tuple, set, dict

In [ ]:
```python
# A tuple is a collection which is ordered and unchangeable.
# In Python tuples are written with round brackets.
t1 = tuple()
t2 = ()
t3 = (1, 2, 3)
```

In [ ]:
```python
# compare types of all three variables and check that they are equal
type(t1) == type(t2) == type(t3)
```

In [ ]:
```python
# create new tuple winter
winter = ('dec', 'jan', 'feb')
```

In [ ]:
```python
# access to tuples element by it's index, same as in lists
winter[1]
```

In [ ]:
```python
# tuples are unchengable, it you cannot add or delete elements from it
# error
winter += ('mar')
```

In [ ]:
```python
# tuples are unchengable, it you cannot add or delete elements from it
# error
winter[1] = 'mar'
```

In [ ]:
```python
# list object can be converted to tuple
# let's create random list object
my_list = [1, 2, 3]
type(my_list)
```

In [ ]:
```python
# convert list object to a tuple using tuple() function and check type
my_tuple = tuple(my_list)
print (my_tuple, type(my_tuple))
```

In [ ]:
```python
# use keyword "in" to check if a value lies inside tupple()
# True
'jan' in winter
```

In [ ]:
```python
# False
'mar' in winter
```

In [ ]:
```python
# use len() function to count nummber of elements inside tuple
print(len(winter))
```

In [ ]:
```python
# tuples has 2 main methods (build-in own atribute functions)
new_tuple = (1, 1,2, 1,2,3, 1,2,3,4, 1,2,3,4,5)
new_tuple
```

In [ ]:
```python
# .count() is to count specific element in the tuple
new_tuple.count(2)
```

In [ ]:
```python
# the answer is 0 because there is no 10 number in new_tuple
print (new_tuple.count(10), 10 in new_tuple)
```

In [ ]:
```python
# .index() is to find first place of the element in the tuple
# because tuples are ordered collections
# recall that programmers start counting from 0
new_tuple.index(3)
```

In [ ]:
```python
# if element doesn't belong to the tuple,
# then we will get an error
# error
print (new_tuple.index(22))
```

In [ ]:
```python
# function len() is used to length of tuple
len(new_tuple), new_tuple
```

In [ ]:
```python
# when it is good to use tuples ?
```

In [ ]:
```python
# A set is a collection which is unordered and unindexed and changable.
# In Python sets are written with curly brackets {} or using set() function.
s1 = {}
s2 = {1,2,3}
s3 = set()
s4 = set((1,2,3))
```

In [ ]:
```python
# Note: Sets are unordered, so you cannot be sure in which order the items will appear.
```

In [ ]:
```python
# it is impossible to access items in a set by referring to an index,
# since sets are unordered the items has no index.
# but it is possible to check if an element belongs to the set
1 in s4
```

In [ ]:
```python
0 in s4
```

In [ ]:
```python
# as sets are changeble we can add new items to it
# .add() to add single element
s4.add(4)
s4
```

In [ ]:
```python
# .update() to add another set
s4.update({8,9,10})
s4
```

```
In [ ]: # the distinguish of sets from tuples and list is
        # that all values stored in sets only one time
        s4.add(1)
        s4.update({2,2,2,2})
        s4
```

```
In [ ]: # in contrast, in the lists if we use .append() and .extend(),
        # then all elements will be added to the end of the list
        list4 = [1,2,3]
        print('initial list :', list4)
        list4.append(1)
        list4.extend([2,2,2,2])
        print('after .append() and .extend() :',list4)
```

```
In [ ]: # tuples as you remember are unchangable
```

```
In [ ]: # len() function will count number of all elements in the set
        len(s4), s4
```

```
In [ ]: # to remove element we use .remove() or .discard() methods
        # the difference between them is that .remove() will rise and error
        # if element is not in the set
        # and .discard() will not
        10 in s4, s4.remove(10), s4
```

```
In [ ]: # error
        s4.remove(20)
```

```
In [ ]: # not error
        s4.discard(20)
```

```
In [ ]: # .union() create a set containing the union of sets, but keep unchanged arguments,
        # in contrast to .update()
        weekdays = {1,2,3,4,5}
        weekends = {0, 6}
        week = weekdays.union(weekends)
        week
```

```
In [ ]: # note if you see ordered set it means that you use new version of python
        # in early versions the output of calling set was in random order
```

```
In [ ]: # sets has built-in methods that equal to mathematical to check properties of a sets
        week.difference(weekends)
```

```
In [ ]: week.issuperset(weekdays)
```

```
In [ ]: week.intersection(weekdays)
```

```
In [ ]: # draw circles in the board as examples
```

```
In [ ]: # A dictionary is a collection which is unordered, changeable and indexed.
        # In Python dictionaries are written with curly brackets, and they have keys and values.
        d1 = dict()
        d2 = dict(key1="value1", key2='value2')
        d3 = {'key3': 'value3', 'key4':'value4'}
        d1, d2, d3
```

```
In [ ]: student = {'name': 'Bob', 'age':18, 'courses': ['intro to cs', 'intro to prog'], 'city':'NS' }
        student
```

```
In [ ]: # You can access the items of a dictionary by referring to its key name, inside square brackets:
        student['name']
```

```
In [ ]: # You can change the value of a specific item by referring to its key name:
        student['age'] = 23
        student
```

```
In [ ]: # Adding an item to the dictionary is done by using a new index key and assigning a value to it:
        student['graduated'] = True
        student
```

```
In [ ]: # .keys() method to show only keys of dict
        student.keys()
```

```
In [ ]: # .values method to show only values of the dict
        student.values()
```

```
In [ ]: # .items() to return pairs of key and value
        student.items()
```

```
In [ ]: # To determine if a specified key is present in a dictionary use the in keyword:
        'city' in student
```

```
In [ ]: # To determine how many items (key-value pairs) a dictionary has, use the len() method.
        len(student)
```

```
In [ ]: # A dictionary can also contain many dictionaries, this is called nested dictionaries.
        student['certificates'] = {'BSc':"buisiness in it", 'MBA':'management in it'}
        student
```

```
In [ ]: student['certificates']['MBA']
```

```
In [ ]: # delete items in dicts
        # The del keyword removes the item with the specified key name:
        del student['city']
        student
```

```
In [ ]: # practice
```