PYTHON DAY - 2

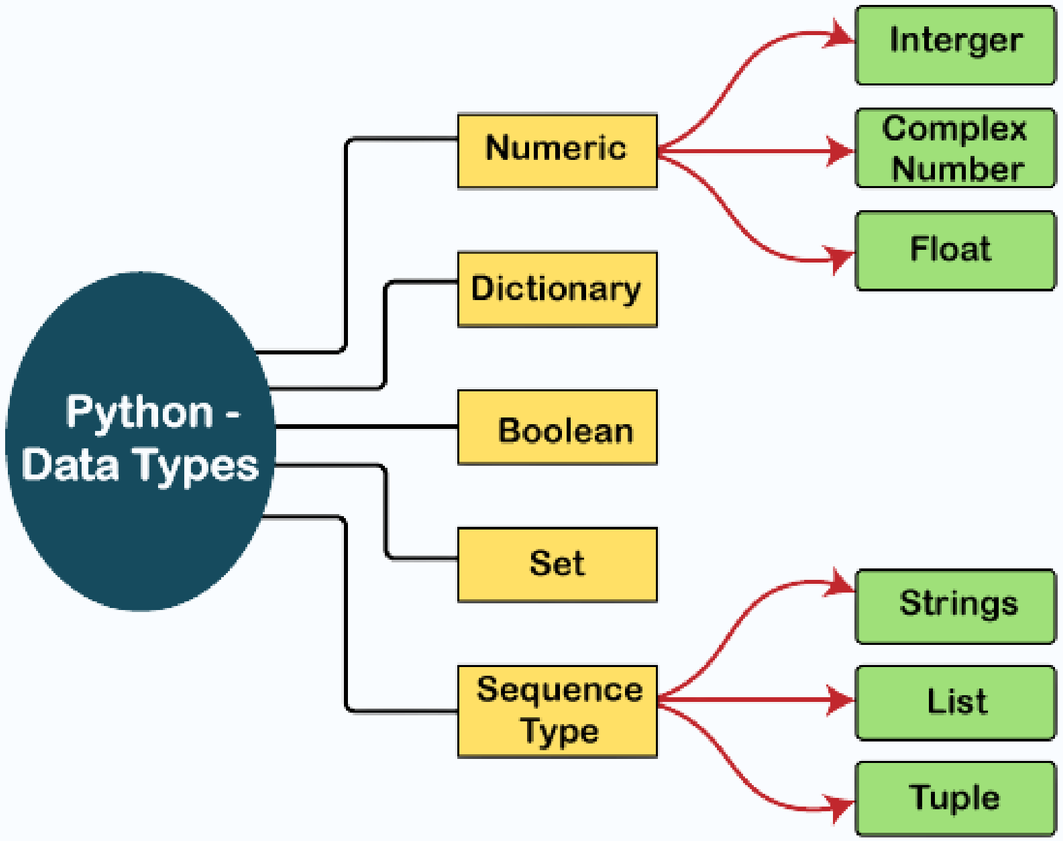
Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

DATA TYPES

• DATA TYPES

1. Numeric
2. Sequence
3. Set
4. Dictionary
5. Boolean

DATA TYPES



- Data types refer to the type or classification of data that a variable or value can hold
- Data types are automatically determined by the interpreter based on the value assigned to a variable

NUMBERS

10

11.2

2 + 5j

```
10        --------> integer

11.2      --------> Float

2 + 5j    --------> Complex number
```

NUMBER TYPE CONVERSION

int(5.32) → 5

float(11) → 11.0

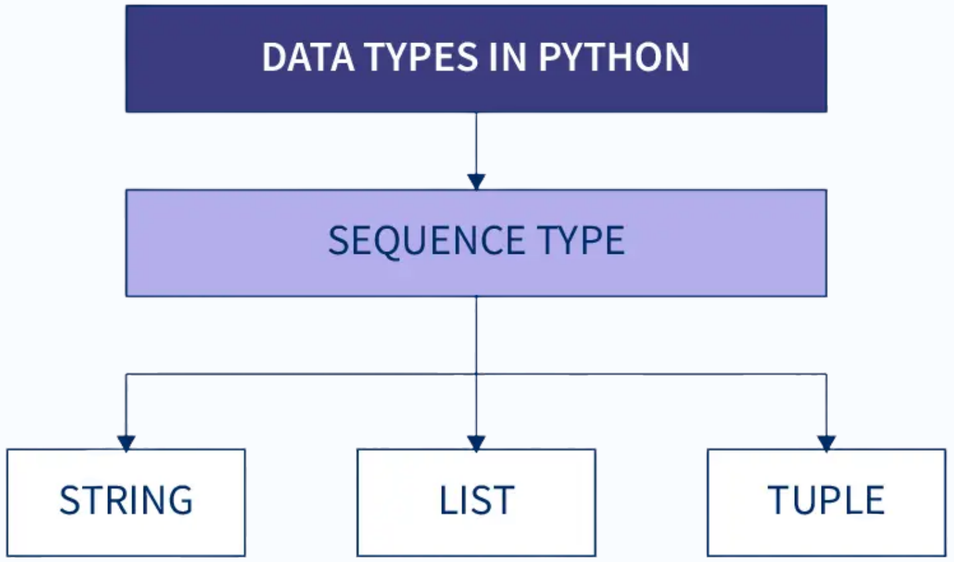
complex(10) → 10 + 0j

```
int(5.32)


float(11)


complex(10)
```

