9.DataTypes

April 29, 2020

Dictionary

- unordered
- map type object (value –> key)
- key-value collection
- keys unique and immutable
- value can be any valid python object
- iterable

[1]:
```
info = {
    'name': 'python',
    'versions': [ '1.X', '2.X', '3.X'], #  ('1.X', '2.X', '3.X')
    'father': 'Guido Van Rossum',
    'packages': {
        'ml': [ 'sci-kit learn', 'scipy', 'tensorflow', 'keras', 'opencv'],
        'ds': [ 'numpy', 'pandas', 'matplotlib', 'seaborn', 'sympy'],
        'web': [ 'flask', 'django', 'requests'],
        'gui': [ 'tkinter', 'wxpython', 'kivi'],
        'automation': [ 'ansible', 'salt']
        }

    }
```

[6]:
```
# how will i print django ?
info['packages']['web'][1]
```

[6]:
```
'django'
```

[2]:
```
from time import sleep
from tqdm import tqdm
for _ in tqdm(range(120)):
    sleep(1)
```

```
100%|
    | 120/120 [02:01<00:00,  1.01s/it]
```

```
hello --> variable / identifier

'hello' --> value
```

[10]:
```
packages = 'name'
info['packages']
```

[10]:
```
{'ml': ['sci-kit learn', 'scipy', 'tensorflow', 'keras', 'opencv'],
 'ds': ['numpy', 'pandas', 'matplotlib', 'seaborn', 'sympy'],
 'web': ['flask', 'django', 'requests'],
 'gui': ['tkinter', 'wxpython', 'kivi'],
 'automation': ['ansible', 'salt']}
```

[7]:
```
print(info)
```

```
{'name': 'python', 'versions': ['1.X', '2.X', '3.X'], 'father': 'Guido Van
Rossum', 'packages': {'ml': ['sci-kit learn', 'scipy', 'tensorflow', 'keras',
'opencv'], 'ds': ['numpy', 'pandas', 'matplotlib', 'seaborn', 'sympy'], 'web':
['flask', 'django', 'requests'], 'gui': ['tkinter', 'wxpython', 'kivi'],
'automation': ['ansible', 'salt']}}
```

[11]:
```
from pprint import pprint

# pretty print and we use to print complex data structures to show them in␣
 ↪simple manner
```

[14]:
```
pprint(info, indent=5)
```

```
{    'father': 'Guido Van Rossum',
     'name': 'python',
     'packages': {    'automation': ['ansible', 'salt'],
                      'ds': [    'numpy',
                                 'pandas',
                                 'matplotlib',
                                 'seaborn',
                                 'sympy'],
                      'gui': ['tkinter', 'wxpython', 'kivi'],
                      'ml': [    'sci-kit learn',
                                 'scipy',
                                 'tensorflow',
                                 'keras',
                                 'opencv'],
                      'web': ['flask', 'django', 'requests']},
     'versions': ['1.X', '2.X', '3.X']}
```

[17]:
```
info['packages']['ds'][-1]
```

[17]:
```
'sympy'
```

dictionaries are un-ordered

[18]:
```
info['packages']['ml']
```

[18]:
```
['sci-kit learn', 'scipy', 'tensorflow', 'keras', 'opencv']
```

[19]:
```
info['packages']['ml'].append('nltk')
```

[20]:
```
pprint(info)
```

```
{'father': 'Guido Van Rossum',
 'name': 'python',
 'packages': {'automation': ['ansible', 'salt'],
              'ds': ['numpy', 'pandas', 'matplotlib', 'seaborn', 'sympy'],
              'gui': ['tkinter', 'wxpython', 'kivi'],
              'ml': ['sci-kit learn',
                     'scipy',
                     'tensorflow',
                     'keras',
                     'opencv',
                     'nltk'],
              'web': ['flask', 'django', 'requests']},
 'versions': ['1.X', '2.X', '3.X']}
```

[21]:
```
old_list = info['packages']['ml']
```

[22]:
```
old_list
```

[22]:
```
['sci-kit learn', 'scipy', 'tensorflow', 'keras', 'opencv', 'nltk']
```

[23]:
```
info['packages']['ml'] = []
```

[24]:
```
pprint(info)
```

```
{'father': 'Guido Van Rossum',
 'name': 'python',
 'packages': {'automation': ['ansible', 'salt'],
              'ds': ['numpy', 'pandas', 'matplotlib', 'seaborn', 'sympy'],
              'gui': ['tkinter', 'wxpython', 'kivi'],
              'ml': [],
              'web': ['flask', 'django', 'requests']},
 'versions': ['1.X', '2.X', '3.X']}
```

