—

Catch → By Value → Key

1. ✱ Matched (Result Input) → One Key
   → XYZ belong to YZF age.
   → More than one Key.
   → XYZ, OZF, MZF belong to same XYZ age.
2. ✱ Not matched → Not found Match try Again!
3. ✱. If Press Q or q → exit the loop → Print — Thank you.

Python function

✱ Scope var → global scope
✱ Func var → local scope

The advantages of using functions are:

- Reducing duplication of code
- Decomposing complex problems into simpler pieces
- Improving clarity of the code
- Reuse of code

✱ Modularity Improved
✱ Memory efficient

- built-in functions → len() sum() min() max()
- user defined functions → → def demo():
  1 Tab →
  1 Tab →

positional arguments

Set

- ☆ Set can contain only unique value.
- ☆ Set can store only Immutable value.
- ☆ Set object is mutable (C U R D)
- ☆ Set object is Iterable but does not support Indexing Slicing.
- ☆ set() ← empty set
- ☆ Unordered
- ☆ {1, 1.2, 'String', (1,2,3)}

positional arguments

Follow to position to provide the Argument.

We can Bypass the order $P_1, P_2, P_3, P_4 - - -$

keyword arguments

We can Bypass the order of input.

$K_4, K_2, K_3, K_1, - - - -$

Note: Once we Provide an argument as Keyword, Rest of the Argument should be keyword.

Combo

$P_1, P_2, P_3, K_6, K_4, K_5 - - - -$

non-default argument

$ND_1, ND_2, ND_3 \ldots$

Con. Error: D_1, ND_2, D_3

default argument

$D_1, D_2, D_3 \ldots$

One start making an argument default Rest of the argus follow the Default Argum be default.

Combo

$ND_1, ND_2, D_3, D_4, D_5 \ldots$

#create a function of odd_even

```
def odd_even(n=5):
```

→ `n%2==0`

→ Even

Odd

Input → Math → fact

return out, n, fact

'Even' / 'Odd' → out

Number → n

factorial of the given Num. → fact

```
Enter a number: 8
The value 8 is Even and factorial is 40320.
```

bounce.py → time - +

↑ fact_1 ( )

↑ odd_even ( )

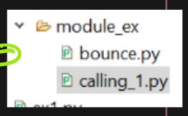
calling_1.py

Module ← Python file

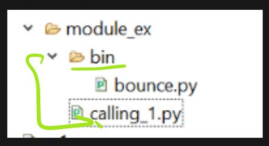
The Name of Python file Use as a Name of Module

bounce.py → bounce

(1) Both files with In same dir



(2) Module file (bounce.py) with In sub dir of calling_1.py



One Python file Executing Another Python file.

Namespace

demo.py

__name__

executed by Its main file

'__main__'

eg.py

import demo

executed by Any other Python file

Its file Name

"demo"

```
print("This is file ex1 and namespace returns ",__name__)
```

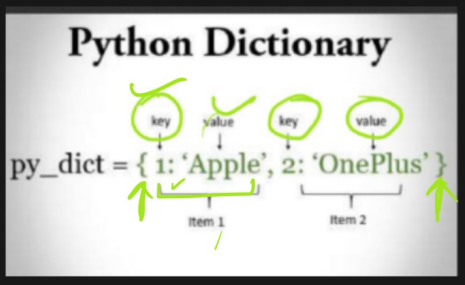
```
import ex1
print("This is file file2 and namespace returns ",__name__)
```

```
import file2
print("This is file file3 and namespace returns ",__name__)
```