2. SEQUENCE

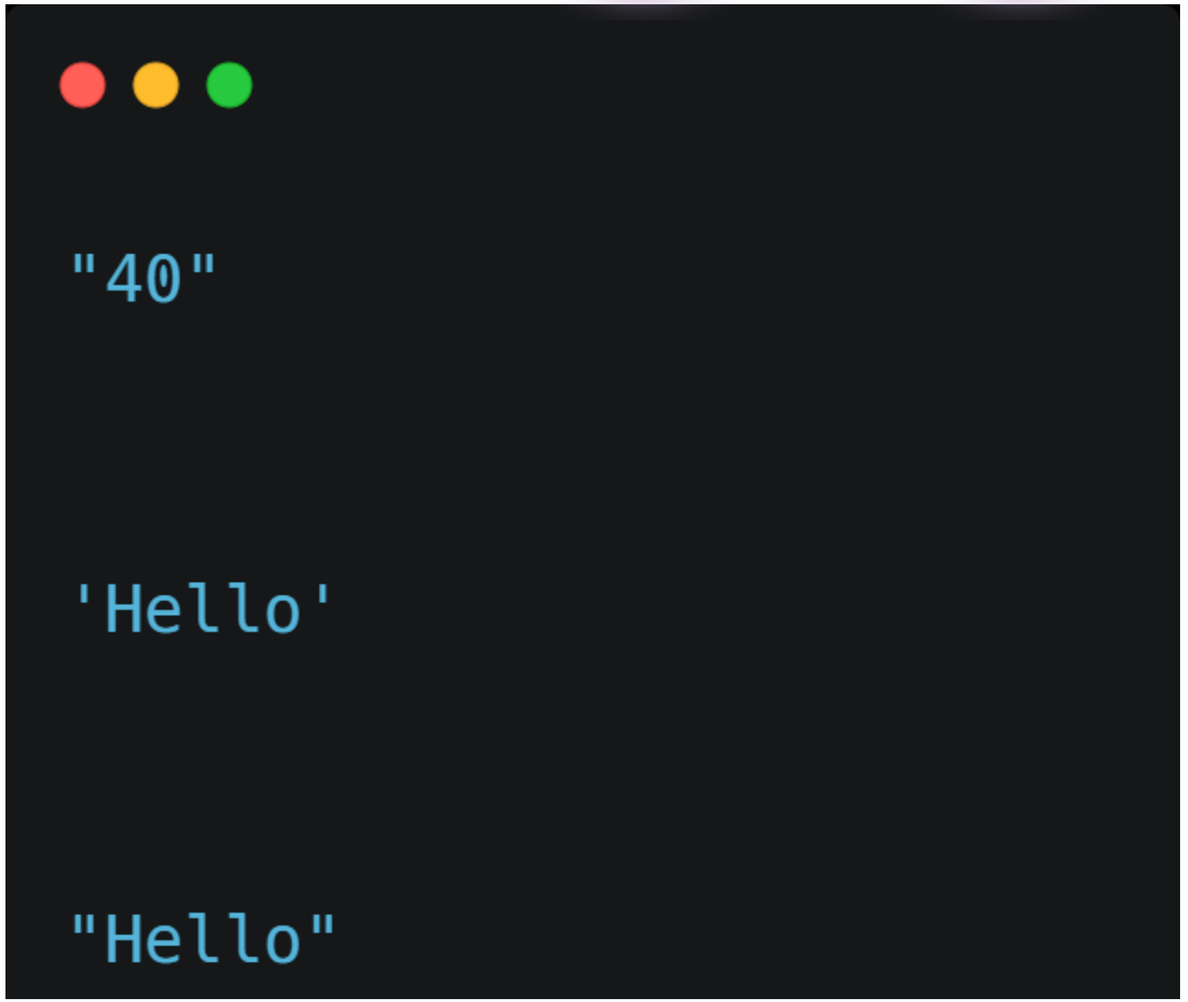


STRING

"40"

