Dictionaries, nested structures

CS195 - Lecture 9
Instructor: Dr. V

Lecture 9

- dictionaries
- copy() vs assignment
- nested structures
  - shared references
  - shallow vs deep copy
- inline for loops
  - inline for loop with if/else
  - nested inline for loop

python dictionaries

- dictionaries are
  - iterable (by key)
  - mutable
  - **unordered**
  - each key in a dict is **unique** and **immutable**
  - each item in a dict is a key-value pair

- in a dictionary you look up items by key, not index

```
d = {"Temi":20, "Bobby":22}
print( d["Bobby"] )  # outputs 22
```

python dictionaries

```
1  age = {'Temi':20, 'Bobby':22}
2  age['Rhonda'] = 22  # add one item to dict
3  age.update( Sam=17, Joe=27} ) # add multiple items to dict
4  age.update( {'B.B.':17, 'Jay Z':52} ) # add multiple items to dict
5
6  # update items in dictionary
7  age['Joe'] = 28
8  age['Temi'] += 1
9
10 #remove item from dictionary
11 del age['Bobby']
12
13 #what do you think this prints?
14 print( age['Temi'] )
15 print( len(age) )
```

python dictionaries

```
1  d = {'x':1,'y':2,'z':3}
2
3  print('x' in d) # check if key is in dict
4
5  print( d['x'] )
6  print( d['xx'] ) # KeyError
7  print( d.get('xx') ) # if key 'xx' isn't found, return None
8  print( d.get('xx','hi') ) # if key 'xx' isn't found, return 'hi'
9
10 d['yy']=22
11 d.setdefault('yy',0) # if key 'yy' isn't found, add 'yy':0
12
13 d.clear()
14
15
```

iterating through python dictionaries

```
1  d = {'x':1,'y':2,'z':3}
2
3  for k in d: #iterate through dictionary keys
4      print(f'{k}:{d[k]}')
5
6  for k,v in d.items(): #iterate through dictionary items
7      print(f'{k}:{v}')
8
9  for v in d.values(): #iterate through dictionary values
10     print(v)
11
12 while d:
13     print( d.popitem() )
14
15
```

creating a dictionary

```
1  # create blank dict
2  d = {}
3  d = dict()
4  # create dict with items
5  d = {'name':'Dionysia', 'age':28, 'location':'Athens'}
6  d = dict(name='Dionysia', age=28, location='Athens')
7  # create dict from another dict (or dict-like object)
8  d = dict({'name':'Dionysia','age':28,'location':'Athens'})
9  # create dictionary from sequence of key-value pairs
10 d = dict([('name','Dionysia'),('age',28),('location','Athens')])
11 # create dictionary with just keys (all values are None)
12 d = dict.fromkeys( ['x','y','z'] )
13 # create dictionary with just keys, set all values to 10
14 d = dict.fromkeys( ['x','y','z'], 10 )
15
```

copying vs assignment

- when you assign a variable to another **mutable** variable, changing anything in one of the variables also changes it in the other
  - `d1 = {"Temi":20, "Bobby":22}`
  - `d2 = d1`
  - `d2["Temi"] = 21`
  - `print(d1) # outputs {"Temi":21, "Bobby":22}`
- if you do not want this behavior, you have to copy, rather than assign

copying mutable structures

```
1  l1 = ['x','y','z']
2  s1 = {'x','y','z'}
3  d1 = {'x':1,'y':2,'z':3}
4
5  # copying using copy method
6  l2 = l1.copy()
7  s2 = s1.copy()
8  d2 = d1.copy()
9
10 # copying using list(), set(), dict()
11 l3 = list(l1)
12 s3 = list(s1)
13 d3 = dict(d1)
14
15
```

== vs is

```
1  l1 = [1,2,3]
2  l2 = l1         # assign l2 to the same address in memory as l1
3  l3 = list(l1)   # create new list l3 as a copy of l1
4
5  # what does this print?
6  print( l1 == l2 )
7  print( l1 == l3 )
8
9  # what does this print?
10 print( l1 is l2 )
11 print( l1 is l3 )
12
13 # == operator checks if two vars look the same
14 # is operator checks if two vars point to the same address
15
```

...but what if you have a mutable item inside...

```
1  l1 = [ 10, 'a', [1,2] ]
2  l2 = list(l1)  #create a copy of l1
3
4  l2[0] += 1
5  l2[1] += 'b'
6  l2[2] += [3,4]
7
8  #what does this print?
9  print(l1)
10 print(l2)
11
12
13
14
15
```

shallow vs deep copy

```
1  from copy import deepcopy
2
3  l1 = [ 10, 'a', [1,2] ]
4  l2 = list(l1)     #create a *shallow* copy of l1
5  l3 = deepcopy(l1) #create a *deep* copy of l1
6
7  l2[2] += [3,4]
8  l3[2] += [5,6]
9
10 #what does this print?
11 print(l1)
12 print(l2)
13 print(l3)
14
15
```

python inline for loops

```
1  l = []
2  for x in range(5):
3      l.append(x**2)
4
5  # what does this print?
6  print(l)
7
8
9  # equivalent code with an inline for-loop:
10 l = [x**2 for x in range(5)]
11 print(l)
12
13
14
15
```

python inline for loops with filter

```
1  PRIME = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
2  53, 59, 61, 67, 71, 73, 79, 83, 89, 97}
3
4  #example of filtering a list:
5  #  get a list of number to 100 that are not in PRIME
6  l = []
7  for x in range(100):
8      if x not in PRIME:
9          l.append(x)
10
11
12 # equivalent code with an inline for-loop:
13 l = [x for x in range(100) if x not in PRIME]
14
15
```

inline for-loops to create list, set, tuple, dict

```
1  PRIME = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
2  53, 59, 61, 67, 71, 73, 79, 83, 89, 97}
3
4  # inline for-loop to create a list
5  l = [x for x in range(100) if x not in PRIME]
6
7  # inline for-loop to create a set
8  s = {x for x in range(100) if x not in PRIME}
9
10 # inline for-loop to create a tuple
11 t = tuple(x for x in range(100) if x not in PRIME)
12
13 # inline for-loop to create a dict (key is x, value is x**2)
14 d = dict((x,x**2) for x in range(100) if x not in PRIME)
15
```

nested inline for loops

```
1  # create a list with every combination of 1,2,3 and a,b,c
2  l = []
3  for i in (1,2,3):
4      for j in 'abc':
5          l.append( (i,j) )
6  #what do you think this prints?
7  print(l)
8
9
10 # equivalent code using inline for-loop
11 l = [(i,j) for i in (1,2,3) for j in 'abc']
12 print(l)
13
14
15
```

nested inline for loops

```
1  l = []
2  for i in (1,2,3):
3      for j in (1,2,3):
4          if i!=j:
5              l.append( (i,j) )
6  #what do you think this prints?
7  print(l)
8
9
10 # equivalent code using inline for-loop
11 l = [(i,j) for i in (1,2,3) for j in (1,2,3) if i!=j]
12 print(l)
13
14
15
```

nested inline for loops

```
1  # you can nest more that two for loops
2
3  # create list with every combination of (1,2,3), (a,b,c), (x,y,z)
4  l = [(i,j,k) for i in (1,2,3) for j in 'abc' for k in 'xyz']
5
6  print(l)
7
8
9
10
11
12
13
14
15
```

Assignment 8

- create a jupyter notebook a8.*yourLastName*.ipynb
  - add a markdown block atop the notebook with your name, class number/section, assignment number
  - add the following python code blocks
    - create a dictionary, d1, with three str keys (you pick what keys), each with a different numeric value (you pick the values); print(d1)
    - change one of the values in d1 to be a list [1,2,3]; print(d1)
    - create a deep copy of d1, called d2; add 4 to the value in d2 that was a list (that value will now be [1,2,3,4]; print(d1); print(d2)