If we executed file3 → output

ex1 = ex1
file2 = file2
file3 → __main__

Python Dictionaries



{ } ⇒ empty

{ pair1, pair2, pair3 }

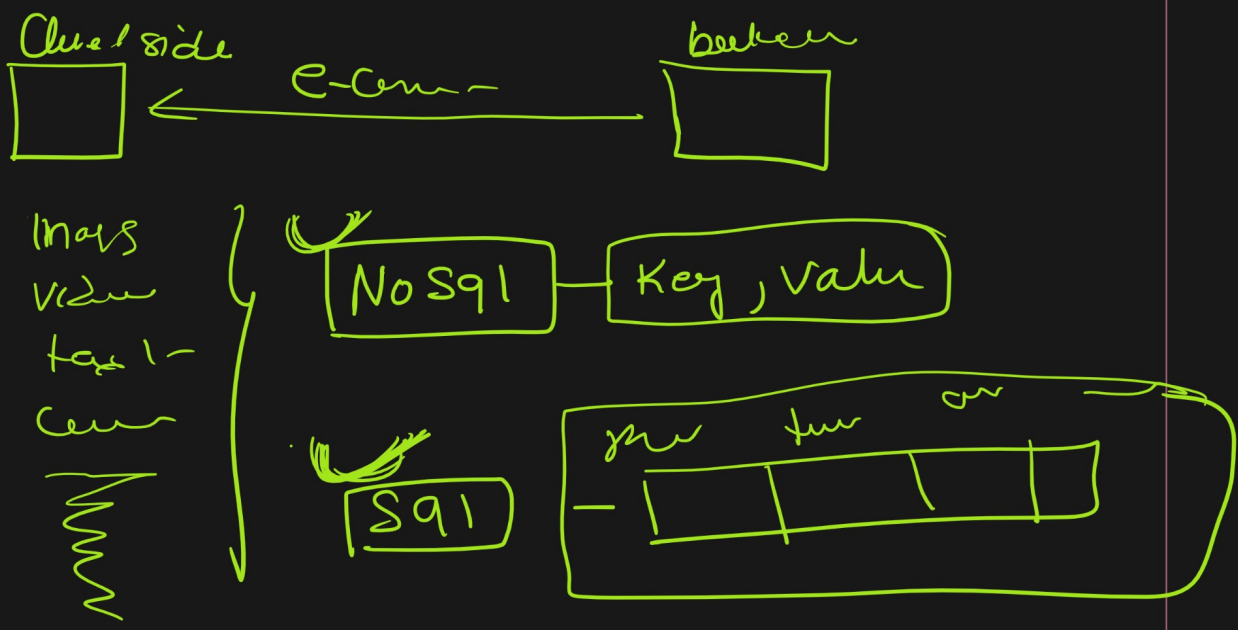
{ key1:value1, key2:value2, --- }

* dict Its objects does not support Indexing Slicing.
* dict Its objects are Iterable.
* dict Object is mutable → C U R D
* dict object is Unordered.

dict = 'name'

"name"

Json



Dict

C U R D

- #{} → Duplicate Key*
  - * Key should be Immutable → str, int, float, tuple
- #dict() → Key should be an Identifier - Duplicate Key Not allowed.

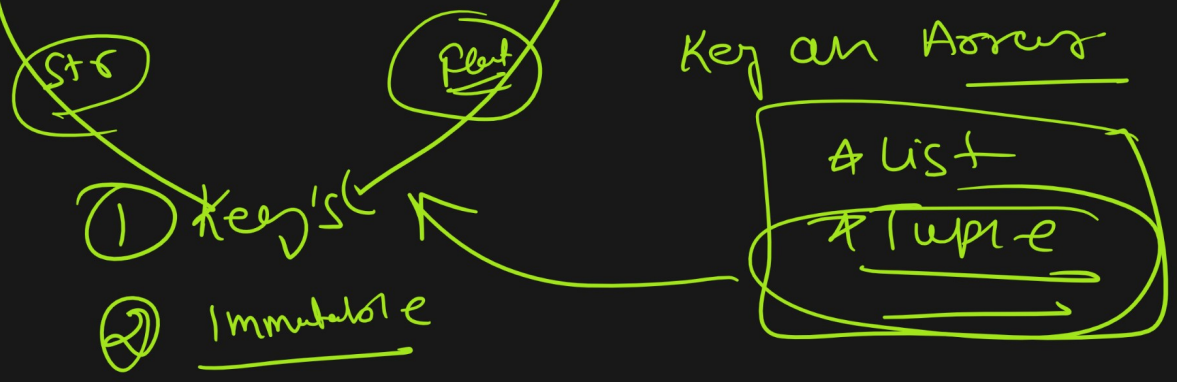
- d[Key]=value
- d[NewKey]=NewValue
- d.update(d1)
- * d.clear()
- * d.copy()

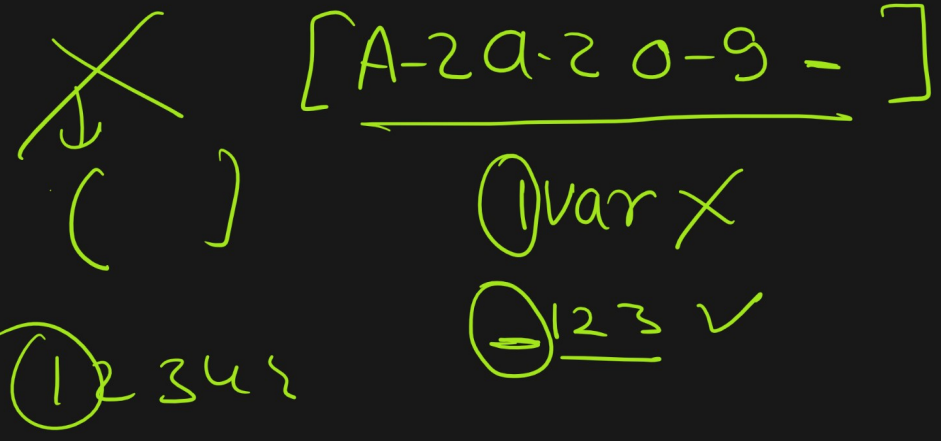
- * value=d[Key]
- * d.get(key, default)
- * d.values()
- * d.keys()
- * d.items()
- * for i in d: Pri(i) → Key