'Hello'

"Hello"



LENGTH OF A STRING

```
length

len(string)
```

Length of a string

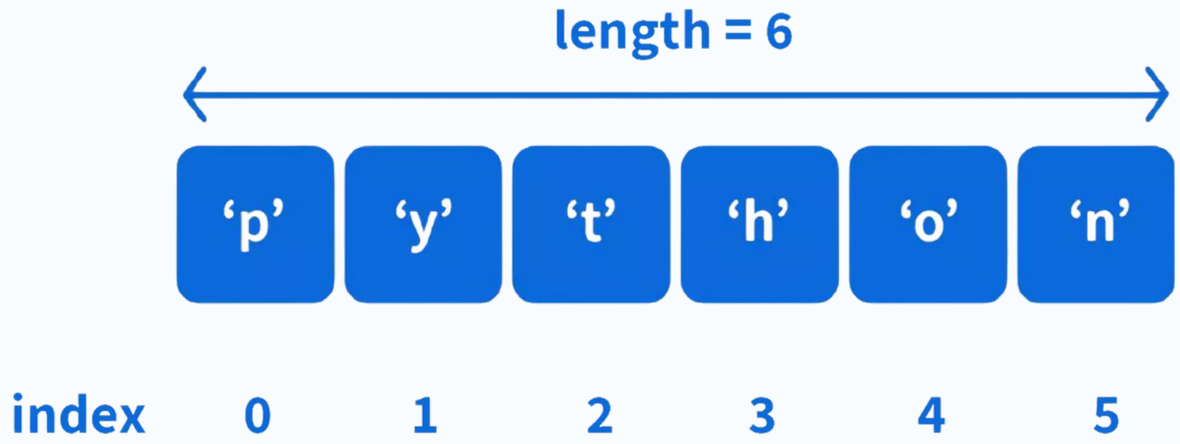
String ⇨

F	A	C	E

Length = 4

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

INDEXING

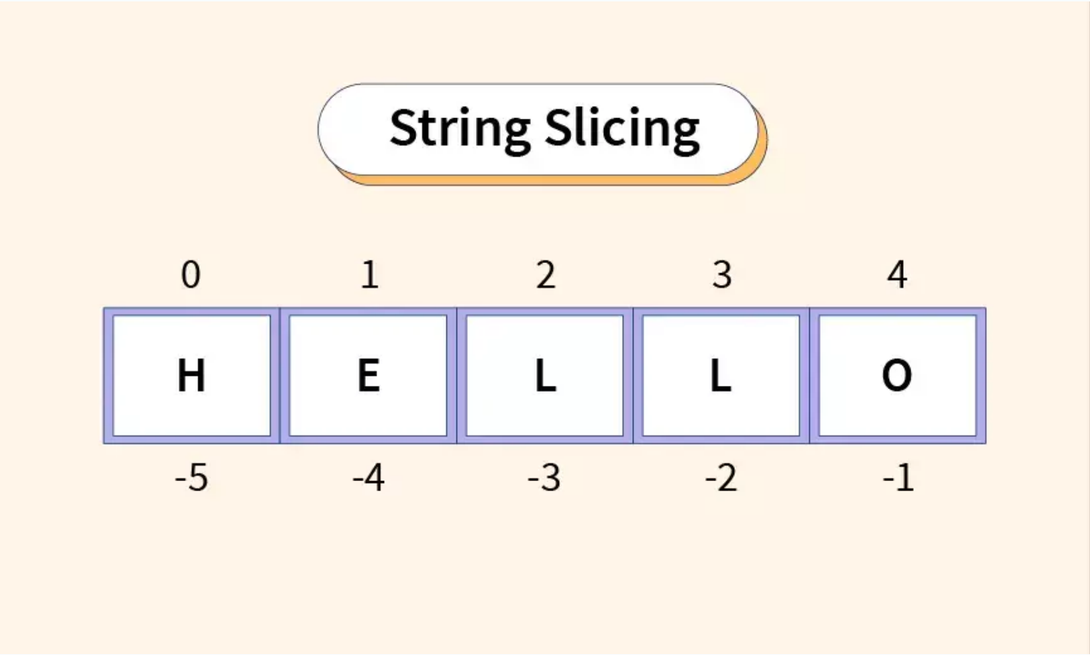


index

```
x = "python"

print(x[0])
```

STRING SLICING



Slicing

```
x = "Hello"


print(x[1:4])   ----> output: ell

print(x[-4:-1]) ----> output: ell
```

