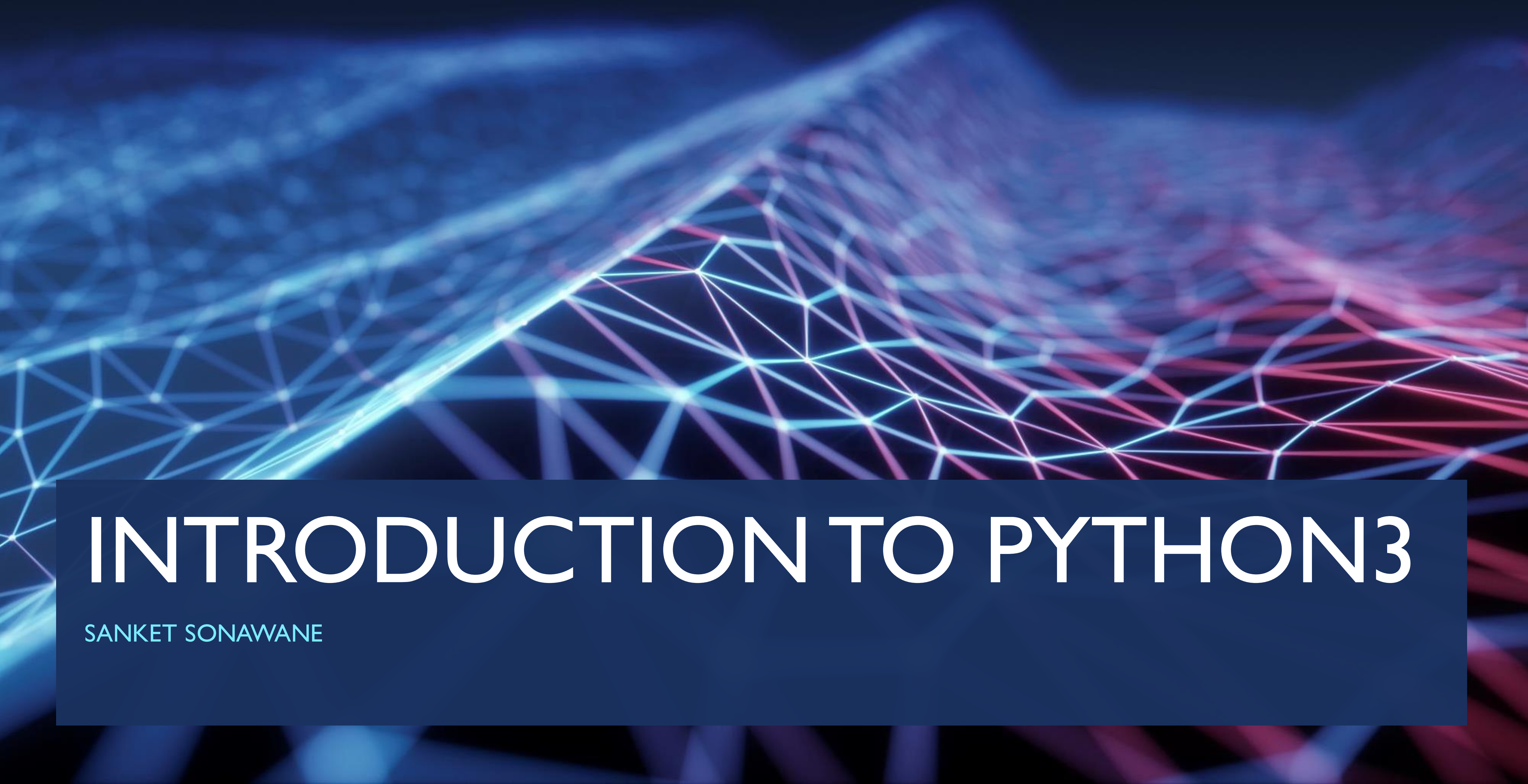


INTRODUCTION TO PYTHON3

SANKET SONAWANE

KEY FEATURES OF PYTHON

- Emphasis on **code readability, shorter codes, ease of writing**.
- Programmers can express logical concepts in **fewer lines of code** in comparison to languages such as C++ or Java.
- Python **supports multiple programming paradigms**, like object-oriented, imperative and functional programming or procedural.
- It provides **extensive support libraries**(Django for web development, Pandas for data analytics etc)
- **Dynamically typed language**(Data type is based on value assigned)
- Philosophy is "Simplicity is the best".

INDENTATION

- Python uses indentation to highlight the blocks of code. Whitespace is used for indentation in Python. All statements with the same distance to the right belong to the same block of code. If a block has to be more deeply nested, it is simply indented further to the right. You can understand it better by looking at the following lines of code.

