Welcome to the

Python Camp

Day 03

Log into the virtual machine

Username: student

Password: StudentDLH2024

Content of the course

- **Variables and data types**
- **Logical and arithmetic operations**
- **Conditionals**
- **Loops**
- Lists
- Dictionaries

- Sorting

Recap

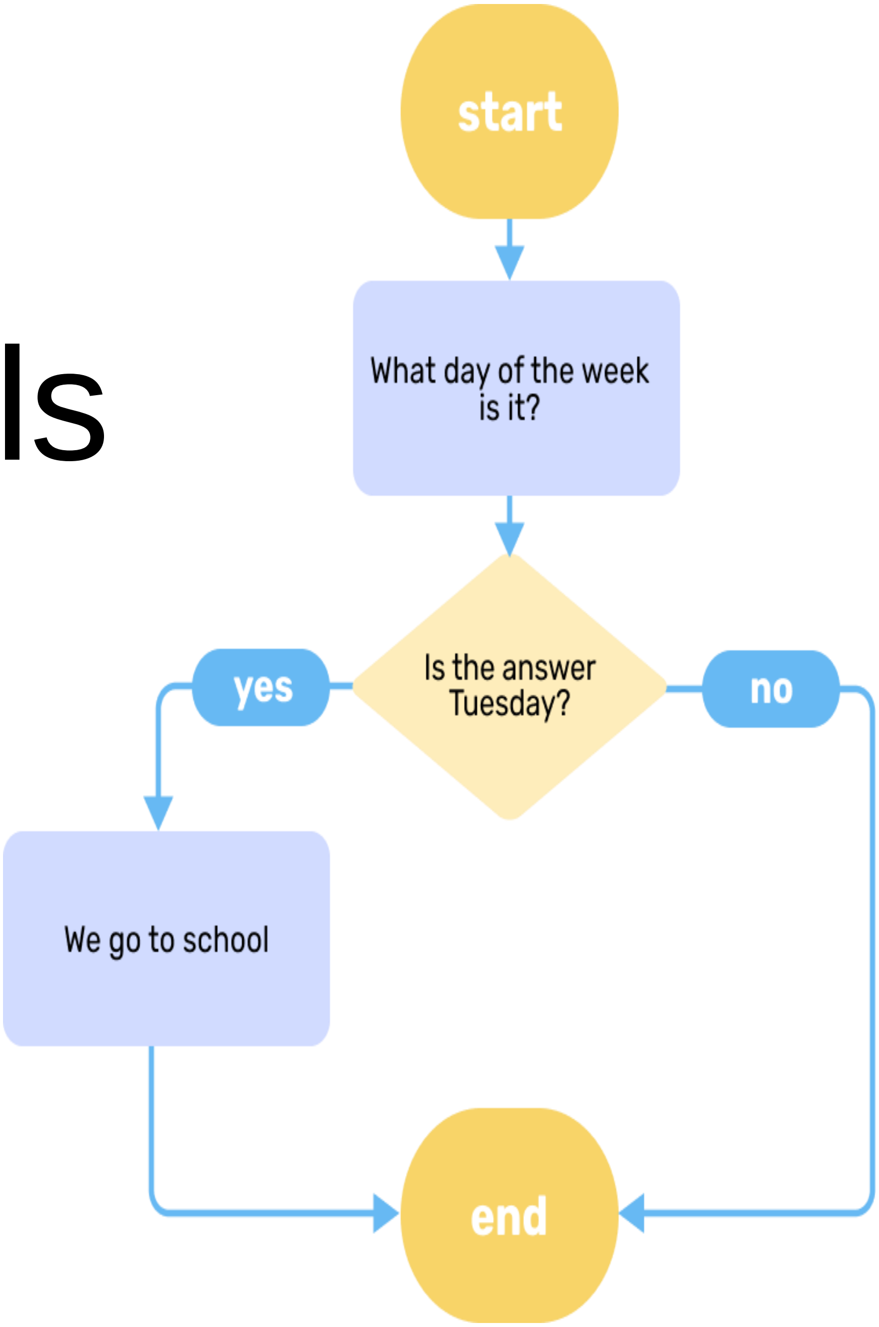
Python libraries

- Random

```
import random
radom_number = random.randint(1, 100)

print(radom_number)
```

Conditionals

```
a = int(input("Enter a number: "))
if a > 0:
    print("Positive number")
elif a < 0:
    print("Negative number")
else:
    print("Number equal to zero")
```



start

What day of the week is it?

Is the answer Tuesday?

yes

no

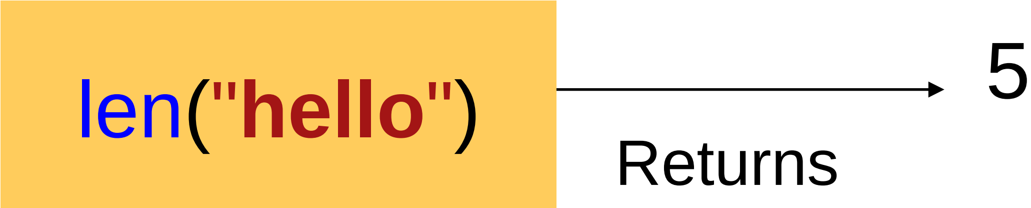
We go to school

end

Comparison operators

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equality
!=	Inequality

Function len()

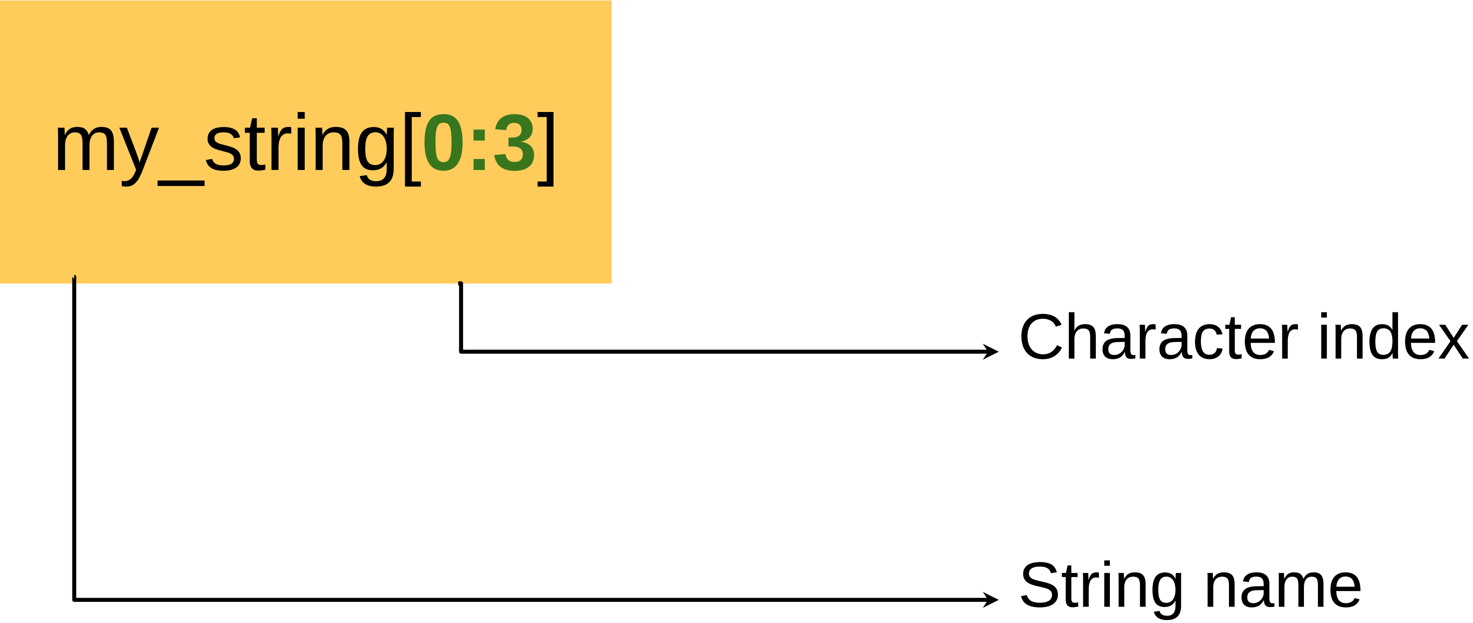


`len("hello")`

Returns

5

Extracting chars by index

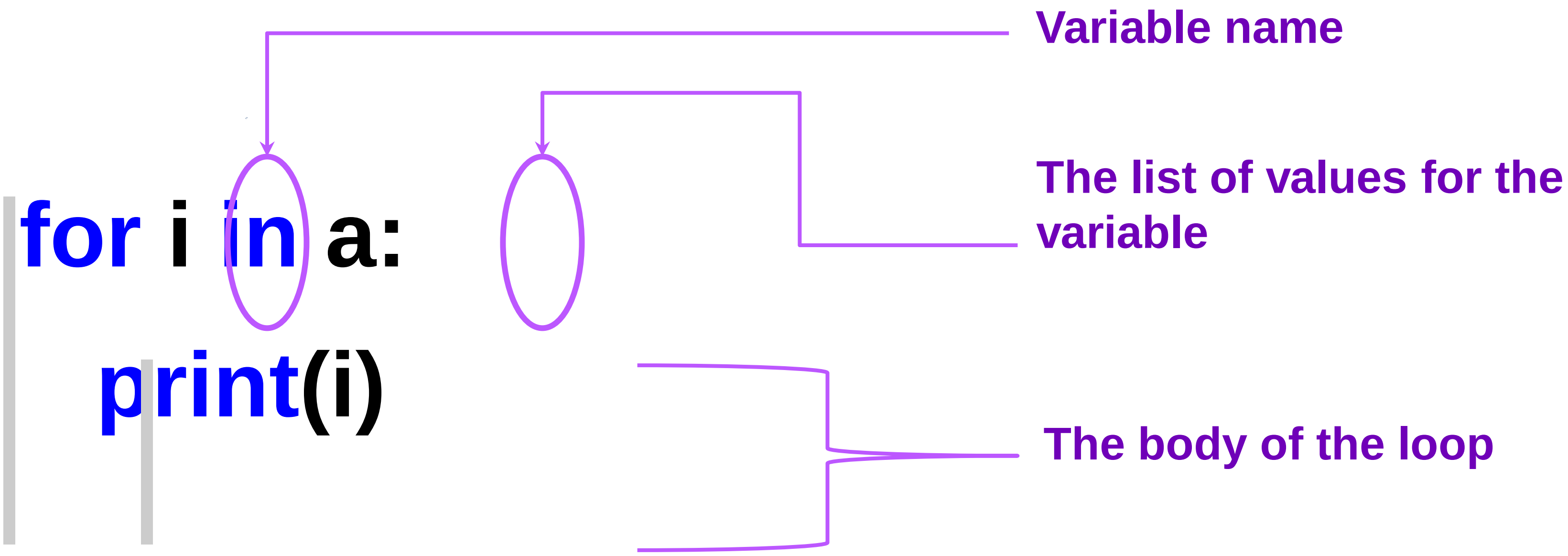


my_string[**0:3**]