STRING CONCATENATION

"Hello" + "world"

"Hello world"

Concatenation

```
x = "Hello"

y = "world"



print(x + y)
```

STRING FORMAT

```
Format

age = 20

print(f"His age is {age}")
```

- we cannot directly combine strings and numbers
- But we can combine strings and numbers by using the format() method!
- placeholders should be created with {}

STRING METHODS

Method
.capitalize()
.upper()
.lower()
.replace
.split()
.find()

CAPITALIZE

capitalize

```
x = "hello world"


print(x.capitalize())
```

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

UPPER & LOWER

Upper & Lower

```
x = "hello WORLD"


print(x.upper())

print(x.lower())
```

REPLACE

Replace

```
x = "hello world"


y = x.replace("l", "k")


print(y)
```

SPLIT

Split

```
x = "hello world"


y = x.split(" ")


print(y)
```

FIND

Find

```
x = "hello world"


y = x.find("w")


print(y)
```

LISTS

```
x = ["apple", "orange", "banana"]
```

- A list is a collection of items enclosed in square brackets []

- Lists are ordered, and can be of any type (e.g. integers, strings, etc.)

- Lists are mutable, items can be modified after they are created.

ACCESSING ITEMS IN LIST

```
x = ["apple", "orange", "banana"]


print(x[0])
```

- Access the items present in a list with the help of index number

ACCESS A RANGE OF INDEX

```
x = ["apple", "orange", "banana", "kiwi", "strawberry"]


print(x[1 : 5])
```

CHANGE ITEMS PRESENT IN LIST

```
x = ["apple", "orange", "banana"]


x[2] = "pineapple"


print(x)
```

INSERT

```
x = ["apple", "orange", "banana"]


x.insert(1, "pineapple")


print(x)
```

APPEND

```
x = ["apple", "orange", "banana"]


x.append("pineapple")


print(x)
```

- This will add the specified item in the end of the list

EXTEND

```
x = ["apple", "orange", "banana"]

y = ["kiwi", "pineapple", "strawberry"]


x.extend(y)


print(x)
```

- This will append elements from another list

REMOVE

```
x = ["apple", "orange", "banana"]


x.remove("apple")


print(x)
```

- Remove specified string from the list

POP

```
x = ["apple", "orange", "banana"]


x.pop(2)


print(x)
```

- Remove item present in the specified index

CLEAR

```
x = ["apple", "orange", "banana"]


x.clear()


print(x)
```

- Clears the entire list

SORTING LIST

```
x = ["orange", "watermelon", "banana", "apple"]


x.sort()


print(x)
```

- This will sort the list in alphabetical order, numbers will be sorted in increasing order

SORT DESCENDING

```
x = ["orange", "watermelon", "banana", "apple"]


x.sort(reverse = True)


print(x)
```

- This will sort the list in descending order

COPY LIST

```
x = ["orange", "banana", "apple"]


y = x.copy()


print(y)
```

- This will create a new list by copying

TUPLE

```
x = ("apple", "orange", "banana")
```

- Tuple is a collection of items enclosed in parentheses ()

- Like lists, tuples are ordered and can be of any type.

- However, tuples are immutable, which means that their items cannot be modified after they are created.

ACCESSING ITEMS IN TUPLE

```
x = ("apple", "orange", "banana")


print(x[0])
```

- Access the items present in a tuple with the help of index number

ACCESS A RANGE OF INDEX

```
x = ("apple", "orange", "banana", "kiwi", "strawberry")


print(x[1 : 4])
```

TUPLE IS IMMUTABLE

```
x = ("apple", "orange", "banana")


y = list(x)


y[0] = "kiwi"


x = tuple(y)


print(y)
```

- Changing elements in tuple is not possible, instead,
- Convert tuple into list and then change the item
- Then again convert the changed list back into tuple

REMOVE

```python
x = ("apple", "orange", "banana")


y = list(x)


y.remove("banana")


x = tuple(y)


print(y)
```

- To remove specified string from the tuple, convert tuple into a list, then remove and convert back into tuple

COMBINE TWO TUPLE

```
x = ("a", "b", "c")

y = (1, 2, 3)


z = x + y

print(z)
```

- Combine two tuple with the help of "+"

