SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Artificial Intelligence and Data Science

Course Name – Computational Thinking and Python Programming

I Year / I Semester

Unit 2-DATA, EXPRESSIONS, STATEMENTS

Value:

Value can be any letter ,number or string.

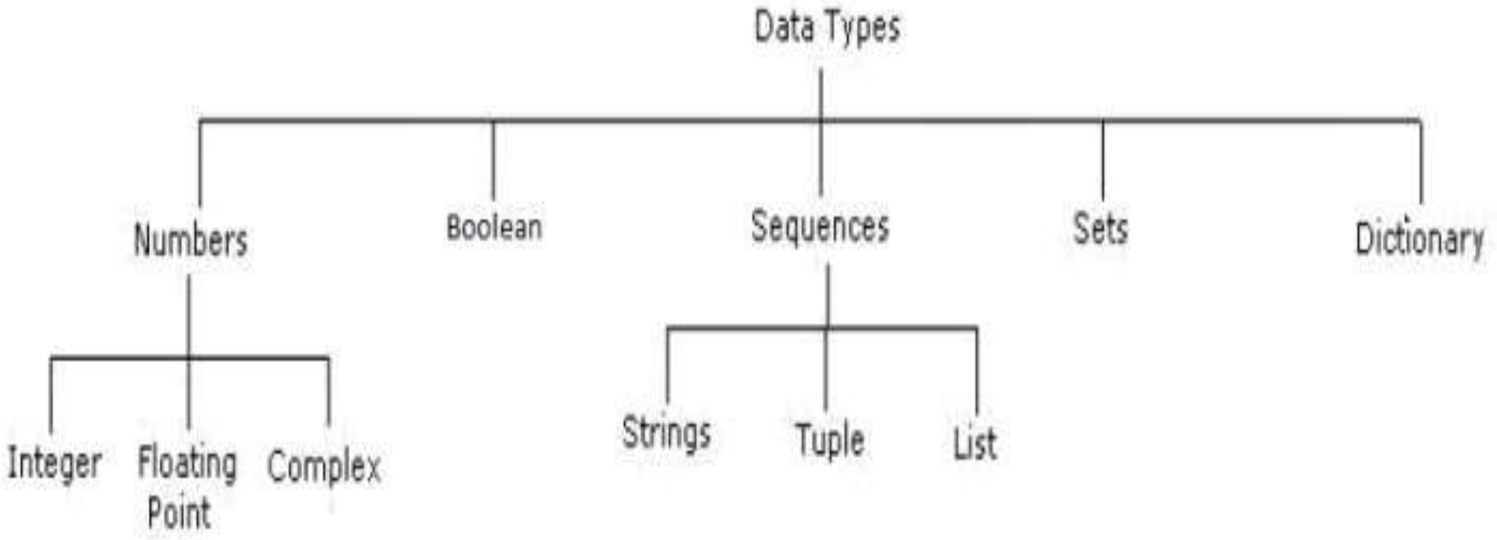
Eg, Values are 2, 42.0, and 'Hello, World!'. (These values belong to different datatypes.)

Data type:

Every value in Python has a data type.

It is a set of values, and the allowable operations on those values.

Python has four standard data types:



Numbers:

Number data type stores **Numerical Values.**
This data type is immutable [i.e. values/items cannot be changed].
Python supports integers, floating point numbers and complex numbers.
They are defined as,

Integers	Long	Float	Complex
- They are often called just integers or **int**. - They are positive or negative whole numbers with no decimal point.	-They are long integers. -They can also be represented in octal and hexadecimal representation.	-They are written with a decimal point dividing the integer and the fractional parts.	-They are of the form **a + bj**, where a and b are floats and j represents the square root of -1 (which is an imaginary number). -The real part of the number is a, and the imaginary part is b.
Eg, 56	**Eg, 5692431L**	**Eg, 56.778**	**Eg, square root of -1 is a complex number**

Sequence:

•A sequence is an **ordered collection of items**, indexed by positive integers.
•It is a combination **of mutable** (value can be changed) **and immutable** (values cannot be changed) data types.
There are three types of sequence data type available in Python, they are

1. **Strings**
2. **Lists**
3. **Tuples**

1. Strings

A String in Python consists of a series or sequence of characters - letters, numbers, and special characters.
Strings are marked by quotes:
· single quotes (' ') Eg, 'This a string in single quotes'
· double quotes (" ") Eg, "'This a string in double quotes'"
· triple quotes(""" """) Eg, This is a paragraph. It is made up of multiple lines and sentences."""“
Individual character in a string is accessed using a subscript (index).
Characters can be accessed using indexing and slicing operations
Strings are immutable i.e. the contents of the string cannot be changed after it is created.

Indexing:

String A	H	E	L	L	O
Positive Index	0	1	2	3	4
Negative Index	-5	-4	-3	-2	-1

- Positive indexing helps in accessing the string from the beginning
- Negative subscript helps in accessing the string from the end.
- Subscript 0 or –ve n(where n is length of the string) displays the first element.

Example: A[0] or A[-5] will display “H”

- Subscript 1 or –ve (n-1) displays the second element.