- del d[Key]
- deleted value = d.pop(key, default)

```python
d={}
d={"name":"Nishant","Age":67,"Dept":"L&D",98575.67:"hdbv#55625632bhvdsgh'
   "name":["Naresh","Suresh"]}

print(d)
print(id(d),type(d),len(d))
```





* Prob-1

```python
#################################################################


d=dict(name="Nishant",age=89,dept="L&D",_888878="Test")
d1={"one":1,"two":{"username":"Test","password":"2133"}}
#create dict d3 with pairs of d1 and d ,Note without any change in d and d1 dict.
#=======3 line============
#d3=
#=======2 lines============
#d3=
#=======1 line============
#d3=
print(d3)
print(id(d3),type(d3),len(d3))

#################################################################
```

Prob-1 (I) Create A Dict. of Five (5) family Member Age

Note
- Name.
- Age → Value
- Name → Key.

* Two of more Names belongs to same Age.

Loop (for)

Print → Data Created

Flow

Enter the 1. Name = Jhon.
Enter the Jhon Age = 34
Enter the 2 Name = Geeta.
Enter the Geeta Name = 14

→ {'Raj': 34, 'Susi': 34, 'geeta': 58, 'Jeme': 60, 'mansi': 32}

Name → Key
Age → value

II (∞) While Loop — Enter the Age to get Name (Press Q to exit):—

Catch → By Value → Key

(1) * Matched (Result Input) → One Key → XYZ belong to YZF age.

→ more than one Key → XYZ, OZF, MZF belong to same XYZ age.

(2) * Not matched → Not found Match try Again!

(3) * If Press Q or q → exit the loop print — Thank you.

Python function

var = 10 ①

Script

Script var → global scope

function var → local scope

The advantages of using functions are:

- Reducing duplication of code
- Decomposing complex problems into simpler pieces
- Improving clarity of the code
- Reuse of code

↑ Improve te Modularity
↑ Memory efficient

- built-in functions

len(), min(), max() etc.

- user defined functions

```
def demo( ):
    ...
```

Set

① Set contains only unique values.

② Set contains only Immutable values.

③ Set are unordered. or does not support Indexing Slicing.

④ Set object are Mutable CURD

⑤ set() ← empty set

⑥ {1, 1.2, 'Nisit', (1.2,1)}

positional arguments

We can not Bypass the order.

P_1, P_2, P_3, P_4 - - --

keyword arguments

We can Bypass the order by Using keyword argu

K_5, K_3, K_2, K_4, K_1 - —

Note : Once we make an argument keyword, Rest of the Argument should be keyword.

Combo

$P_1, P_2, P_3, K_5, K_4, K_6, K_7$ - - .

non-default argument

ND_1, ND_2, ND_3 - - -

Error

D_1, ND_2, D_3 --

default argument

D_1, D_2, D_3 - - -

Note Once we make an Argument default Rest of the Argum should be same

Combo

$ND_1, ND_2, D_3, D_4, D_5$ - —

bounce.py → def odd-even(number=5):

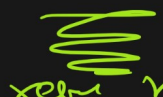
① 

return out, number, fact

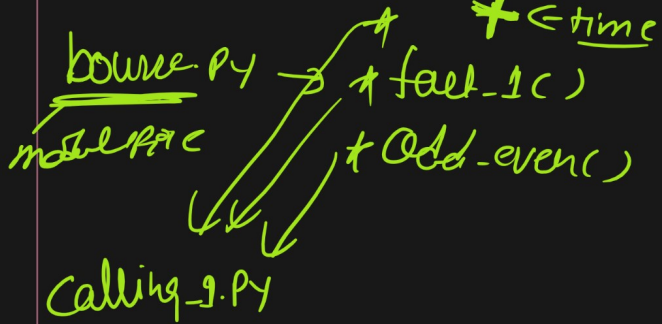
Even → ↓ odd

fact ↓ factorial

② def fact_1(n):



return myfact.

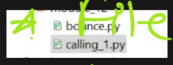


Python file treated as A module.

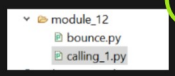
from or import

file name used as a module

bounce.py → bounce



① When Both the file In same dir



② When module In sub dir of calling file