DSM_1Feb

February 2, 2023

1 DSM Class 1st Feb, 2023

1.1 Quick Coding Notes

1.1.1 Written by: Anshuman Rath# DSM_1st Feb

[1]:
```
name = "Data science masters"
name.swapcase() # swaps case
```

[1]:
```
'dATA SCIENCE MASTERS'
```

[2]:
```
name.title() # changes first letter of each word to Upper Case
```

[2]:
```
'Data Science Masters'
```

[3]:
```
msg = "hello Everyone"
msg.capitalize() # makes only first letter capital
```

[3]:
```
'Hello everyone'
```

[4]:
```
msg = "hello"
reversed(msg) # reverses the string and displays the address where it stores␣
 ↪the reversed string
```

[4]:
```
<reversed at 0x7f406419a770>
```

[8]:
```
"".join(reversed(msg)) # to display the stored reversed string
```

[8]:
```
'olleh'
```

[13]:
```
"Krish".join("Naik") # takes all items in an iterable and joins them into one␣
 ↪string
# every character of "Naik" is tryng to get concatenated with "Krish"
```

[13]:
```
'NKrishaKrishiKrishk'
```

[10]:
```
" ".join("abcd")
```

[10]:
```
'a b c d'
```

[12]:
```
" ".join(reversed("ant"))
# "ant" will get reversed to "tna" and every character of "tna" will␣
 ↪concatenate with " "
```

[12]:
```
't n a'
```

[15]:
```
" PWSkills ".join(reversed("ant"))
```

[15]:
```
't PWSkills n PWSkills a'
```

[17]:
```
list(reversed("ant")) # an example of typecasting
# generates list of characters in reverse order
```

[17]:
```
['t', 'n', 'a']
```

[18]:
```
# REMOVING CHARACTERS FROM THE END OF THE STRING
```

[30]:
```
string_a = " pw skills "
string_a.strip() # remove the whitespace from the beginning and at the end of␣
 ↪the string
```

[30]:
```
'pw skills'
```

[25]:
```
string_b = "chelloc"
string_b.strip("c")
# can also remove the given characters from the beginning and the end of the␣
 ↪original string
```

[25]:
```
'hello'
```

[29]:
```
string_c = "  Hello everyone "
string_c.lstrip() # removes leading (left) white spaces
```

[29]:
```
'Hello everyone '
```

[28]:
```
string_c = "  Hello everyone "
string_c.rstrip() # removes trailing (right) white spaces
```

[28]:
```
'  Hello everyone'
```

[31]:
```
string_n = "Greeting to PW Skill"
string_n.replace("to","from")
# replaces one String (specified initially) with another String (specified␣
 ↪later)
```

[31]:
```
'Greeting from PW Skill'
```

[32]:
```
"hello \t world" # \t is considered as a part of the String
```

[32]:
```
'hello \t world'
```

[33]:
```
"hello \t world".expandtabs() # \t is considered as tab space
```

[33]:
```
'hello    world'
```

[35]:
```
str1 = "Welcome to pwskills.Welcome to Dat cience Masters"
# Replace Dat with Data and cience with Science
# Answer -
str1.replace("Dat","Data").replace("cience","Science")
```

[35]:
```
'Welcome to pwskills.Welcome to Data Science Masters'
```

[1]:
```
str1 = "Cars"
str1.isupper() # returns True if String is in Upper Case
```

[1]:
```
False
```

[2]:
```
str2 = "BUS"
str2.isupper()
```

[2]:
```
True
```

[3]:
```
str3 = "hello"
str3.islower() # returns True if String is in Lower Case
```

[3]:
```
True
```

[5]:
```
str4 = " Hi everyone "
str4.isspace() # returns True if String has spaces only
```

[5]:
```
False
```

[6]:
```
str5 = "   "
str5.isspace()
```

[6]:
```
True
```

[7]:
```
str6 = "Anshuman"
str6.endswith("n") # returns True if String ends with specified character
```

[7]:
```
True
```

[8]:
```
a = "abcd1234"
a.isalnum() # returns True if String has alphanumeric characters
```

[8]:
```
True
```

[10]:
```
# To count number of characters in a String using For Loop

string = "hello"
count = 0
for i in string:
    count=count+1
print(count)
```

```
5
```

