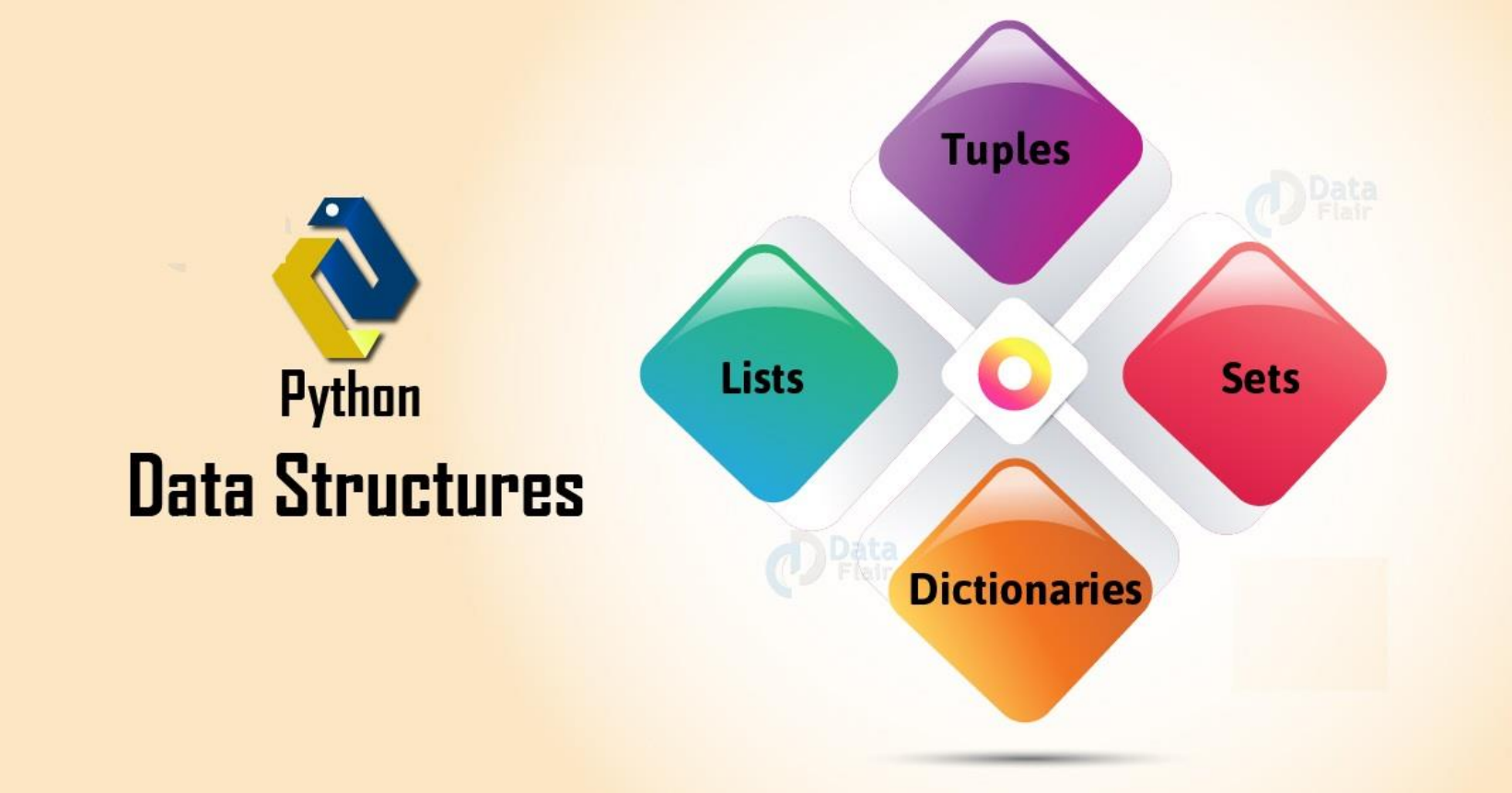
Tuples, Dictionaries, and Sets

Introduction

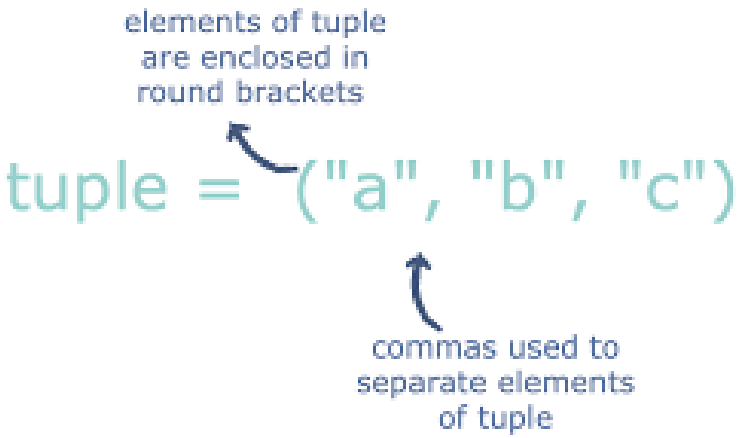
- Data structure as a way of organizing and storing data such that we can access and modify it efficiently.



tuple

- Tuples are similar to lists, except tuples are immutable.

- Tuple Items are :
  - Ordered
  - Unchangeable
  - Allow Duplicates



tuple

```
my_list = [1, 2, 3, 4, 5, 6, 7]     # Make a list
my_tuple = (1, 2, 3, 4, 5, 6, 7)    # Make a tuple
print('The list:', my_list)         # Print the list
print('The tuple:', my_tuple)       # Print the tuple
print('The first element in the list:', my_list[0])   # Access an element
print('The first element in the tuple:', my_tuple[0]) # Access an element
print('All the elements in the list:', end=' ')
for elem in my_list:                # Iterate over the elements of a list
    print(elem, end=' ')
print()
print('All the elements in the tuple:', end=' ')
for elem in my_tuple:                # Iterate over the elements of a tuple
    print(elem, end=' ')