Example: A[1] or A[-4] will display “E”

Operations on string:

i. Indexing
ii. Slicing
iii. Concatenation
iv. Repetitions
v. Member ship

Creating a string	>>> s="good morning"	Creating the list with elements of different data types.
Indexing	**>>> print(s[2])** o **>>> print(s[6])** O	❖ Accessing the item in the position 0 ❖ Accessing the item in the position 2
Slicing(ending position -1)	**>>> print(s[2:])** od morning	- Displaying items from 2nd till last.
Slice operator is used to extract part of a data type	**>>> print(s[:4])** Good	- Displaying items from 1st position till 3rd .
Concatenation	**>>>print(s+"friends")** good morningfriends	-Adding and printing the characters of two strings.
Repetition	**>>>print(s*2)** good morninggood morning	Creates new strings, concatenating multiple copies of the same string
in, not in (membership operator)	>>> s="good morning" **>>>"m" in s** **True** **>>> "a" not in s** **True**	Using membership operators to check a particular character is in string or not. Returns true if present.

2. Lists

-

- List is an ordered sequence of items. Values in the list are called elements / items.
- It can be written as a list of comma-separated items (values) between **square brackets[ ]**.
- Items in the lists can be of different data types.

Operations on list:

- Indexing
- Slicing
- Concatenation
- Repetitions
- Updation, Insertion, Deletion

Creating a list	**>>>list1=["python", 7.79, 101, "hello"]** **>>>list2=["god",6.78,9]**	Creating the list with elements of different data types.
Indexing	**>>>print(list1[0])** python **>>> list1[2]** 101	❖ Accessing the item in the position 0 ❖ Accessing the item in the position 2
Slicing(ending position -1) Slice operator is used to extract part of a string, or some part of a list Python	**>>> print(list1[1:3])** [7.79, 101] **>>>print(list1[1:])** [7.79, 101, 'hello']	- Displaying items from 1st till 2nd. - Displaying items from 1st position till last.
Concatenation	**>>>print(list1+list2)** ['python', 7.79, 101, 'hello', 'god', 6.78, 9]	-Adding and printing the items of two lists.
Repetition	**>>> list2*3** **['god', 6.78, 9, 'god', 6.78, 9, 'god', 6.78, 9]**	Creates new strings, concatenating multiple copies of the same string
Updating the list	**>>> list1[2]=45** >>>print(list1) ['python', 7.79, 45, 'hello']	Updating the list using index value
Inserting an element	**>>> list1.insert(2,"program")** >>> print(list1) ['python', 7.79, 'program', 45, 'hello']	Inserting an element in 2nd position
Removing an element	**>>> list1.remove(45)** >>> print(list1) ['python', 7.79, 'program', 'hello']	Removing an element by giving the element directly

3. Tuple:

- A tuple is same as list, except that the set of elements is **enclosed in parentheses** instead of square brackets.
- **A tuple is an immutable list.** i.e. once a tuple has been created, you can't add elements to a tuple or remove elements from the tuple.
- Benefit of Tuple:
- Tuples are faster than lists.
- If the user wants to protect the data from accidental changes, tuple can be used.
- Tuples can be used as keys in dictionaries, while lists can't.

Basic Operations:

Creating a tuple	**>>>t=("python", 7.79, 101, "hello")**	Creating the tuple with elements of different data types.
Indexing	**>>>print(t[0])** python **>>> t[2]** 101	❖ Accessing the item in the position 0 ❖ Accessing the item in the position 2
Slicing(ending position -1)	**>>>print(t[1:3])** (7.79, 101)	❖ Displaying items from 1st till 2nd.
Concatenation	**>>> t+("ram", 67)** ('python', 7.79, 101, 'hello', 'ram', 67)	❖ Adding tuple elements at the end of another tuple elements
Repetition	**>>>print(t*2)** ('python', 7.79, 101, 'hello', 'python', 7.79, 101, 'hello')	❖ Creates new strings, concatenating multiple copies of the same string

Mapping

-This data type is unordered and mutable.
-Dictionaries fall under Mappings.

Dictionaries:

•Lists are ordered sets of objects, whereas **dictionaries are unordered sets**.
•Dictionary is created by using **curly brackets**. i,e. {}
•Dictionaries **are accessed via keys** and not via their position.
•A dictionary is an associative array (also known as hashes). Any key of the dictionary is associated (or mapped) to a value.
•The values of a dictionary can be any Python data type. So dictionaries are **unordered key-value-pairs(**The association of a key and a value is called a key-value pair)
•Dictionaries don't support the sequence operation of the sequence data types like strings, tuples and lists.

Creating a dictionary	>>> food = {"ham":"yes", "egg" : "yes", "rate":450 } >>>print(food) {'rate': 450, 'egg': 'yes', 'ham': 'yes'}	Creating the dictionary with elements of different data types.
Indexing	>>>> print(food["rate"]) 450	Accessing the item with keys.
Slicing(ending position -1)	>>>print(t[1:3]) (7.79, 101)	Displaying items from 1st till 2nd.

Data type	Compile time	Run time
int	a=10	a=int(input("enter a"))
float	a=10.5	a=float(input("enter a"))
string	a="panimalar"	a=input("enter a string")
list	a=[20,30,40,50]	a=list(input("enter a list"))
tuple	a=(20,30,40,50)	a=tuple(input("enter a tuple"))