[25]:
```
info['packages']['ml'] = old_list
```

[26]:
```
pprint(info)
```

```
{'father': 'Guido Van Rossum',
 'name': 'python',
 'packages': {'automation': ['ansible', 'salt'],
              'ds': ['numpy', 'pandas', 'matplotlib', 'seaborn', 'sympy'],
              'gui': ['tkinter', 'wxpython', 'kivi'],
```

```
                'ml': ['sci-kit learn',
                       'scipy',
                       'tensorflow',
                       'keras',
                       'opencv',
                       'nltk'],
                'web': ['flask', 'django', 'requests']},
 'versions': ['1.X', '2.X', '3.X']}
```

[28]:
```
info['packages']['ml'].insert(2, ['python', 'is', 'awesome'])
```

[29]:
```
pprint(info)
```

```
{'father': 'Guido Van Rossum',
 'name': 'python',
 'packages': {'automation': ['ansible', 'salt'],
              'ds': ['numpy', 'pandas', 'matplotlib', 'seaborn', 'sympy'],
              'gui': ['tkinter', 'wxpython', 'kivi'],
              'ml': ['sci-kit learn',
                     'scipy',
                     ['python', 'is', 'awesome'],
                     'tensorflow',
                     'keras',
                     'opencv',
                     'nltk'],
              'web': ['flask', 'django', 'requests']},
 'versions': ['1.X', '2.X', '3.X']}
```

[33]:
```
info['packages']['ml'][2][1]
```

[33]:
```
'is'
```

[34]:
```
d = {
    'name': 'Sachin',
    'age': 24,
    'country': 'India'
}
```

[35]:
```
d.keys()
```

[35]:
```
dict_keys(['name', 'age', 'country'])
```

[36]:
```
d.values()
```

[36]:
```
dict_values(['Sachin', 24, 'India'])
```

[37]:
```
d.items()
```

[37]:
```
dict_items([('name', 'Sachin'), ('age', 24), ('country', 'India')])
```

[38]:
```
x = { 'ml': [ 'java', 'c', 'c++']}
```

[40]:
```
old = x['ml']
```

[41]:
```
new =   [ 'scala', 'octave', 'python']
```

[42]:
```
print(old)
```

```
['java', 'c', 'c++']
```

[43]:
```
print(new)
```

```
['scala', 'octave', 'python']
```

[44]:
```
x['ml'] = [ old, new]
```

[45]:
```
x['ml']
```

[45]:
```
[['java', 'c', 'c++'], ['scala', 'octave', 'python']]
```

[46]:
```
x
```

[46]:
```
{'ml': [['java', 'c', 'c++'], ['scala', 'octave', 'python']]}
```

[47]:
```
new + old
```

[47]:
```
['scala', 'octave', 'python', 'java', 'c', 'c++']
```

[48]:
```
d
```

[48]:
```
{'name': 'Sachin', 'age': 24, 'country': 'India'}
```

[49]:
```
d.keys()
```

[49]:
```
dict_keys(['name', 'age', 'country'])
```

[50]:
```
d.values()
```

[50]:
```
dict_values(['Sachin', 24, 'India'])
```

[51]:
```
d.items()
```

[51]:
```
dict_items([('name', 'Sachin'), ('age', 24), ('country', 'India')])
```

[52]:
```
d.get('name')
```

[52]:
```
'Sachin'
```

[53]:
```
d.get('aldkfj')
```

[54]:
```
d['name']
```

[54]:
```
'Sachin'
```

[55]:
```
d['alsdkjf']
```

```
␣
↪---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call␣
↪last)

<ipython-input-55-ffd49c0194fd> in <module>
----> 1 d['alsdkjf']


KeyError: 'alsdkjf'
```

[56]:
```
d.get('alkdsfj', 'no information')
```

[56]:
```
'no information'
```

[57]:
```
d.get('name', 'no information')
```

[57]:
```
'Sachin'
```

[58]:
```
d
```

[58]:
```
{'name': 'Sachin', 'age': 24, 'country': 'India'}
```

[59]:
```
help(d.setdefault)
```

```
Help on built-in function setdefault:

setdefault(key, default=None, /) method of builtins.dict instance
    Insert key with a value of default if key is not in the dictionary.

    Return the value for key if key is in the dictionary, else default.
```

[60]:
```
item = d.setdefault('name', 'abracadabra')
```

[62]:
```
print(item)
```

```
Sachin
```

[63]:
```
d
```

[63]:
```
{'name': 'Sachin', 'age': 24, 'country': 'India'}
```

[64]:
```
item = d.setdefault('color', 'fair')
```

[65]:
```
print(item)
```

```
fair
```

[66]:
```
d
```

[66]:
```
{'name': 'Sachin', 'age': 24, 'country': 'India', 'color': 'fair'}
```

[67]:
```
item = d.get('abcd', 'efgh')
print(item)
print(d)
```

```
efgh
{'name': 'Sachin', 'age': 24, 'country': 'India', 'color': 'fair'}
```

[68]:
```
item = d.setdefault('abcd', 'efgh')
print(item)
print(d)
```

```
efgh
{'name': 'Sachin', 'age': 24, 'country': 'India', 'color': 'fair', 'abcd':
'efgh'}
```

[69]:
```
item = d.setdefault('name', 'bigad diya')
print(item)
print(d)
```

```
Sachin
{'name': 'Sachin', 'age': 24, 'country': 'India', 'color': 'fair', 'abcd':
'efgh'}
```

[70]:
```
d = { 'name': 'sachin',
    'age': 24,
    'language': [ 'hindi', 'english']}
```

[71]:
```
print(d)
```

```
{'name': 'sachin', 'age': 24, 'language': ['hindi', 'english']}
```

[73]:
```
pprint(d, indent=5)
```

```
{'age': 24, 'language': ['hindi', 'english'], 'name': 'sachin'}
```

[74]:
```
d['name'] = 'Sachin Yadav' # it will update name
```

[75]:
```
print(d)
```

```
{'name': 'Sachin Yadav', 'age': 24, 'language': ['hindi', 'english']}
```

[76]:
```
d['country'] = "India" # you can add new value also
```

[77]:
```
print(d)
```

```
{'name': 'Sachin Yadav', 'age': 24, 'language': ['hindi', 'english'], 'country':
'India'}
```

[78]:
```
d['old'] = d['age']
```

[80]:
```
pprint(d)
```

```
{'age': 24,
 'country': 'India',
 'language': ['hindi', 'english'],
 'name': 'Sachin Yadav',
 'old': 24}
```

[81]:
```
del d['age']
```

[82]:
```
pprint(d)
```

```
{'country': 'India',
 'language': ['hindi', 'english'],
 'name': 'Sachin Yadav',
 'old': 24}
```

[83]:
```
pprint(d, indent=5) # indent == space
```

```
{    'country': 'India',
     'language': ['hindi', 'english'],
     'name': 'Sachin Yadav',
     'old': 24}
```

[84]:
```
d
```

[84]:
```
{'name': 'Sachin Yadav',
 'language': ['hindi', 'english'],
 'country': 'India',
```

```
 'old': 24}
```

[85]:
```
d.update(name='sachin')
```

[87]:
```
pprint(d)
```

```
{'country': 'India',
 'language': ['hindi', 'english'],
 'name': 'sachin',
 'old': 24}
```

[88]:
```
d.update(country='USA', old=30, name='michle', blood_group='B+ive')
```

[89]:
```
pprint(d)
```

```
{'blood_group': 'B+ive',
 'country': 'USA',
 'language': ['hindi', 'english'],
 'name': 'michle',
 'old': 30}
```

[94]:
```
d
```

[94]:
```
{'language': ['hindi', 'english'],
 'country': 'USA',
 'old': 30,
 'blood_group': 'B+ive',
 'name': 'sachin'}
```

[95]:
```
value = d.pop('name')

print(f"name : {value} is delted from dictionary")
```

```
name : sachin is delted from dictionary
```

[96]:
```
d
```

[96]:
```
{'language': ['hindi', 'english'],
 'country': 'USA',
 'old': 30,
 'blood_group': 'B+ive'}
```

[98]:
```
help(d.pop)
```

```
Help on built-in function pop:

pop(…) method of builtins.dict instance
    D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
```

```
If key is not found, d is returned if given, otherwise KeyError is raised
```

[99]:
```
d
```

[99]:
```
{'language': ['hindi', 'english'],
 'country': 'USA',
 'old': 30,
 'blood_group': 'B+ive'}
```

[100]:
```
value = d.pop('old', 'laksdjflkjdf')
print(value)
```

```
30
```

[101]:
```
d
```

[101]:
```
{'language': ['hindi', 'english'], 'country': 'USA', 'blood_group': 'B+ive'}
```

[102]:
```
value = d.pop('abcd', 'no such key')
print(value)
```

```
no such key
```

[103]:
```
d
```

[103]:
```
{'language': ['hindi', 'english'], 'country': 'USA', 'blood_group': 'B+ive'}
```

[104]:
```
d.pop('alkdjf')
```

```
␣
↪---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call␣
↪last)

<ipython-input-104-d2da54d70b6f> in <module>
----> 1 d.pop('alkdjf')


KeyError: 'alkdjf'
```

[105]:
```
d['ldkfj']
```

```
␣
↪---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call␣
↪last)

<ipython-input-105-ac57778ad8b3> in <module>
----> 1 d['ldkfj']


KeyError: 'ldkfj'
```

[106]:
```
l = [ 1, 2, 3, 4]
```

[107]:
```
l[ 100 ]
```

```
␣
↪---------------------------------------------------------------------------

IndexError                                Traceback (most recent call␣
↪last)

<ipython-input-107-cd73b6faf110> in <module>
----> 1 l[ 100 ]


IndexError: list index out of range
```

[108]:
```
d
```

[108]:
```
{'language': ['hindi', 'english'], 'country': 'USA', 'blood_group': 'B+ive'}
```

[109]:
```
d.popitem() # it can delte any random key-value pair
```

[109]:
```
('blood_group', 'B+ive')
```

unordered

[110]:
```
enmey = {
    'thanos': 100,
    'joker': 120
}
```

[111]:
```
enmey.popitem()
```

[111]:
```
('joker', 120)
```

[112]:
```
enmey.pop('thanos')
```

[112]:
```
100
```

[113]:
```
enmey
```

[113]:
```
{}
```

[115]:
```
print(*[func for func in dir(dict) if func[0].islower() and func[0] != '_'],␣
↪sep='\n')
```

```
clear
copy
fromkeys
get
items
keys
pop
popitem
setdefault
update
values
```

[118]:
```
d = { 'name': 'sachin', 'age': 24, 'country': "India", }
```

[119]:
```
d
```

[119]:
```
{'name': 'sachin', 'age': 24, 'country': 'India'}
```

[120]:
```
new_dict = d.fromkeys(['a', 'b', 'c'])
```

[121]:
```
new_dict
```

[121]:
```
{'a': None, 'b': None, 'c': None}
```

[122]:
```
new_dict = d.fromkeys(['a', 'b', 'c'], 'ha ha ha ha')
```

[123]:
```
new_dict
```

[123]:
```
{'a': 'ha ha ha ha', 'b': 'ha ha ha ha', 'c': 'ha ha ha ha'}
```

[125]:
```
d
```

[125]:
```
{'name': 'sachin', 'age': 24, 'country': 'India'}
```

[127]:
```
java = info.fromkeys(info.keys())
```

[128]:
```
java
```

[128]:
```
{'name': None, 'versions': None, 'father': None, 'packages': None}
```

[129]:
```
info.keys()
```

[129]:
```
dict_keys(['name', 'versions', 'father', 'packages'])
```

[130]:
```
d
```

[130]:
```
{'name': 'sachin', 'age': 24, 'country': 'India'}
```

[132]:
```
n = d.fromkeys(d.keys(), 'default') # shift+ tab
```

[133]:
```
n
```

[133]:
```
{'name': 'default', 'age': 'default', 'country': 'default'}
```

[135]:
```
d = { 'name': 'sachin', 'name': 'aakash', 'name':'swagat'}
```

[136]:
```
d
```

[136]:
```
{'name': 'swagat'}
```

[ ]:
```
d
```

[137]:
```
d = dict([ ('name', 'sachin'), ('age', 10), ('country', 'india') ] )
d
```

[137]:
```
{'name': 'sachin', 'age': 10, 'country': 'india'}
```

[138]:
```
d.update([ ('name', 'Sachin Yadav'), ('language', ['hindi', 'english']),␣
↪('blood_group', 'B+ive'), ])
```

[139]:
```
d
```

[139]:
```
{'name': 'Sachin Yadav',
 'age': 10,
 'country': 'india',
 'language': ['hindi', 'english'],
 'blood_group': 'B+ive'}
```

0.1 Set

- collection of unique immutable elements

- set is mutable type
- iterable
- unordered
- follow set theory

[140]:
```
from time import sleep
from tqdm import tqdm
for _ in tqdm(range(120)):
    sleep(1)
```

```
100%|
    | 120/120 [02:00<00:00,  1.00s/it]
```

```
unviersal space

intersection

union

difference

subset

superset

disjoint set
```

[141]:
```
s = { 1, 2, 3, 4, 5, 56 }
```

[142]:
```
print(type(s))
print(id(s))
```

```
<class 'set'>
2089170899912
```

[143]:
```
print(type({}))
```

```
<class 'dict'>
```

[144]:
```
print(type(set()))
```

```
<class 'set'>
```

[146]:
```
print(*[func for func in dir(set) if func[0].islower() and func[0] != '_'],␣
↪sep='\n')
```

```
add
clear
```

```
copy
difference
difference_update
discard
intersection
intersection_update
isdisjoint
issubset
issuperset
pop
remove
symmetric_difference
symmetric_difference_update
union
update
```

[147]:
```
s1 = { 1, 2, 3, 4, 5, 1, 1, 1, 1,1 ,1 ,1, }
```

[148]:
```
print(s1)
```

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

[149]:
```
s1[2]
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-149-7087e83d753a> in <module>
----> 1 s1[2]

TypeError: 'set' object is not subscriptable
```

[150]:
```
s1 = { 1, 2, 3, 4, 5, 6, 7}
s2 = { 5, 6, 7, 8, 9, 10}
```

[151]:
```
s = s1.intersection(s2)
```

[152]:
```
print(s)
```

```
{5, 6, 7}
```

[153]:
```
s = s1.union(s2)
```

[154]:
```
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

[155]:
```
s = s1.difference(s2)
```

[156]:
```
print(s)
```

```
{1, 2, 3, 4}
```

[157]:
```
s  = s1.symmetric_difference(s2)
```

[158]:
```
s
```

[158]: {1, 2, 3, 4, 8, 9, 10}

[159]:
```
s1.isdisjoint(s2)
```

[159]: False

[160]:
```
s1.issubset(s2)
```

[160]: False

[161]:
```
s1
```

[161]: {1, 2, 3, 4, 5, 6, 7}

[162]:
```
s1.add(8)
```

[163]:
```
s1
```

[163]: {1, 2, 3, 4, 5, 6, 7, 8}

[164]:
```
s1.update([1, 2, 3,4 , 5, 6, 7, 8, 9, 10, 11, 12, 14])
```

[165]:
```
s1
```

[165]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14}

[166]:
```
s1.discard(10)
```

[167]:
```
s1
```

[167]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 14}

[168]:
```
s1.discard(1)
```

[169]:
```
s1
```

[169]:
```
{2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 14}
```

[170]:
```
s1.discard(100)
```

[171]:
```
s1
```

[171]:
```
{2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 14}
```

[172]:
```
s1.remove(4)
```

[173]:
```
s1
```

[173]:
```
{2, 3, 5, 6, 7, 8, 9, 11, 12, 14}
```

[174]:
```
s1.remove(100)
```

```
␣
↪---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call␣
↪last)

<ipython-input-174-685aafef4353> in <module>
----> 1 s1.remove(100)


KeyError: 100
```

[175]:
```
s1.pop()
```

[175]:
```
2
```

[176]:
```
s1.pop()
```

[176]:
```
3
```

[177]:
```
s1
```

[177]:
```
{5, 6, 7, 8, 9, 11, 12, 14}
```

0.1.1 Frozen Set

immutable set

[178]:
```
s = frozenset({1, 2, 3, 4,5 })
```

[179]:
```
type(s)
```

[179]:
```
frozenset
```

[180]:
```
s = { 1, 2, 3}
```

[181]:
```
s = { 'hello', 'hi', (1, 2,3 )}
```

[182]:
```
s
```

[182]:
```
{(1, 2, 3), 'hello', 'hi'}
```

[183]:
```
s  = { 1, [1, 2]}
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-183-bbe922b89940> in <module>
----> 1 s  = { 1, [1, 2]}


TypeError: unhashable type: 'list'
```

is_lower

isLower

[ ]:
```

```