print()
print('List slice:', my_list[2:5])     # Slice a list
print('Tuple slice:', my_tuple[2:5])   # Slice a tuple
print('Try to modify the first element in the list . . .')
my_list[0] = 9          # Modify the list
print('The list:', my_list)
print('Try to modify the first element in the list . . .')
my_tuple[0] = 9         # Is tuple modification possible?
print('The tuple:', my_tuple)
```

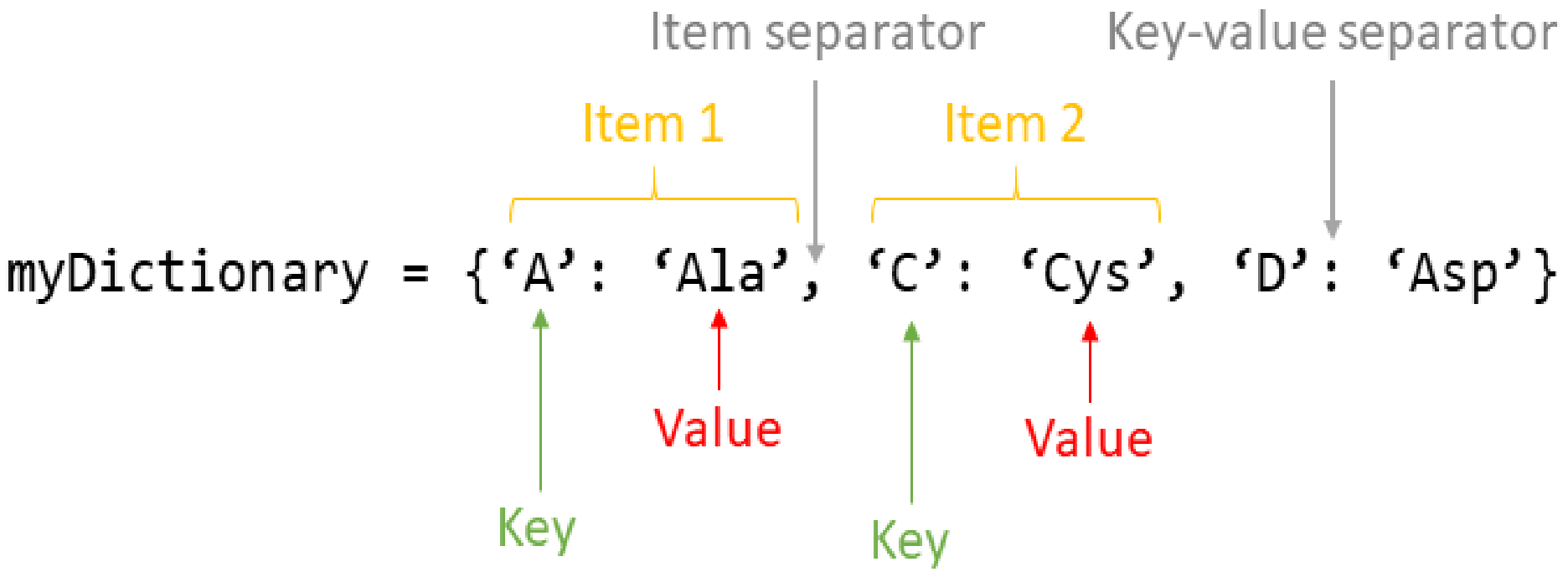
```
The list: [1, 2, 3, 4, 5, 6, 7]
The tuple: (1, 2, 3, 4, 5, 6, 7)
The first element in the list: 1
The first element in the tuple: 1
All the elements in the list: 1 2 3 4 5 6 7
All the elements in the tuple: 1 2 3 4 5 6 7
List slice: [3, 4, 5]
Tuple slice: (3, 4, 5)
Try to modify the first element in the list . . .
The list: [9, 2, 3, 4, 5, 6, 7]
Try to modify the first element in the list . . .
Traceback (most recent call last):
  File "tupletest.py", line 26, in <module>
    main()
  File "tupletest.py", line 22, in main
    my_tuple[0] = 9
TypeError: 'tuple' object does not support item assignment
```

