Day - 26

**List Comprehension**

**NATO Alphabet : project**

List & Dictionary & DataFrame Comprehensions, NATO Alphabet

**26.1 List Comprehension**

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

* unique to python
* create a new list from an existing list
* Sequences: work with any SEQUENCES such as ***list***, ***string***, ***range***, ***tuple.*** Because *SEQUENCES* has specific order.

|  |  |
| --- | --- |
| Regular way | List comprehension |
| fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  newlist = []  **for** x **in** fruits:  **if** "a" **in** x:      newlist**.append**(x)  **print**(newlist) | #*conditional List comprehension*  new\_fruits = [item **for** item **in** fruits **if** ("a" **in** item)]  **print**(a\_in\_fruits\_list) |

* The Syntax:

newlist = [expression **for** item **in** iterable **if** condition == **True**]

The return value is a new list, leaving the old list unchanged.

* ***condition***: The condition is like a filter that only accepts the items that valuate to True.
* The ***iterable*** can be any iterable object, like a ***list***, ***tuple***, ***set*** etc.

newlist = [x **for** x **in** **range**(10)]

Same example, but with a condition:

newlist = [x **for** x **in** **range**(10) **if** x **<** 5]

* ***expression***: The expression is the current item in the iteration, but it is also the outcome, which you can manipulate before it ends up like a list item in the new list:
* The ***expression*** can also contain ***conditions***, not like a filter, but as a way to manipulate the outcome:

newlist = [x **if** x **!=** "banana" **else** "orange" **for** x **in** fruits]

* Examples:

#*conditional List comprehension*

a\_in\_fruits\_list = [item **for** item **in** fruits **if** ("a" **in** item)]

newlist = [x **for** x **in** fruits **if** x **!=** "apple"]

**print**(a\_in\_fruits\_list)

**print**(newlist)

* The ***condition*** is *optional* and can be omitted:

newlist = [x **for** x **in** fruits]

More Examples

#*------------  Examples:  -------------*

#*--------------   apply to*numbers*---------------*

numbers = [1, 2, 3]

new\_numbers = [i+ 1 **for** i **in** numbers]

**print**(new\_numbers)

#*--------------   apply to*strings*---------------*

name = "Bandhu"

letters\_on\_name = [ch **for** ch **in** name]

**print**(letters\_on\_name)

#*--------------   working with*range*---------------*

dubled = [2\*i **for** i **in** **range**(1, 5)]

**print**(dubled)

#*--------------   apply to*strings*---------------*

#*only add names have four letters. Filter the names*

#*in this case, new\_item and item's name are same; because we just want a list of items that's passed the condition.*

names = ['Alex', 'Beth', 'Caroline', 'Eleanor' ]

four\_letter\_names = [item **for** item **in** names **if** (**len**(item) **==** 4)]

**print**(four\_letter\_names)

#*Above names in uppercase*

upper\_case\_names = [nm**.upper**() **for** nm **in** names]

**print**(upper\_case\_names)

* Exercise 26.1: Squared numbers:

num = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

sqrd = [n\*n **for** n **in** num]

**print**(sqrd)

#*python squrd\_num\_li\_cmphnsn.py*

* Exercise 26.2: Filter evens from above numbers:

num = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

evn = [m **for** m **in** num **if** (m%2 **==** 0)]

**print**(evn)

#*python squrd\_num\_li\_cmphnsn.py*

* Exercise 26.3: Working with two files. Find the common number from two files.

# practiced version

**with** **open**("file1.txt") **as** file1:

    file1\_data = file1**.readlines**()

**with** **open**("file2.txt") **as** file2:

        file2\_data = file2**.readlines**()

        common = [**int**(item**.strip**()) **for** item **in** file1\_data **if** (item **in** file2\_data)]

**print**(common)

**print**(common)

# instructor solution

**with** **open**("file1.txt") **as** file1:

    file1\_data = file1**.readlines**()

**with** **open**("file2.txt") **as** file2:

    file2\_data = file2**.readlines**()

common\_2 = [**int**(num) **for** num **in** file1\_data **if** (num **in** file2\_data)]

**print**("Instructor version : ",common\_2)

#*python two\_file\_common\_num.py*

* Exercise 26.4: Applying to US states game. Used:

missing\_states = [state **for** state **in** all\_states **if** (state **not** **in** guessed\_states)]

**import** turtle

**import** pandas

screen = turtle**.Screen**()

screen**.title**("U.S. States Game")

image = "blank\_states\_img.gif"

screen**.addshape**(image)

turtle**.shape**(image)

data = pandas**.read\_csv**("50\_states.csv")

all\_states = data**.**state**.to\_list**()

guessed\_states = []

**while** **len**(guessed\_states) **<** 50:

    answer\_state = screen**.textinput**(title=f"{len(guessed\_states)}/50 States Correct",

                                    prompt="What's another state's name?")**.title**()

**if** answer\_state **==** "Exit":

        missing\_states = [state **for** state **in** all\_states **if** (state **not** **in** guessed\_states)]

        new\_data = pandas**.DataFrame**(missing\_states)

        new\_data**.to\_csv**("states\_to\_learn.csv")

**break**

**if** answer\_state **in** all\_states:

        guessed\_states**.append**(answer\_state)

        t = turtle**.Turtle**()

        t**.hideturtle**()

        t**.penup**()

        state\_data = data[data**.**state **==** answer\_state]

        t**.goto**(**int**(state\_data**.**x), **int**(state\_data**.**y))

        t**.write**(answer\_state)

**26.2 Dictionary Comprehension**

* Dictionary from list:

new\_dict = {new\_key:new\_value **for** item **in** list condition == **True** }

* Dictionary from Existing Dictionary:

new\_dict = {new\_key:new\_value **for** (key,value) **in** dict.items()condition == **True** }

* In both cases notice the **{}** instead of **[]**. Because these are basically new Dictionary.

