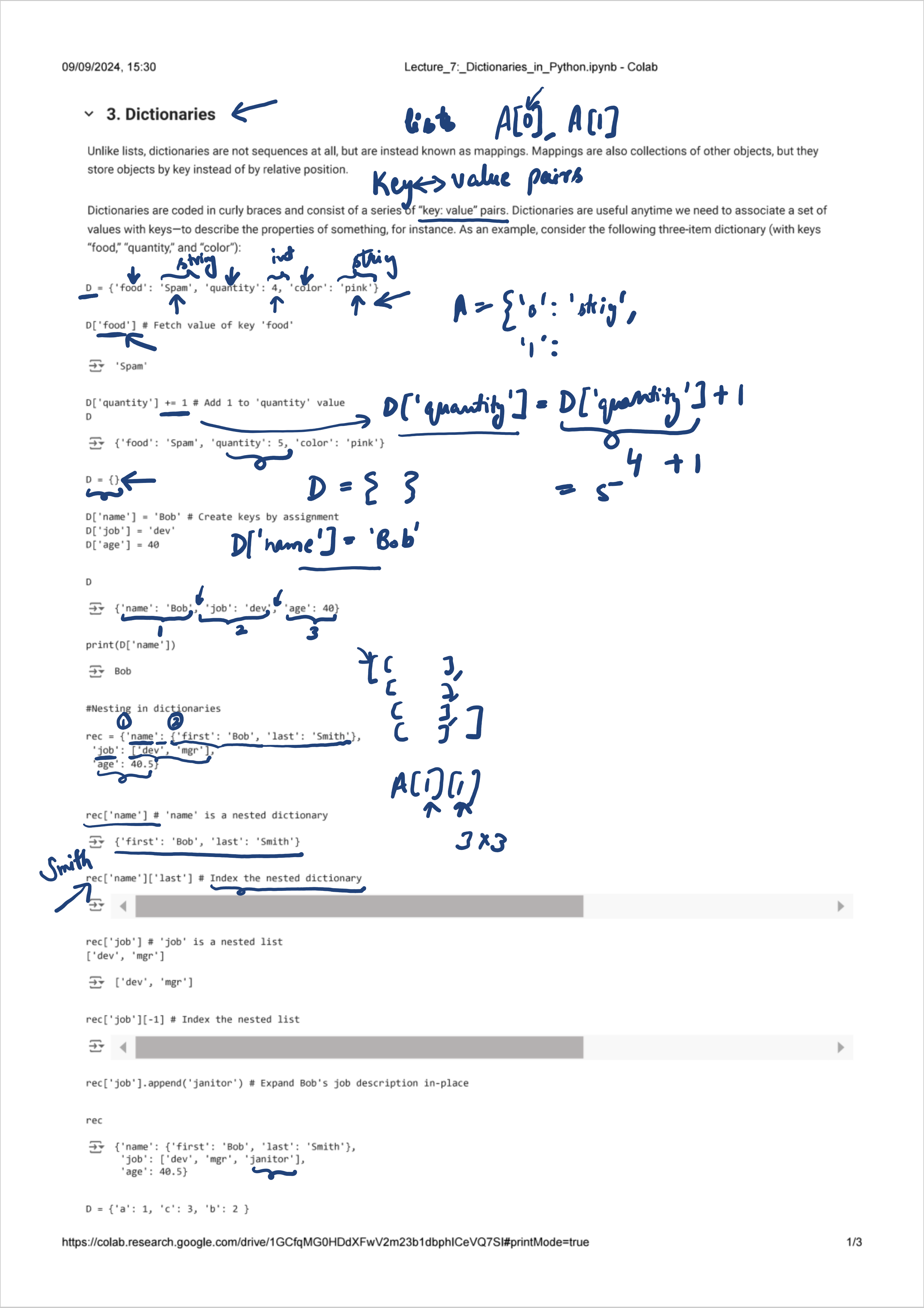
Lecture_7_Dictionaries_in_python

09 September 2024 15:32

Lecture_7_Dictionaries_in_Python.ipynb



3. Dictionaries

list A[0] A[1]

Unlike lists, dictionaries are not sequences at all, but are instead known as mappings. Mappings are also collections of other objects, but they store objects by key instead of by relative position.