lists versus tuples

Feature	List	Tuple
Mutability	mutable	immutable
Creation	`lst = [i, j]`	`tpl = (i, j)`
Element access	`a = lst[i]`	`a = tpl[i]`
Element modification	`lst[i] = a`	*Not possible*
Element addition	`lst += [a]`	*Not possible*
Element removal	`del lst[i]`	*Not possible*
Slicing	`lst[i:j:k]`	`tpl[i:j:k]`
Slice assignment	`lst[i:j] = []`	*Not possible*
Iteration	`for elem in lst:...`	`for elem in tpl:...`

Dictionary

- **A Python dictionary is an associative container which permits access based on a key, rather than an index.**



Dictionary

- **The keys in a dictionary may have different types**
- **The values in a dictionary may have different types**
- **The values in a dictionary may be mutable objects**
- **The order of key:value pairs in a dictionary are independent of the order of their insertion into the dictionary.**

```
d = {'Fred': 44, 'Ella': 39, 'Owen': 40, 'Zoe': 41}
print(d)
```

Despite the order supplied during d's initialization, on one system the code above prints

```
{'Fred': 44, 'Ella': 39, 'Zoe': 41, 'Owen': 40}
```

Dictionary method

Method	Description
dict.clear()	Removes all the key-value pairs from the dictionary.
dict.copy()	Returns a shallow copy of the dictionary.
dict.fromkeys()	Creates a new dictionary from the given iterable (string, list, set, tuple) as keys and with the specified value.
dict.get()	Returns the value of the specified key.
dict.items()	Returns a dictionary view object that provides a dynamic view of dictionary elements as a list of key-value pairs. This view object changes when the dictionary changes.
dict.keys()	Returns a dictionary view object that contains the list of keys of the dictionary.
dict.pop()	Removes the key and return its value. If a key does not exist in the dictionary, then returns the default value if specified, else throws a KeyError.
dict.popitem()	Removes and return a tuple of (key, value) pair from the dictionary. Pairs are returned in Last In First Out (LIFO) order.
dict.setdefault()	Returns the value of the specified key in the dictionary. If the key not found, then it adds the key with the specified defaultvalue. If the defaultvalue is not specified then it set None value.
dict.update()	Updates the dictionary with the key-value pairs from another dictionary or another iterable such as tuple having key-value pairs.
dict.values()	Returns the dictionary view object that provides a dynamic view of all the values in the dictionary. This view object changes when the dictionary changes.

Dictionary method

```
dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

print(dict)
print("----------------------------------------")

print(dict.keys())
print("----------------------------------------")

print(dict.values())
print("----------------------------------------")

print(dict.items())
print("----------------------------------------")

dict.update(Maggie=19)
print(dict)
print("----------------------------------------")

print(dict.pop('Tiffany'))
```

Dictionary method

```
{'Tim': 18, 'Charlie': 12, 'Tiffany': 22, 'Robert': 25}
----------------------------------------
dict_keys(['Tim', 'Charlie', 'Tiffany', 'Robert'])
----------------------------------------
dict_values([18, 12, 22, 25])
----------------------------------------
dict_items([('Tim', 18), ('Charlie', 12), ('Tiffany', 22), ('Robert', 25)])
----------------------------------------
{'Tim': 18, 'Charlie': 12, 'Tiffany': 22, 'Robert': 25, 'Maggie': 19}
----------------------------------------
22
```

