Python Data Structures

Types of Data Structures:

- `List | Tuple`
- `Set | Dictionary`

Data Structures type	Mutable	Comments	Indexing	Ordered	Duplicacy
tuple ()	immutable	immutable version of list	possible	yes	allowed
list []	mutable	-	possible	yes	allowed
set {}	mutable	-	not	no	not
dict {key:value}	mutable	-	possible	no	not

💫 `immutable => can't be changed`
💫 `mutable => can be changed`

```
In [ ]:
```

Lists

Create an empty list with the name ‘a’, print the value of a and type(a).

```
In [1]: # create empty list, name it 'a'
        a = []
```

```
In [2]: # print the value of a
        a
```

```
Out[2]: []
```

```
In [3]: # print the type of a
        type(a)
```

```
Out[3]: list
```

Create a list , languages = ['R','Python', 'SAS', 'Scala', 42],

In [4]:
```
 languages = ['R','Python', 'SAS', 'Scala', 42]
```

Print the number of elements in the list

In [5]:
```
type(languages)
```

Out[5]:
```
list
```

Using for loop iterate and print all the elements in the list

In [6]:
```


for i in languages:
    print(i)
```

```
R
Python
SAS
Scala
42
```

Select the second item, 'Python' and store it in a new variable named 'temp'

In [7]:
```
temp = languages[1]
```

Print the value of temp and type(temp)

In [8]:
```

print(temp)
type(temp)
```

```
Python
```

Out[8]:
```
str
```

Append the element 'Java' in the list

In [9]:
```

languages.append("java")
```

Remove the element 42 from the list and print the list

In [10]:
```
languages.remove(42)

print(languages)
```

```
['R', 'Python', 'SAS', 'Scala', 'java']
```

Create a list, colors = [‘Red’, ‘Blue’, ‘White’]

In [11]:
```
colors = ['Red', 'Blue', 'White']
```

Append the element 'Black' to colors

In [12]:
```
colors.append('Black')
```

Append the color 'Orange' to second position (index=1) and print the list

In [13]:
```
colors.insert(1,'Orange')

colors
```

Out[13]:
```
['Red', 'Orange', 'Blue', 'White', 'Black']
```

Print the list

In [14]:
```
colors
```

Out[14]:
```
['Red', 'Orange', 'Blue', 'White', 'Black']
```

Create another list, colors2 = [‘Grey’, ‘Sky Blue’]

In [15]:
```
colors2 = ['Grey', 'Sky', 'Blue']
```

Add the elements of colors2 to colors using extend function in the list

In [16]:
```
colors.extend(colors2)
colors
```

Out[16]:
```
['Red', 'Orange', 'Blue', 'White', 'Black', 'Grey', 'Sky', 'Blue']
```

Print len of colors and its elements

In [17]:
```
len(colors)
```

Out[17]:
```
8
```

Sort the list and print it.

In [18]:
```
colors.sort()
colors
```

Out[18]:
```
['Black', 'Blue', 'Blue', 'Grey', 'Orange', 'Red', 'Sky', 'White']
```

Create a string, sent = ‘Coronavirus Caused Lockdowns Around The World.”

In [19]:
```
sent = "Coronavirus Caused Lockdowns Around The World."
```

Use split function to convert the string into a list of words and save it in variable words and print the same

In [20]:
```

words = sent.split(', ')
words
```

Out[20]:
```
['Coronavirus Caused Lockdowns Around The World.']
```

Convert each word in the list to lower case and store it in variable words_lower. Print words_lower

In [21]:
```
words_lower = [i.lower() for a,i in enumerate(words)]
print(words_lower)
```

```
['coronavirus caused lockdowns around the world.']
```

Check whether ‘country’ is in the list

In [22]:
```
if words == 'country':
    print('country is present in list')
else:
    print('county is not present in list')
```

```
county is not present in list
```

Remove the element ‘the’ from the list and print the list.

In [24]:
```
words.pop(0)
```

Out[24]:
```
'Coronavirus Caused Lockdowns Around The World.'
```

Select the first 4 words from the list words_lower using slicing and store them in a new variable x4

In [ ]:
```
x4 = words_lower[0:5]
```

In [ ]:
```
# print x4
x4
```

Convert the list of elements to single string using join function and print it

In [ ]:
```

```

Sets

Create stud_grades = ['A','A','B','C','C','F']

In [ ]:
```
stud_grades = ['A','A','B','C','C','F']
```

Print the len of stud_grades

In [ ]:
```
stud_grades
```

Create a new variable, stud_grades_set = set(stud_grades)

In [ ]:
```
stud_grades_set = set(stud_grades)
```

Print stud_grades_set.