Character index

String name

For Loops



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

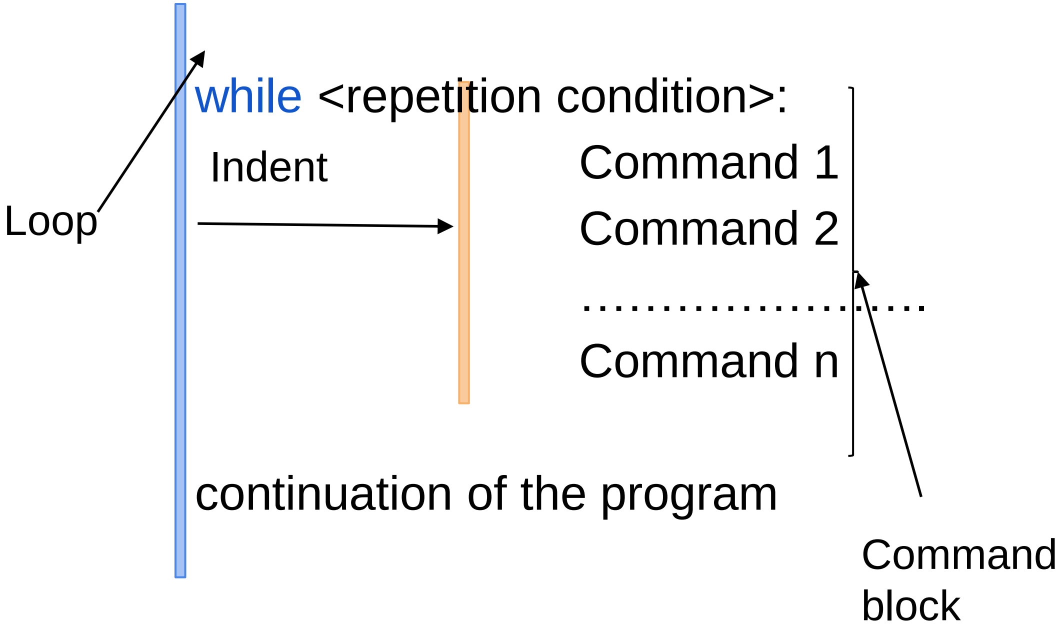
Variable name

The list of values for the variable

The body of the loop

for all i **in the** list a we make :
<repetitive actions>

While Loops

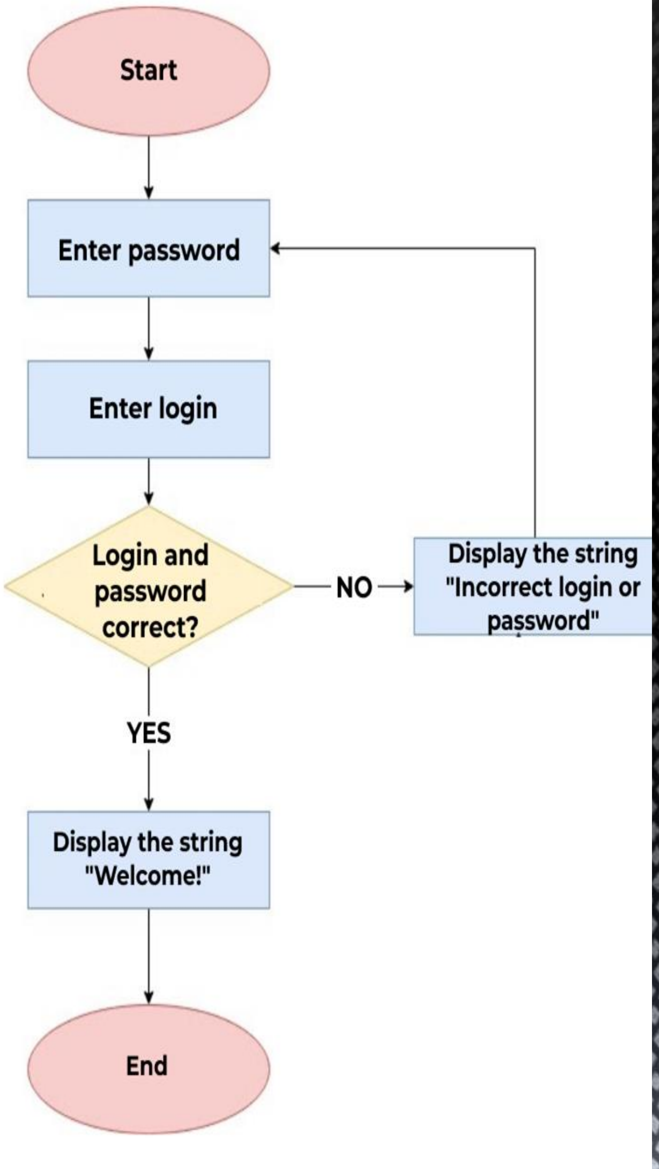


```
while <repetition condition>:
                Command 1
                Command 2
                ………………………….
                Command n

continuation of the program
```

Loop

Indent

Command block



Start
Enter password
Enter login
Login and password correct?
NO
Display the string "Incorrect login or password"
YES
Display the string "Welcome!"
End

List

Is an ordered sequence of items

```
checklist = ["one" , "two" , "three"]
```

```
checklist2 = [1 , 2 , 3]
print(checklist[2])
```

```
Output:
3
```

Items in the first list	Index (starting from 0)
one	0
two	1
three	2
Items in the second list	**Index (starting from 0)**
1	0
2	1
3	2

Python List

There are many built-in types in Python that allow us to group and store multiple items. Python lists are the most versatile among them.

For example, we can use a Python list to store a playlist of songs so that we can easily add, remove, and update songs as needed.

Create a Python List

We create a list by placing elements inside square brackets [], separated by commas.

For example,

```
 # a list of three elements
ages = [19, 26, 29]
print(ages)

# Output: [19, 26, 29]
```

List Characteristics

Lists are:

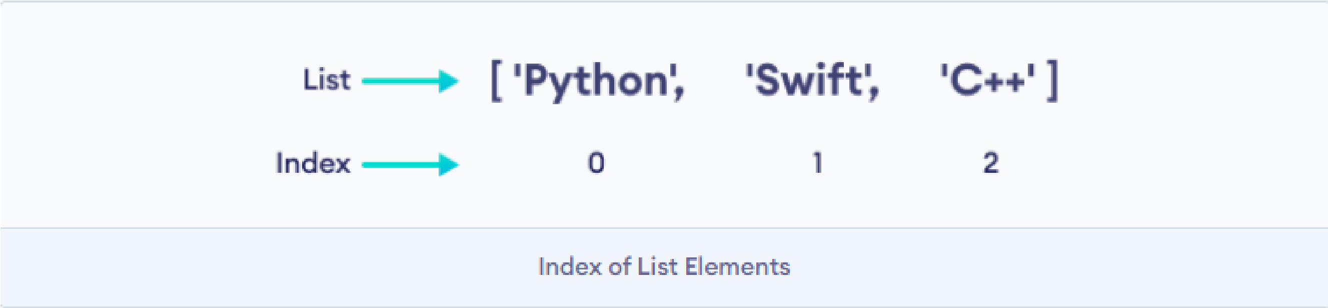
Ordered - They maintain the order of elements.

Mutable - Items can be changed after creation.

Allow duplicates - They can contain duplicate values.

Access List Elements

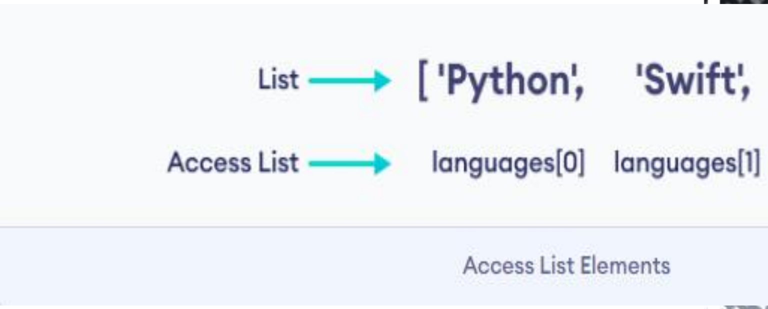
Each element in a list is associated with a number, known as an **index**.
The index of first item is **0**, the index of second item is **1**, and so on.



Index of List Elements

We use these index numbers to access list items. For example,

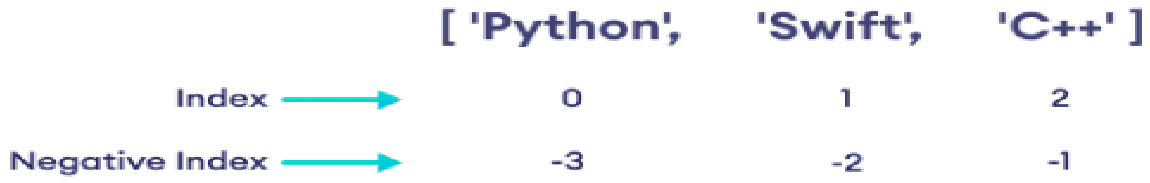
```
languages = ['Python', 'Swift', 'C++']

# Access the first element
print(languages[0])   # Python

# Access the third element
print(languages[2])   # C++
```



Access List Elements

Negative Indexing in Python

Python also supports negative indexing. The index of the last element is **-1**, the second-last element is **-2**, and so on.



	['Python',	'Swift',	'C++']
Index →	0	1	2
Negative Index →	-3	-2	-1

Python Negative Indexing

Negative indexing makes it easy to access list items from last.
Let's see an example,

```
languages = ['Python', 'Swift', 'C++']

# Access item at index 0
print(languages[-1])   # C++

# Access item at index 2
print(languages[-3])   # Python
```

Slicing of a list in Python

In Python, it is possible to access a section of items from the list using the slicing operator ’:’. For example,

```
my_list = ['p', 'r', 'o', 'g', 'r', 'a', 'm']

# items from index 2 to index 4
print(my_list[2:5])

# items from index 5 to end
print(my_list[5:])

# items beginning to end
print(my_list[:])

#items before a specific position
print(my_list[:2])
```

```
['o', 'g', 'r']
['a', 'm']
['p', 'r', 'o', 'g', 'r', 'a', 'm']
[‘p’,’r’,’o’]
```

Let’s do exercises on List

Iterating Through a List

We can use a for loop to iterate over the elements of a list. For example,

```
fruits = ['apple', 'banana', 'orange']

# iterate through the list
for fruit in fruits:
    print(fruit)
```

```
apple
banana
orange
```

Check an item exists in List

```
fruits = ['apple', 'cherry', 'banana']

print('orange' in fruits)    # False
print('cherry' in fruits)    # True
```

Python Tuple

A tuple is a collection similar to a Python list. The primary difference is that we cannot modify a tuple once it is created.

```
numbers = (1, 2, -5)
print(numbers)

# Output: (1, 2, -5)
```

Tuple Characteristics

Tuples are:
Ordered - They maintain the order of elements.
Immutable - They cannot be changed after creation.
Allow duplicates - They can contain duplicate values.

Tuple Cannot be Modified

```
cars = ('BMW', 'Tesla', 'Ford', 'Toyota')
# trying to modify a tuple
cars[0] = 'Nissan' # error
print(cars)
```

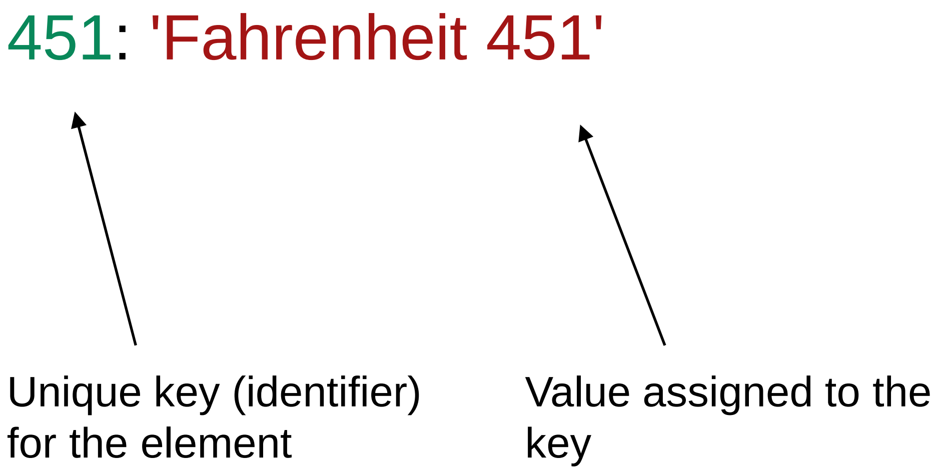
Python Tuple Length

```
cars = ('BMW', 'Tesla', 'Ford', 'Toyota')
print('Total Items:', len(cars))
# Output: Total Items: 4
```

Dictionaries

Dictionaries

A collection of key: value pairs.
A Key is a unique value in the dictionary



451: 'Fahrenheit 451'