Dictionary

- d = {"one": 1, "two": 2, "three": 3}

```
# Get
d['one']   # => 1
d['five'] # raises KeyError


# Set
d['two'] = 22   # Modify an existing key
d['four'] = 4   # Add a new key
```

Example:

```
1  items = [2,4,5,4,7,4,6,2,5,2]
2
3  stats = {}
4  for i in items:
5      if i in stats:
6          stats[i] += 1
7      else:
8          stats[i] = 1
9
10 print(stats)
```

```
{2: 3, 4: 3, 5: 2, 7: 1, 6: 1}
```

sets

- Python provides a data structure that represents a mathematical set.
- As with mathematical sets, we use curly braces {} in Python code to enclose the elements of a literal set.
- Unlike Python lists, sets are unordered and may contain no duplicate elements.

```
>>> S = {10, 3, 7, 2, 11}
>>> S
{2, 11, 3, 10, 7}
>>> T = {5, 4, 5, 2, 4, 9}
>>> T
{9, 2, 4, 5}
```

```
>>> L = [10, 13, 10, 5, 6, 13, 2, 10, 5]
>>> S = set(L)
>>> L
[10, 13, 10, 5, 6, 13, 2, 10, 5]
>>> S
{10, 2, 13, 5, 6}
```

```
>>> S = {x**2 for x in range(10)}
>>> S
{0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
```

Set operations

Operation	Mathematical Notation	Python Syntax	Result Type	Meaning
Union	$A \cup B$	`A | B`	`set`	Elements in A or B or both
Intersection	$A \cap B$	`A & B`	`set`	Elements common to both A and B
Set Difference	$A - B$	`A - B`	`set`	Elements in A but not in B
Symmetric Difference	$A \oplus B$	`A ^ B`	`set`	Elements in A or B, but not both
Set Membership	$x \in A$	`x in A`	`bool`	x is a member of A
Set Membership	$x \notin A$	`x not in A`	`bool`	x is not a member of A
Set Equality	$A = B$	`A == B`	`bool`	Sets A and B contain exactly the same elements
Subset	$A \subseteq B$	`A <= B`	`bool`	Every element in set A also is a member of set B
Proper Subset	$A \subset B$	`A < B`	`bool`	A is a subset B, but B contains at least one element not in A

```
>>> S = {2, 5, 7, 8, 9, 12}
>>> T = {1, 5, 6, 7, 11, 12}
>>> S | T
{1, 2, 5, 6, 7, 8, 9, 11, 12}
>>> S & T
{12, 5, 7}
>>> 7 in S
True
>>> 11 in S
False
```

set vs list

```
Set: <class 'set'>  List: <class 'list'>
List elapsed: 44.99767441164282
Set elapsed: 0.48652052551967984
```

```
#  Data structure size
size = 1000
#  Make a big set
S = {x**2 for x in range(size)}
#  Make a big list
L = [x**2 for x in range(size)]
#  Verify the type of S and L
print('Set:', type(S), ' List:', type(L))
from time import perf_counter
#  Search size
search_size = 1000000
#  Time list access
start_time = perf_counter()
for i in range(search_size):
    if i in L:
        pass
stop_time = perf_counter()
print('List elapsed:', stop_time - start_time)

#  Time set access
start_time = perf_counter()
for i in range(search_size):
    if i in S:
        pass
stop_time = perf_counter()
print('Set elapsed:', stop_time - start_time)
```

Example

```
Input : Dict = ["go","bat","me","eat","goal","boy", "run"]
        arr = ['e','o','b', 'a','m','g', 'l']
Output : go, me, goal.
```

```python
def charCount(word):
    dict = {}
    for i in word:
        dict[i] = dict.get(i, 0) + 1
    return dict


def possible_words(lwords, charList):
    for word in lwords:
        flag = 1
        chars = charCount(word)
        for key in chars:
            if key not in charList:
                flag = 0
            else:
                if charList.count(key) != chars[key]:
                    flag = 0
        if flag == 1:
            print(word)

input = ['go', 'bat', 'me', 'eat', 'goal', 'boy', 'run']
characterList = ['e', 'o', 'b', 'a', 'm', 'g', 'l']
possible_words(input, characterList)
```