#*create a new dictionary from a list*

**import** random

names = ['Alex', 'Beth', 'Caroline', 'Dave', 'Eleanor', 'Freddie' ]

student\_random\_score = {nam : random**.randint**(60, 89) **for** nam **in**  names }

**print**(student\_random\_score)

#*create a new dictionary from a dictionary*

passed\_students = {nam : score **for** (nam, score) **in** student\_random\_score**.items**() **if** (score **>** 70)}

**print**(passed\_students)

* notice "***key-value***" pair is as ***tuple***, "***(nam, score)***" for "***name : score***"
* (nam, score) as items applied on student\_random\_score**.items**() which returns all ***key-values*** as ***tuples***.

**print**(student\_random\_score**.items**()) #*returns all key-values as tuples*

returnes

*dict\_items([('Alex', 62), ('Beth', 79), ('Caroline', 72), ('Dave', 86), ('Eleanor', 67), ('Freddie', 61)])*

* Exercise 26.5: Count letters in words of given sentence.

sentence = "What is the Airspeed Velocity of an Unladen Swallow?"

#*Don't change code above*

#*Write your code below:*

word\_lenghts = {wrd: **len**(wrd) **for** wrd **in** sentence**.split**()}

**print**(word\_lenghts)

#*python letter\_count\_of\_sentence.py*

* Exercise 26.6: Celsius to Fahrenheit. Do not use ***:*** instead of ***,*** in ***(key, value)*** tuples.

weather\_c = {

    "Monday": 12,

    "Tuesday": 14,

    "Wednesday": 15,

    "Thursday": 14,

    "Friday": 21,

    "Saturday": 22,

    "Sunday": 24,

}

#*Don't change code above*

#*Write your code  below:*

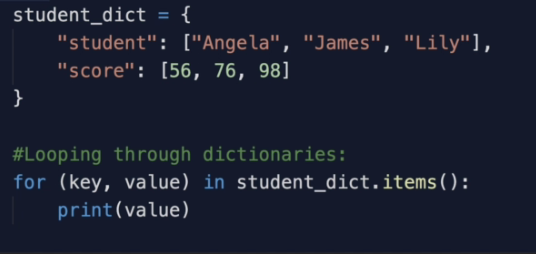
weather\_f = {day:(cel\*(9/5)) + 32 **for** (day, cel) **in** weather\_c**.items**()}

**print**(weather\_f)

#*python frnh\_cel\_dic\_cmph.py*

**26.3 iterate over a pandas data frame**

We can use loops with pandas ***dataframes*** and iterate over a pandas ***data frame***.



**for** (key, value) **in** dictionary.items()

* In similar way we can loop through Pandas DataFrame using ***iterrows()***: (we can also use exact same syntax).

**import** pandas

weather\_c = {

    "Monday": 12,

    "Tuesday": 14,

    "Wednesday": 15,

    "Thursday": 14,

    "Friday": 21,

    "Saturday": 22,

    "Sunday": 24,

}

# looping through Dictionary

**for** (day, temp) **in** weather\_c**.items**():

**print**(f"Day : {day} , Temp : {temp}\n")

# dictionary to list. "Scaler" to "vector"

weather\_data = {

    "day": [dAy **for** (dAy, teMp) **in** weather\_c**.items**()],

    "temp": [teMp **for** (dAy, teMp) **in** weather\_c**.items**()]

}

weather\_data\_frme = pandas**.DataFrame**(weather\_data)

**print**(weather\_data\_frme)

#*Gives titles of each columns*

**for** (key, value) **in** weather\_data\_frme**.items**():

**print**(key) #*Gives titles of each columns*

**print**(value) #*Gives  column values*

* ***iterrows() l***oop of Panda:

#*--------------    iterrows() --------------*

# the iterrows() helps to loop through rows of data frame

**for** (inDex, rOw) **in** weather\_data\_frme**.iterrows**():

    #*print(inDex)*

    #*print(rOw)*

    #*Accessing specific row*

**print**(rOw**.**temp) #*shows temperature*

    #*print(rOw.day) # shows dayes*

**if** rOw**.**day **==** "Friday":

**print**(rOw**.**temp)

#*python lopp\_panda\_dataframe.py*

* Exercise 26.7: NATO Alphabet Project.

Practiced version

**import** pandas

nato\_data = pandas**.read\_csv**("nato\_phonetic\_alphabet.csv")

name = **input**("Enter the name :")**.upper**()

list\_of\_letters = name**.strip**()

dictionary\_of\_name\_latters = {row**.**letter: row**.**code **for** (idx, row) **in** nato\_data**.iterrows**() **if** (row**.**letter **in** list\_of\_letters)}

list\_of\_code = [dictionary\_of\_name\_latters[ch] **for** ch **in** list\_of\_letters]

**print**(dictionary\_of\_name\_latters)

**print**(list\_of\_code)

#*python main.py*

Instructors solution

**import** pandas

nato\_data = pandas**.read\_csv**("nato\_phonetic\_alphabet.csv")

phonetic\_dic = {row**.**letter: row**.**code **for** (idx, row) **in** nato\_data**.iterrows**()}

name = **input**("Enter the name :")**.upper**()

list\_of\_code = [phonetic\_dic[ch] **for** ch **in** name]

**print**(list\_of\_code)

#*python main.py*