Unique key (identifier) for the element

Value assigned to the key

Dictionaries

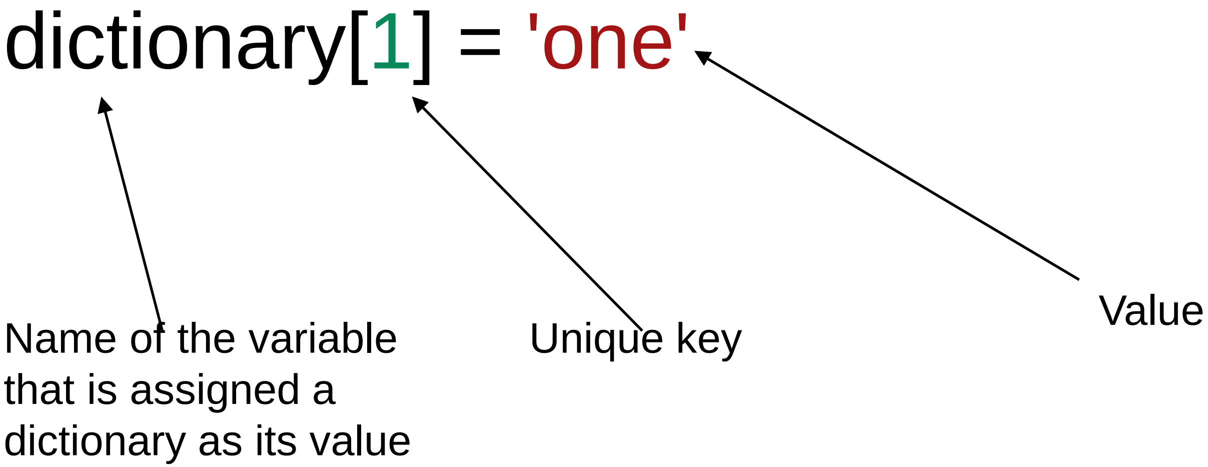
```
dictionary = {451: 'Fahrenheit 451’,
              20000: 'Twenty Thousand Leagues Under the Sea',
              1: 'One Flew Over the Cuckoo's Nest,
              1861: 'Great Expectations’,
              12: 'Alice in Wonderland’}

print(dictionary[451])
```

Output:

```
'Fahrenheit 451'
```

Dictionaries

Adding or changing a value



```
dictionary[1] = 'one'
```

Name of the variable
that is assigned a
dictionary as its value

Unique key

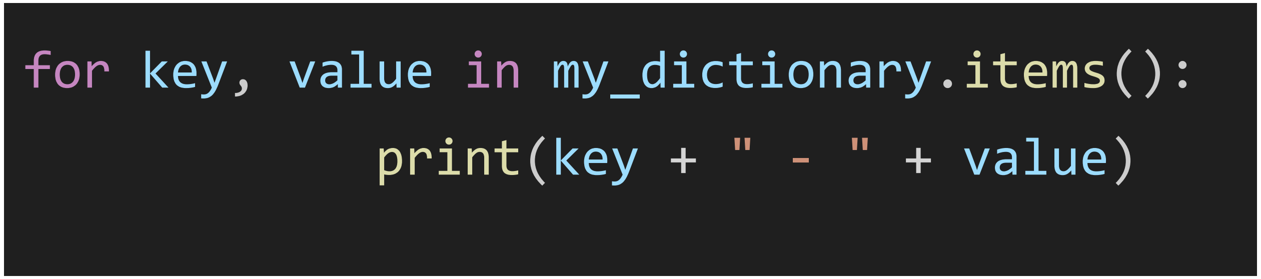
Value

Dictionaries

dict.keys()	Returns a view of all the keys
dict.values()	Returns a view of all the values
dict.items()	Returns a view of all the key-values pairs
dict.pop(key)	Removes the value associated with key
dict.popitem()	Removes and return arbitrary key-value pair

Dictionaries

How to loop through the items of a dictionary



```
for key, value in my_dictionary.items():
            print(key + " - " + value)
```

Now YOUR TURN !

Let’s do exercises number 1

Sets

Sets

Represents an unordered collection of unique items.

```
many = {1, 2, 3, 1, 2, 3}
print(many)
```

6 items are recorded, 3 of which are duplicates

Output:

```
{1, 2, 3}
```

3 items are displayed, no duplicates

Set operation

A.union(B)	Returns a set that is the union of the sets **A** and **B**.
A.update(B)	Adds to set **A** all the elements from set **B**.
A.intersection(B)	Returns a set that is an intersection of sets **A** and **B**.
A.intersection_update(B)	Leaves in set **A** only those elements that are in set **B**.
A.difference(B)	Returns the element in **A** but not in **B**.
A.difference_update(B)	Deletes from **A** all the elements in **B**.

Set operation

A.symmetric_difference(B)	Returns all the elements in **A** or in **B**, but not in both of them at the same time.
A.issubset(B)	Returns true if **A** is a subset of **B**.
A.issuperset(B)	Returns true if **B** is a subset of **A.**
A < B	The equivalent of **A <= B** and **A != B**
A > B	The equivalent of **A >= B** and **A != B**

Now YOUR TURN !

Let’s do exercises number 2