[11]:
```
# To display all letters in a String

string = "hello"
for i in string:
    print(i)
```

```
h
e
l
l
o
```

[14]:
```
# To display all letters in a String w.r.t. indices

string = "hello"
for i in range(len(string)):
    print(i,"=",string[i])
```

```
0 = h
1 = e
2 = l
3 = l
4 = o
```

[18]:
```
# To display all letters in a String in reverse

string = "hello"
for i in range(len(string)-1,-1,-1): # last index (len(string)-1) to -1 in␣
 ↪reverse order (-1 = stepsize)
    print(string[i])
```

```
o
l
```

```
l
e
h
```

[17]:
```
# To display all letters in a String in reverse using while loop

ch = len(string)-1
while ch >=0:
    print(string[ch])
    ch=ch-1
```

```
o
l
l
e
h
```

[19]:
```
# To display all letters in a String in reverse (another solution)

for i in range(len(string)):
    print(string[len(string)-(i+1)])
```

```
o
l
l
e
h
```

[21]:
```
# Check if characters in String name are vowels or not

name = "pwskills"
vowels = "AaEeIiOoUu"
for ch in name:
    if ch in vowels:
        print(f"{ch} is a vowel")
    else:
        print(f"{ch} is not a vowel")
```

```
p is not a vowel
w is not a vowel
s is not a vowel
k is not a vowel
i is a vowel
l is not a vowel
l is not a vowel
s is not a vowel
```

2 LISTS

[22]:
```
a = ["Krish", "Naik", 2, 3]
type (a)
```

[22]:
```
list
```

[23]:
```
str1 = "hello"
list(str1) # since list is iterable, here it will display all characters in the␣
 ↪String
```

[23]:
```
['h', 'e', 'l', 'l', 'o']
```

[25]:
```
str2 = "PW Skills Data Science Masters Class"
list(str2.split(" ")) # will remove (split) the specified characters (" ")
```

[25]:
```
['PW', 'Skills', 'Data', 'Science', 'Masters', 'Class']
```

[26]:
```
str2.split(" ") # will also give the same output as the above (in List)
# this means the return type of split() is also list
```

[26]:
```
['PW', 'Skills', 'Data', 'Science', 'Masters', 'Class']
```

[28]:
```
list1 = ["pw","skills","data","science",1,2]
list1[0] # to access the element in list present in specified index
```

[28]:
```
'pw'
```

[41]:
```
lst1 = ["pw","skills","data","science","masters"]
lst1[::-2] # same functionality as string, this returns every 2nd element␣
 ↪starting from last index in reverse
```

[41]:
```
['masters', 'data', 'pw']
```

[34]:
```
lst1 = ["pw","skills","data","science","masters"]
lst1[-3::-2] # returns every 2nd element starting from -3 index ("data") in␣
 ↪reverse
```

[34]:
```
['data', 'pw']
```

[40]:
```
lst1 = ["pw","skills","data","science","masters"]
lst1[-2:-4:-2] # returns every 2nd element starting from -3 index ("data") to␣
 ↪index -4 (excluding index -4)
```

[40]:
```
['science']
```

[42]:
```
# CONCATENATION OF LISTS
```

[43]:
```
lst1 = ["pw","skills","data","science","masters"]
lst1 + ["class",23]
```

[43]:
```
['pw', 'skills', 'data', 'science', 'masters', 'class', 23]
```

[45]:
```
lst1 = ["pw","skills","data","science","masters"]
lst1 + [["class",23]]  # multi nested list (double sq brackets) - list within a list
```

[45]:
```
['pw', 'skills', 'data', 'science', 'masters', ['class', 23]]
```

[48]:
```
lst1 = ["pw","skills","data","science","masters"]
lst2 = lst1 + [["class",23]]
print(lst2)
```

```
['pw', 'skills', 'data', 'science', 'masters', ['class', 23]]
```

[52]:
```
# Suppose I want to print ['class', 23] from lst2

print(lst2[5])
```

```
['class', 23]
```

[53]:
```
# Suppose I want to print only ['class'] from lst2

print(lst2[5][0]) # class is in the 0th index of the multi nested list - ["class",2]
```

```
class
```

[55]:
```
print(lst2*2) # repeats list for 2 times
```

```
['pw', 'skills', 'data', 'science', 'masters', ['class', 23], 'pw', 'skills',
'data', 'science', 'masters', ['class', 23]]
```

[56]:
```
lst1 = ["pw","skills","data","science","masters"]
for element in lst1:
    print(element)
```

```
pw
skills
data
science
masters
```

[57]:
```
# check if science is present in lst1

if "science" in lst1:
```

```
    print("Present")
```

```
Present
```

[60]:
```
# check elements in a list

lst = [1,2,3,4]
print(4 in lst) # returns True
print(5 in lst) # returns False
```

```
True
False
```

[66]:
```
lst1 = ["Monkey", "Zebra", "Donkey", "Lion"]
print(max(lst1))  # returns and prints the alphabetically highest value (using␣
 ↪the concept of ASCII)
print(min(lst1))  # returns and prints the alphabetically lowest value (using␣
 ↪the concept of ASCII)
```

```
Zebra
Donkey
```

[67]:
```
lst2 = [1,3,2,9,8,6]
print(max(lst2)) # returns and print the max value
print(min(lst2)) # returns and print the min value
```

```
9
1
```

[68]:
```
## APPEND FUNCTION
```

[75]:
```
lst = ["Data","Science"]
lst.append("Masters") # adds the element and updates the list (in place␣
 ↪operation)
print(lst)
```

```
['Data', 'Science', 'Masters']
```

[74]:
```
lst = ["Data","Science","Masters"]
lst.pop() # removes the last element and updates the list (in place operation)
print(lst)
```

```
['Data', 'Science']
```

[76]:
```
lst = ["Data","Science","Masters"]
lst.pop(1) # removes the element at index 1 and updates the list
print(lst)
```

```
['Data', 'Masters']
```

[78]:
```
lst = ["Data","Science","Masters"]
removed_element = lst.pop(0) # returns removed element
print(removed_element)
```

```
Data
```

[80]:
```
lst = ["Data","Science","Masters"]
lst[100] # will give Index Error
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[80], line 2
      1 lst = ["Data","Science","Masters"]
----> 2 lst[100] # will give Index Error

IndexError: list index out of range
```

[82]:
```
lst = ["q","e","f","s","t","u"]
lst[::-1] # reverses the list
```

[82]:
```
['u', 't', 's', 'f', 'e', 'q']
```

[83]:
```
lst = ["q","e","f","s","t","u"]
lst.reverse() # reverses the list and updates it (in place operation)
print(lst)
```

```
['u', 't', 's', 'f', 'e', 'q']
```

[85]:
```
lst = ["q","e","f","s","t","u"]
lst.sort() # sorts (ascending order) the list and updates it (in place␣
 ↪operation)
print(lst)
```

```
['e', 'f', 'q', 's', 't', 'u']
```

[90]:
```
lst = ["q","e","f","s","t","u"]
lst.sort(reverse = True) # sorts (descending order) the list and updates it (in␣
 ↪place operation)
print(lst)
```

```
['u', 't', 's', 'q', 'f', 'e']
```

[88]:
```
# To add multi nested list to a list, use append
lst = ["Data", "Science"]
lst.append(["Masters", "Class"])
```

```
print(lst)
```

```
['Data', 'Science', ['Masters', 'Class']]
```

[89]:
```
# To add more than one element to a list (without multi nesting), use extend
lst = ["Data", "Science"]
lst.extend(["Masters", "Class"])
print(lst)
```

```
['Data', 'Science', 'Masters', 'Class']
```

3 NESTED LIST

[91]:
```
lst1 = [1,2,3]
lst2 = [4,5,6]
lst3 = [7,8,9]

# Make a list of list to form a matrix
matrix = [lst1,lst2,lst3]
print(matrix)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

[92]:
```
# return 6 from the created matrix
matrix [1][2] # matrix[1] = [4,5,6] and [2] of it contains 6
```

[92]:
```
6
```

[93]:
```
# return 8 and 9 from matrix
matrix [2][1:]
```

[93]:
```
[8, 9]
```

4 List Comprehension

[101]:
```
[i for i in range(20)] # creates a list having numbers from 0 to 20-1 (19)
```

[101]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

[100]:
```
[i for i in range (2,20)] # creates a list having numbers from 2 to 20-1 (19)
```

[100]:
```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

[104]:
```
# To print even numbers from 1 to 9 and blank space for odd numbers
[i if i%2 == 0 else " " for i in range(1,10)]
```