SET

```
x = {"apple", "orange", "banana"}
```

- Set is a collection of unique items enclosed in curly braces {}
- The items in a set have no defined order, and can be of any type
- sets are mutable
- Duplicate is not allowed

DUPLICATES WILL BE IGNORED

```
x = {"apple", "orange", "banana", "orange"}


print(x)
```

ADD

```
x = {"apple", "orange", "banana"}


x.add("kiwi")


print(x)
```

- Use add method to add new items in the set

UPDATE

```
x = {"apple", "orange", "banana"}

y = {"kiwi", "mango"}

x.update(y)


print(x)
```

- Use update method to combine two set

REMOVE

```
x = {"apple", "orange", "banana"}


x.remove("orange")


print(x)
```

- Use remove method to remove specified item from set

DIFFERENCE

```
x = {1, 2, 3, 4, 5, 6}


y = {5, 6, 7, 8, 9, 10}


print(x.difference(y))
```

- To find the values exist in x alone

INTERSECTION

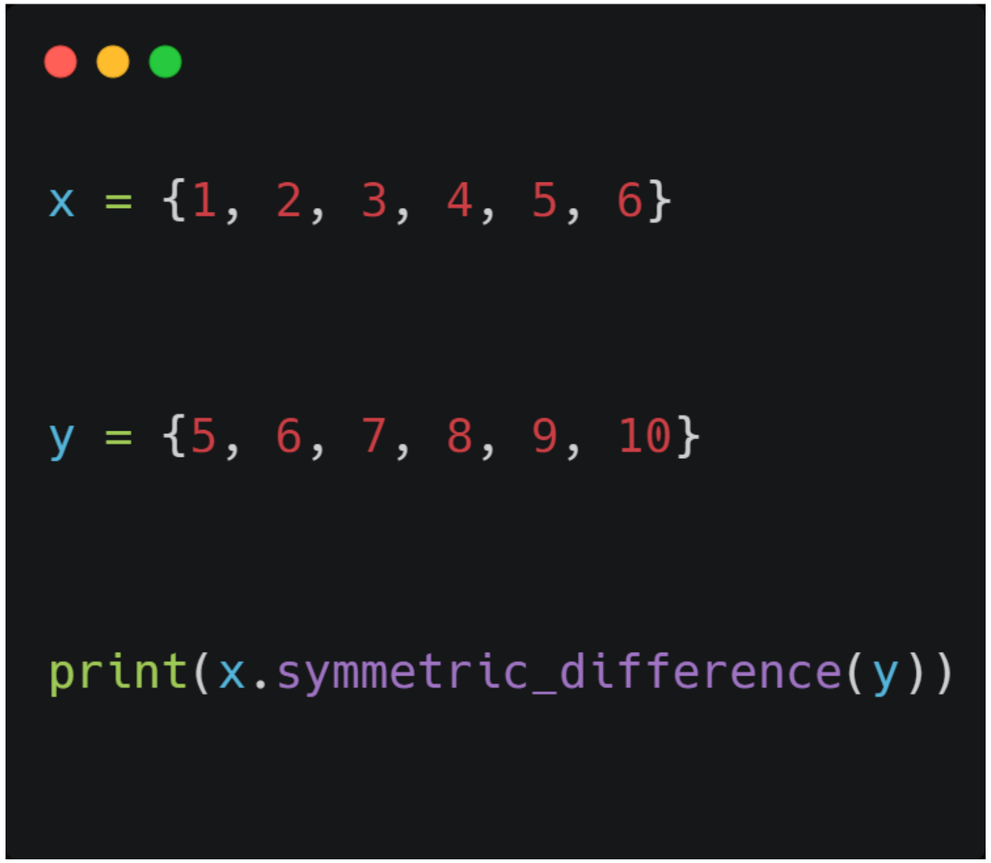
```
x = {1, 2, 3, 4, 5, 6}


y = {5, 6, 7, 8, 9, 10}


print(x.intersection(y))
```

- To find common values exist in both

SYMMETRIC DIFFERENCE



```
x = {1, 2, 3, 4, 5, 6}


y = {5, 6, 7, 8, 9, 10}


print(x.symmetric_difference(y))
```

- All the items which are not common will be in the output set

DICTIONARY

```
dict = {"name" : "john","age" : 30, "country" : "India"}
```

- Dictionary is a collection of key-value pairs enclosed in curly braces {}

- Dictionaries are also mutable, key-value pairs can be added, removed, or modified after they are created.

- Duplicates are not allowed

BOOLEAN

- Boolean is a data type that can have one of two values: True or False

- often used in conditional statements, comparision statements

THANKS FOR WATCHING

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED