sandeepsuryaprasad / **python_tutorials** Private

Code Issues Pull requests Actions Projects Wiki Securi

master python_tutorials / 4_comprehensions / **_comprehensions.py** / Jump to Go to file

Sandeep Suryaprasad added exa... Latest commit 91855fd 2 days ago **History**

0 contributors

219 lines (172 sloc) 7.26 KB Raw Blame

```
import math
'''
1. List comprehensions is a way to build lists from sequences or
    any other iterable type by filtering and transforming items.

2. The general syntax for a list comprehension is as follows:
    [expression for item in iterable if condition]
'''
# List Comprehensions are used for building a new list
# Square Numbers_And_Booleans in the list. Using 'for' loop
nums = [1, 2, 3, 4, 5]

# Square Numbers in the list. Using List 4_Comprehensions
list_evens = [num ** 2 for num in nums]

# List of even numbers between range 1-50
even_numbers = [num for num in range(1, 50) if num % 2 == 0]

# Returns a list containing all vowels in the given string
names = ['laura', 'steve', 'bill', 'james', 'bob', 'greig', 'scott', 'alex',
vowel_names = [name for name in names if name[0] in "aeiou"]

# Filtering all the languages which starts with 'p'
languages = ['Python', 'Java', 'Perl', 'PHP', 'Python', 'JS', 'C++', 'JS', 'P
p_languages = [language for language in languages if language.lower().startsw
# Alternate Solution
p_languages = [language for language in languages if language.lower()[0] == '
```

```python

# Names starting with consonents
names = [name for name in names if not name[0] in "aeiou"]

# Filtering out those names which are less than 6 characters
names = ['apple', 'google', 'yahoo', 'gmail', 'flipkart', 'instagram', 'micro
short_names = [name for name in names if len(name) < 6]

# Raise to the power of list index
a = [1, 2, 3, 4, 5]
i = [value ** index for index, value in enumerate(a)]

# Build a list of tuples with string and its length pair
names = ['apple', 'google', 'yahoo', 'facebook', 'yelp', 'flipkart', 'gmail',
str_len_pair = [(name, len(name)) for name in names]

# Build a list with only even with even length string
names = ['apple', 'google', 'yahoo', 'facebook', 'yelp', 'flipkart', 'gmail',
even_string = [name for name in names if len(name) % 2 == 0]

# Generating List of PI values
pi_list = [round(math.pi, n) for n in range(1, 6)]

# List comprehension to sum the factorial of numbers from 1-5
a = [1, 2, 3, 4, 5]
s = sum([math.factorial(number) for number in a])

# Reverse the item of a list if the item is of odd length string
names = ['apple', 'google', 'yahoo', 'facebook', 'yelp', 'flipkart', 'gmail',
reverse_odd_length = [name[::-1] for name in names if len(name) % 2 != 0]

# Using "else" in Comprehension
# Reverse the item of a list if the item is of odd length string otherwise ke
names = ['apple', 'google', 'yahoo', 'facebook', 'yelp', 'flipkart', 'gmail',
reverse_odd_length = [name if len(name) % 2 == 0 else name[::-1] for name in

# Alternate solution to avoid both "if"  and "else" condition in comprehensio
# Write a seprate function and call the function repretedly.
def process_name(name):
    if len(name) % 2 == 0:
        return name
    else:
        return name[::-1]

reverse_odd_length = [process_name(name) for name in names]
```

```python

data = ['hello', 123, 1.2, 'world', True, 'python']
d = [item[::-1] if isinstance(data, str) else item for item in data]

# Reverse the string if the string is of odd length, otherwise keep it as is.
names = ['apple', 'google', 'yahoo', 'facebook', 'yelp', 'flipkart', 'gmail',
_names = [name[::-1] if len(name) % 2 == 0 else name for name in names]

# Building a list of prime numbers from 1-50.
def is_prime(number):
    for i in range(2, number):
        if number % i == 0:
            return False
    return True

prime_numbers = [ i  for i in range(1, 51) if is_prime(i)]

# Adding items of two lists
a = [1, 2, 3, 4]
b = [5, 6, 7, 8]
total = [ x + y for x, y in zip(a, b)]

# Multiple "for" loops in comprehension
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
# o/p [1, 2, 3, 4, 5, 6, 7, 8, 9]

flattened_matrix = [ ]
# without using comprehension
for row in matrix:
    for item in row:
        flattened_matrix.append(item)

# Using Comprehension
flattened_matrix = [ item  for row in matrix for item in row ]

# Concatinating numbers and letters
letters = "ABCDEFGH"
numbers = [0, 1, 2, 3, 4, 5, 6, 7]

def concat_numbers_letters(some_letter, some_number):
    return f"{some_letter}{some_number}"

result = [ concat_numbers_letters(letter, number) for letter, number in zip(l

# Dictionary Comprehension
# Building a dict of word and length pair
words = "This is a bunch of words"
d = {word: len(word) for word in words.split()}

# Flipping keys and values of the dictionary using dict comprehension
d = {'a': 1, 'b': 2, 'c': 3}

f = {value: key for key, value in d.items()}

sentence = "hello world welcome to python hello hi world welcome to python"
dict_word_count = {word: sentence.count(word) for word in sentence.split(' ')

# Counting the number of each character in a String
my_string = 'guido van rossum'
dict_char_count = {c: my_string.count(c) for c in my_string}
print(dict_char_count)

# Dictionary of character and ascii value pairs
s = 'abcABC'
dict_ascii = {
    c: ord(c)
    for c in s
}
print(dict_ascii)

# Buildings
buildings = {
                'burj khalifa':                    828,
                'Shanghai_Tower':                  632,
                'Abraj_Al_Bait_Clock Tower':       601,
                'Ping_An_Finance_Centre_Shenzhen': 599,
                'Lotte World Tower':               554.5,
                'World Trade Center':              541.3
                }

buildings_feets = {building: height * 3.28 for building, height in buildings.


# Creating Dictionary of city and population pairs using Dict Comprehension
cities = ['Tokyo',
          'Delhi',
          'Shanghai',
          'Sao Paulo',
          'Mumbai'
          ]
```

```python
population = ['38,001,000',
              '25,703,168',
              '23,740,778',
              '21,066,245',
              '21,042,538'
              ]
pairs = {city: population for city, population in zip(cities, population)}

# Dial Codes
dial_codes = [
    (86, 'China'),
    (91, 'India'),
    (1, 'United States'),
    (62, 'Indonesia'),
    (55, 'Brazil'),
    (92, 'Pakistan'),
    (880, 'Bangladesh'),
    (234, 'Nigeria'),
    (7, 'Russia'),
    (81, 'Japan')
    ]

country_codes = {code: country for code, country in dial_codes}

# Building a dictionary whose price value is more than 200
prices = {
    'ACME': 45.23,
    'AAPL': 612.78,
    'IBM': 205.55,
    'HPQ': 37.20,
    'FB': 10.75
}

p = {company: price for company, price in prices.items() if price > 200}

# Set Comprehension
# The difference between Dictionaty Comprehension and Set Comprehension is th
# have key value pair

nums = [1, 2, 3, 4, 5, 6, 1, 2, 3, 4]
s = {num ** 2 for num in nums}
print(s)

# Comprehension with 2 for loops!
n = [(x, y) for x in range(5) for y in range(5)]
```

```python

countries = {"India": ["Bangalore", "Chennai", "Delhi", "Kolkata"],
             "USA": ["Dallas", "New York", "Chicago"],
             "China": ["Bejing", "Shaingai"]
             }

# Get the list of Country and City in a tuple
# [("India", "Bangalore"),("India", "Chennai"),("India", "Delhi"),
# ("India", "Kolkata"),("USA", "Dallas"), ("USA", "New York"),
# ("USA", "Chicago"), ("China", "Bejing"), ("China", "Shaingai")]

l = [(country, city) for country, cities in countries.items() for city in cit
```