[104]:
```
[' ', 2, ' ', 4, ' ', 6, ' ', 8, ' ']
```

[106]:
```
lst = [1,2,3,4,5,6,7,8]
# Find sum of even numbers and even numbers from the above list

sum_even = 0
sum_odd = 0
for i in lst:
    if (i%2==0):
        sum_even = sum_even+i
    else:
        sum_odd = sum_odd+i
print(f"Sum of even numbers = {sum_even}")
print(f"Sum of even numbers = {sum_odd}")
```

```
Sum of even numbers = 20
Sum of even numbers = 16
```

[108]:
```
lst = [1,2,3,4,5,6,7,8]
# Find sum of even numbers and even numbers from the above list using List␣
 ↪Comprehension

sum_even = sum([num for num in lst if num%2 == 0]) # sum() finds sum
sum_odd = sum([num for num in lst if num%2 !=0])
print(f"Sum of even numbers = {sum_even}")
print(f"Sum of even numbers = {sum_odd}")
```

```
Sum of even numbers = 20
Sum of even numbers = 16
```

[111]:
```
lst = [1,2,3,4,5,6,7,8,9,10]
# Find squares of all the numbers using List Comprehension

[num**2 for num in lst] # **2 means number to the power of 2
```

[111]:
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[112]:
```
lst = [-2,-1,0,1,2,3,4]
# Create a list of positive numbers from the list

[num for num in lst if num>0]
```

[112]:
```
[1, 2, 3, 4]
```

[113]:
```
lst = ["apple","banana","cherry","date"]
# Create a list of only first letters of the words in the list
```

```
[word[0] for word in lst]
```

[113]:
```
['a', 'b', 'c', 'd']
```

[115]:
```
temp_celcius = [0,10,20,30,40,50]
# Convert the temperatures in celcius given in list to farenheit

[(temp_farenheit*9/5)+32 for temp_farenheit in temp_celcius]
```

[115]:
```
[32.0, 50.0, 68.0, 86.0, 104.0, 122.0]
```

[117]:
```
lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
# flatten the above list of lists into a single list

[num for sublist in lst for num in sublist]
```

[117]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[135]:
```
# ASSIGNMENT 1
# Given numbers = [1,2,3,4,5,6,7,8,9,10]
# Create a list of all prime numbers from the list (using Code and List␣
 ↪Comprehension)
```

[134]:
```
# ANSWER TO ASSIGNMENT 1 (USING CODE) -

numbers = [1,2,3,4,5,6,7,8,9,10]
prime_list = []
for prime in numbers:
    if (prime > 1 and all(prime % y !=0 for y in range(2,prime))):
        prime_list=prime_list+[prime]
print(prime_list)
```

```
[2, 3, 5, 7]
```

[130]:
```
# ANSWER TO ASSIGNMENT 1 (USING LIST COMPREHENSION) -

numbers = [1,2,3,4,5,6,7,8,9,10]
prime_list = [prime for prime in numbers if prime !=1 and all(prime % y !=0 for␣
 ↪y in range(2,prime))]
print(prime_list)
```

```
[2, 3, 5, 7]
```

[137]:
```
# ASSIGNMENT 2
# Given numbers = [1,2,3,5,]
# Create a list of all the possible combinations of 2 elements from the list␣
 ↪(using Code and List Comprehension)
```

[151]:
```
# ANSWER TO ASSIGNMENT 2 (USING CODE) -

numbers = [1,2,3,4,5]
combinations = []
for a in range(1,len(numbers)):
    for b in range(a+1,len(numbers)+1):
        combinations.append([a,b])
print(combinations)
```

```
[[1, 2], [1, 3], [1, 4], [1, 5], [2, 3], [2, 4], [2, 5], [3, 4], [3, 5], [4, 5]]
```

[150]:
```
# ANSWER TO ASSIGNMENT 2 (USING LIST COMPREHENSION) -

numbers = [1,2,3,4,5]
combinations = [[a,b] for a in range(1,len(numbers)) for b in␣
 ↪range(a+1,len(numbers)+1)]
print(combinations)
```

```
[[1, 2], [1, 3], [1, 4], [1, 5], [2, 3], [2, 4], [2, 5], [3, 4], [3, 5], [4, 5]]
```