Key<->value pairs

Dictionaries are coded in curly braces and consist of a series of "key: value" pairs. Dictionaries are useful anytime we need to associate a set of values with keys—to describe the properties of something, for instance. As an example, consider the following three-item dictionary (with keys "food," "quantity," and "color"):

string int string

```
D = {'food': 'Spam', 'quantity': 4, 'color': 'pink'}
```

A = {'0': 'strig',
'1':

```
D['food'] # Fetch value of key 'food'
```

```
'Spam'
```

```
D['quantity'] += 1 # Add 1 to 'quantity' value
D
```

D['quantity'] = D['quantity'] + 1

0

= 4 + 1

= 5

```
{'food': 'Spam', 'quantity': 5, 'color': 'pink'}
```

```
D = {}
```

D = { }

```
D['name'] = 'Bob' # Create keys by assignment
D['job'] = 'dev'
D['age'] = 40
```

D['name'] = 'Bob'

```
D
```

```
{'name': 'Bob', 'job': 'dev', 'age': 40}
```

1 2 3

```
print(D['name'])
```

```
Bob
```

[[ ],
[ ],
[ ],
[ ]]

```
#Nesting in dictionaries
```

1 2

```
rec = {'name': {'first': 'Bob', 'last': 'Smith'},
 'job': ['dev', 'mgr'],
 'age': 40.5}
```

A[1][1]

3x3

```
rec['name'] # 'name' is a nested dictionary
```

```
{'first': 'Bob', 'last': 'Smith'}
```

Smith

```
rec['name']['last'] # Index the nested dictionary
```

```
rec['job'] # 'job' is a nested list
['dev', 'mgr']
```

```
['dev', 'mgr']
```

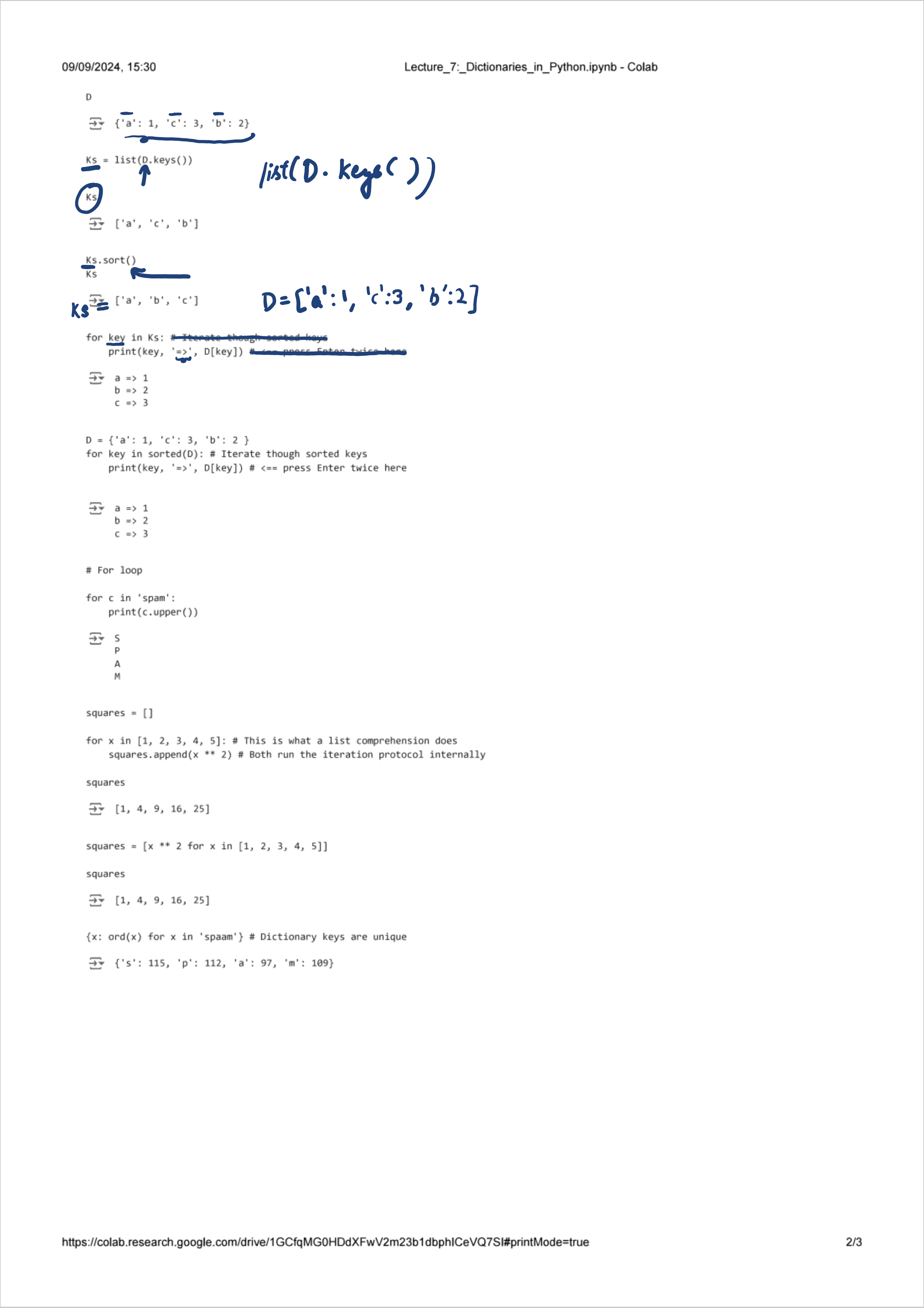
```
rec['job'][-1] # Index the nested list
```

```
rec['job'].append('janitor') # Expand Bob's job description in-place
```

```
rec
```

```
{'name': {'first': 'Bob', 'last': 'Smith'},
 'job': ['dev', 'mgr', 'janitor'],
 'age': 40.5}
```

```
D = {'a': 1, 'c': 3, 'b': 2 }
```



```
D
```

```
{'a': 1, 'c': 3, 'b': 2}
```

```
Ks = list(D.keys())
```

list(D. keys( ))

```
Ks
```

```
['a', 'c', 'b']
```

```
Ks.sort()
Ks
```

Ks =

```
['a', 'b', 'c']
```

D=['a':1, 'c':3, 'b':2]

```
for key in Ks: ~~# Iterate though sorted keys~~
    print(key, '=>', D[key]) ~~# <== press Enter twice here~~
```

```
a => 1
b => 2
c => 3
```

```
D = {'a': 1, 'c': 3, 'b': 2 }
for key in sorted(D): # Iterate though sorted keys
    print(key, '=>', D[key]) # <== press Enter twice here
```

```
a => 1
b => 2
c => 3
```

```
# For loop
```

```
for c in 'spam':
    print(c.upper())
```

```
S
P
A
M
```

```
squares = []
```

```
for x in [1, 2, 3, 4, 5]: # This is what a list comprehension does
    squares.append(x ** 2) # Both run the iteration protocol internally
```

```
squares
```

```
[1, 4, 9, 16, 25]
```

```
squares = [x ** 2 for x in [1, 2, 3, 4, 5]]
```

```
squares
```

```
[1, 4, 9, 16, 25]
```

```
{x: ord(x) for x in 'spaam'} # Dictionary keys are unique
```

```
{'s': 115, 'p': 112, 'a': 97, 'm': 109}
```