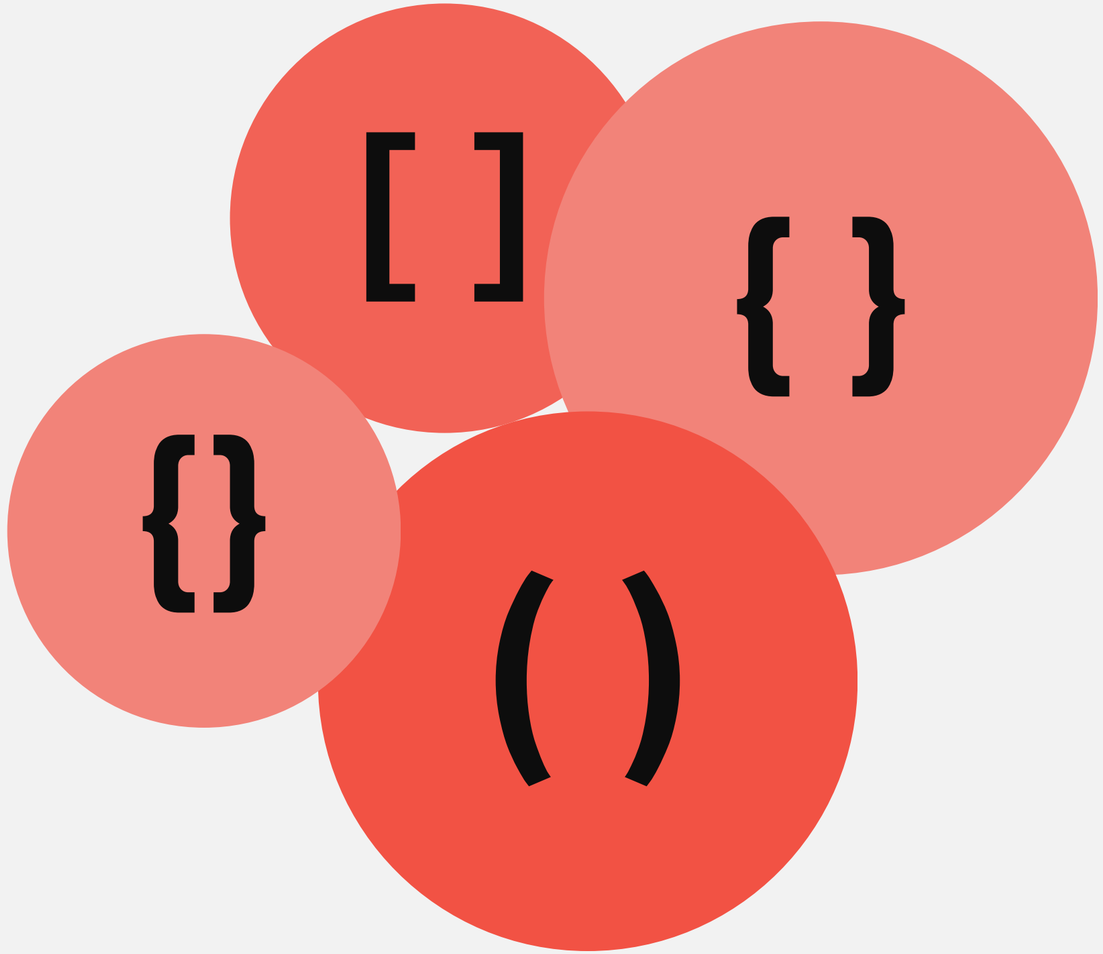
PYTHON DATA CONTAINERS 2 AND FUNCTIONS IN PYTHON



Rishabh IO
linkedin/rishabhio

Set Syntax

If left empty, curly braces will initialise a dict

```
set_identifier = {'a'}
set_identifier = set()
```

non-empty curly braces initialize a set

any valid identifier in Python

all elements are unique

Set Methods

Code Example

add(element): Adds an element to set

```
fruits = {"apple", "banana", "cherry"}
fruits.add("orange")
print(fruits)  # Output: {'apple', 'banana',
'cherry', 'orange'}
type(fruits) # <class 'set'>

fruits.add("apple") # What happens now?
```

Set Methods

Code Example

```
# clear(): Removes all elements from the set
fruits = {"apple", "banana", "cherry"}
fruits.clear()
print(fruits)  # Output: set()

# copy(): Returns a shallow copy of the set
fruits = {"apple", "banana", "cherry"}
fruits_copy = fruits.copy()
print(fruits_copy)  # Output: {'apple', 'banana', 'cherry'}
```

Set Methods

Code Example

```
# difference(other_set)


# Returns a new set containing elements
that are in the set but not in the other_set

set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
diff_set = set1.difference(set2)
print(diff_set)  # Output: {1, 2, 3}
```

Set Methods

Code Examples

```
# difference_update(other_set)

Updates the set, removing elements that
are also in the other_set

set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
set1.difference_update(set2)
print(set1)  # Output: {1, 2, 3}
```

Set Methods

Code Examples

```
# discard(element)
# Removes an element from the set if it is present

fruits = {"apple", "banana", "cherry"}
fruits.discard("banana")
print(fruits)  # Output: {'apple', 'cherry'}
```

Set Methods

Code Examples

```
# intersection(other_set)

Returns a new set containing elements that
are common to both the set and the
other_set.

set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
intersection_set = set1.intersection(set2)
print(intersection_set)  # Output: {4, 5}
```

Set Methods

Code Examples

```
# intersection_update(other_set)

# Updates the set, keeping only elements
that are also in the other_set.

set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
set1.intersection_update(set2)
print(set1)  # Output: {4, 5}
```

Set Methods

Code Examples

```
# isdisjoint(other_set)
# Returns True if the set has no common elements with the other_set, otherwise returns False

set1 = {1, 2, 3}
set2 = {4, 5, 6}
print(set1.isdisjoint(set2))  # Output: True

set3 = {3, 4, 5}
print(set1.isdisjoint(set3))  # Output: False
```

Set Methods

Code Examples

```
# issubset(other_set)
# Returns True if all elements of the set are present in the other_set, otherwise returns False

set1 = {1, 2, 3}
set2 = {1, 2, 3, 4, 5}
print(set1.issubset(set2))  # Output: True

set3 = {4, 5, 6}
print(set1.issubset(set3))  # Output: False
```

Set Methods

Code Examples

```
# issuperset(other_set)
# Returns True if all elements of the other_set are present in the set, otherwise returns False
set1 = {1, 2, 3, 4, 5}
set2 = {1, 2, 3}
print(set1.issuperset(set2))  # Output: True

set3 = {4, 5, 6}
print(set1.issuperset(set3))  # Output: False
```

Set Methods

Code Example

```
# pop() Removes and returns an arbitrary element from the set

fruits = {"apple", "banana", "cherry"}
removed_fruit = fruits.pop()
print(removed_fruit)  # Output: 'apple'
print(fruits)  # Output: {'banana', 'cherry'}
```

Set Methods

Code Example

```
# remove(element)
# Removes an element from the set. Raises a KeyError if the element is not found

fruits = {"apple", "banana", "cherry"}
fruits.remove("banana")
print(fruits)  # Output: {'apple', 'cherry'}
```

Set Methods

Code Example

```
# symmetric_difference(other_set)
Returns a new set containing elements that
are in either the set or the other_set, but
not both

set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
sym_diff_set =
set1.symmetric_difference(set2)
print(sym_diff_set)  # Output: {1, 2, 5, 6}
```

Set Methods

Code Examples

```
# symmetric_difference_update(other_set)
Updates the set, keeping only elements that
are in either the set or the other_set, but not
both

set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
set1.symmetric_difference_update(set2)
print(set1)  # Output: {1, 2, 5, 6}
```

Set Methods

Code Examples

```
# union(other_set)
Returns a new set containing all elements
that are in the set or the other_set

set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
print(union_set)  # Output: {1, 2, 3, 4, 5}
```

Set Methods

Code Examples

```
# update(other_set)
Updates the set, adding all elements from
the other_set to it.

set1 = {1, 2, 3}
set2 = {3, 4, 5}
set1.update(set2)
print(set1)  # Output: {1, 2, 3, 4, 5}
```

Tuple Syntax

Tuple is immutable

```
tuple_identifier = ()
```

parenthesis are used to initialize a tuple


```
tuple_identifier = ()
```


any valid identifier in Python

there can be duplicates

Parenthesis Overloading Problem

Code Example

```
a = (1)
print(type(a))

b = ( True )
print(type(b))

c = ( 'first' )
print(type(c))

d = ( 1, )
print(type(d))
```

Add a subheading

To avoid the parenthesis overloading problem and ensure consistent tuple creation, it is recommended to use a trailing comma when defining a tuple with a single element

Operations on Tuple

Code Examples

```
my_tuple = (1, 2, 3, 4, 5)

# Indexing
print(my_tuple[0]) # Output: 1

# Slicing
print(my_tuple[1:4]) # Output: (2, 3, 4)

# Iterating
for item in my_tuple:
    print(item)  # Output: 1, 2, 3, 4, 5
```

Tuple Methods

Code Examples

```
# count(element)
Returns the number of occurrences of a
specified element in the tuple

my_tuple = (1, 2, 2, 3, 4, 2)
count = my_tuple.count(2)
print(count)  # Output: 3
```

Tuple Methods

Code Examples

```
# index(element)
Returns the index of the first occurrence of a
specified element in the tuple

my_tuple = (1, 2, 3, 4, 5)
index = my_tuple.index(3)
print(index)  # Output: 2
```

blocks of reusable code that perform a specific task

Functions in Python

organizing code
improving readability
promoting reusability

Function Syntax

Keyword

```
def < Function_Name >(params):
    . . . .
```

Block of Code

. . .

. .

4 spaces

Any valid Py Code

Function Example

```
# function example
def add_numbers(a, b):
    return a + b
# calling a function
x = 12
y = 10
result = add_numbers(x, y)
```

Coding Activity

1. Write a function which takes a list as input and return its sum
2. Write a function which takes a list as an input and return the max element in the list
3. Write a function which takes a list as input and returns the reverse of the list in the output

No Built in Functions

4. Write a function which takes a list as input and returns the list in sorted format. (Note: You're not allowed to use in-built functions)

5. Write a function which takes a list as an input and returns the following statistics:

1. Mean / Average
2. Mode

Problem Solving

6.

Step-1 Read a sequence of numbers as an input from user.

Step-2 Read another integer k from the user

Step-3 Ask user to input two number start and end, k times

Step-4 Return the sum of the array from start -> end as a result in the k-sized list