In [ ]:
```
stud_grades_set
```

print the type of stud_grades and stud_grades_set and print their corresponding elements. Try to understand the difference between them.

In [ ]:
```
type(stud_grades_set)
```

Add a new element ‘G’ to stud_grades_set

In [ ]:
```
stud_grades_set.add("G")
```

Add element 'F' to stud_grades_set. and print it.

In [ ]:
```
stud_grades_set.add("F")

stud_grades_set
```

!!Did you notice? set doesn't add an element if it's already present in it, unlike lists.

Remove ‘F’ from stud_grades_set

In [ ]:
```
stud_grades_set.remove("F")
```

Print the elements and the length of stud_grades_set

In [ ]:
```
len( stud_grades_set)
```

Create colors = [‘red’,’blue’,’orange’], and fruits = [‘orange’,’grapes’,’apples’]

In [ ]:
```
colors = ['red','blue','orange']

fruits = ['orange','grapes','apples']
```

Print color and fruits

In [ ]:
```

print(colors)

print(fruits)
```

Create colors_set, and fruits_set. (using set()) and print them

In [ ]:
```
colors_set = set(colors)
fruits_set = set(fruits)

print(colors_set)

print(fruits_set)
```

Find the union of both the sets.

In [ ]:
```
colors_set.union(fruits_set)
```

Find the intersection of both the sets

In [ ]:
```
colors_set.intersection(fruits_set)
```

Find the elements which are Fruits but not colors (using set.difference())

In [ ]:
```
fruits_set.difference(colors_set)
```

In [ ]:

TUPLES

Create temp = [17, ’Virat’, 50.0]

In [ ]:
```
temp = [17, 'Virat', 50.0]
```

Iterate through temp and print all the items in temp

In [ ]:
```
temp
```

replace first element with 11 in temp

In [ ]:
```
temp[0] = 11
```

Set temp1 = tuple(temp)

In [ ]:
```
temp1 = tuple(temp)
```

Iterate through temp1 and print all the items in temp1.

In [ ]:
```

print(temp1)
```

replace first element with 17 in temp1

In [ ]:
```
temp1[0] = 17
```

Oops!! You got an error. Hey Don't worry! Its because Once a tuple is created, you cannot change its values unlike list.

Create city = ("Bangalore", 28.9949521, 72)

In [ ]:
```
city = ("Bandlore", 28.9949521, 72)
```

Print first element of city

In [ ]:
```
print(city[0])
```

Create city2 = (‘Chennai’, 30.01, 74)

In [ ]:
```
city2 = ('Chennai', 30.01,74)
```

Create cities which consist of city and city2

In [ ]:
```
cities = city,city2
```

Print cities

In [ ]:
```
print(cities)
```

Print type of first element in cities

In [ ]:
```
print(type(cities[0]))
```

print the type of cities

In [ ]:
```
print(type(cities))
```

Hey that implies you made a nested tuples!!

DICTIONARY

Create a dictionary d = {"actor":"amir","animal":"cat","earth":2,"list":[23,32,12]}

In [ ]:
```
d = {"actor":"amir","animal":"cat","earth":2,"list":[23,32,12]}
```

Print the value of d[0]

In [ ]:
```
print(d[0])
```

Oops!! again an error. again a fun fact. Dictionary return the value for key if key is in the dictionary, else throws KeyError and we don't have key 0 here :(

Store the value of d[‘actor’] to a new variable actor.

In [ ]:
```
d['actor'] = 'actor'
```

Print the type of actor

In [ ]:
```
print(type('actor'))
```

Store the value of d['list'] in new variable List.

In [ ]:
```
List = d['list']
```

Print the type of List.

In [ ]:
```
print(List)
```

Create d1 = { 'singer' : 'Kr$na' , 'album': 'Still here', 'genre' : 'hip-hop'}

In [ ]:
```
d1 = {'singer':'Kr$na','album':'Still here','genre':'hip-hop'}
```

Merge d1 into d.

In [ ]:
```
def Merge(d1,d):
    return(d1.update(d))
d = {"actor":"amir","animal":"cat","earth":2,"list":[23,32,12]}
d1 = {'singer':'Kr$na','album':'Still here','genre':'hip-hop'}

print(Merge(d,d1))

print(d1)
```

print d

In [ ]:
```
d
```

Print all the keys in d

In [ ]:
```
print(d.keys)
```

Print all the values in d

In [ ]:
```
print(d.values)
```

Iterate over d, and print each key, value pair as this - (actor ----> amir)

In [ ]:
```
d
```

count the number of occurences of charachters in string named "sent" using dictionary and print the same.

In [ ]:
```
sent
```

In [ ]:
```
```

In [ ]:
```
```