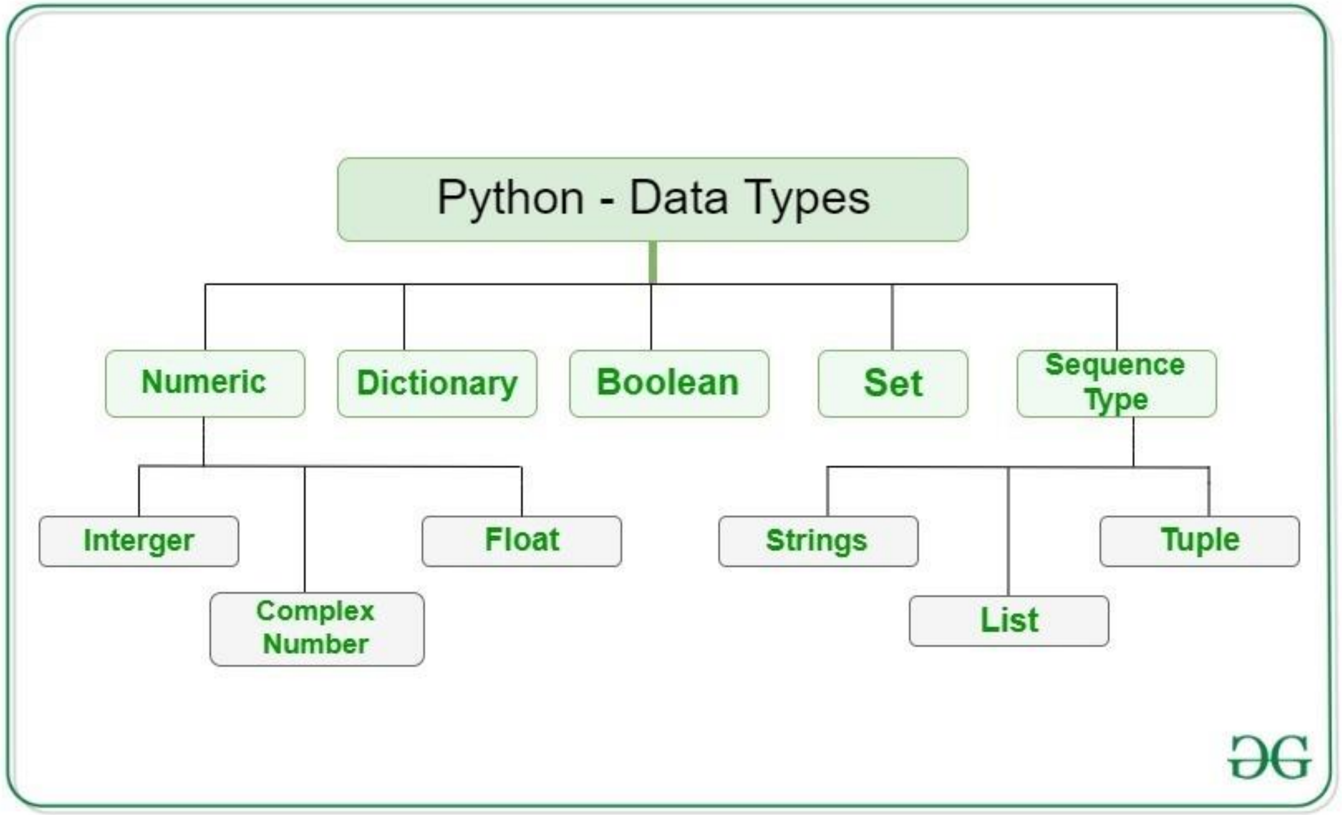
COMMENTS

- Python comments start with #. They are single line comments.
- Multil-line comments are not a feature of python but tripple quoted string (multi-line string) can be used as one.

VARIABLES

- Unlike most languages the data type of a Python variable is not specified by the user. They are dynamically typed according to the value assigned to them.
- Unlike many languages Python variables do not need declaration. They are declared when they are first assigned a value.

DATA TYPES



Extra type - None

OPERATORS

ARITHMATIC OPERATORS

OPERATOR	DESCRIPTION	SYNTAX
+	Addition: adds two operands Also contacatinates 2 iterables of same type	x + y
-	Subtraction: subtracts twooperands	x - y
*	Multiplication: multiplies twooperands	x * y
/	Division (float): divides the firstoperand by the second	x / y
//	Division (floor): divides the firstoperand by the second	x // y
%	Modulus: returns the remainderwhen first operand is divided by the second	x % y
**	Exponent: returns result when first operand is raised to thepower of the second operand	x ** y

RELATIONAL OPERATORS

OPERATOR	DESCRIPTION	SYNTAX
>	Greater than: True if left operand is greater than the right	x > y
<	Less than: True if left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

LOGICAL OPERATORS

OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x

BITWISE OPERATORS

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

ASSIGNMENT OPERATORS

OPERATOR	DESCRIPTION	SYNTAX
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add AND: Add right side operand with left side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign result to left operand	a%=b a=a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b a=a&b
|=	Performs Bitwise OR on operands and assign value to left operand	a|=b a=a|b
^=	Performs Bitwise xOR on operands and assign value to left operand	a^=b a=a^b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a>>=b a=a>>b
<<=	Performs Bitwise left shift on operands and assign value to left operand	a <<= b a= a << b

SPECIAL OPERATORS

Identity Operator (is, is not)

- **is** True if the operands are identical
- **is not** True if the operands are not identical
- **is** works like the === operator in JavaScript and **is not** like !== operator

Membership Operator (in, not in)

- **in** True if value is found in the sequence
- **not in** True if value is not found in the sequence

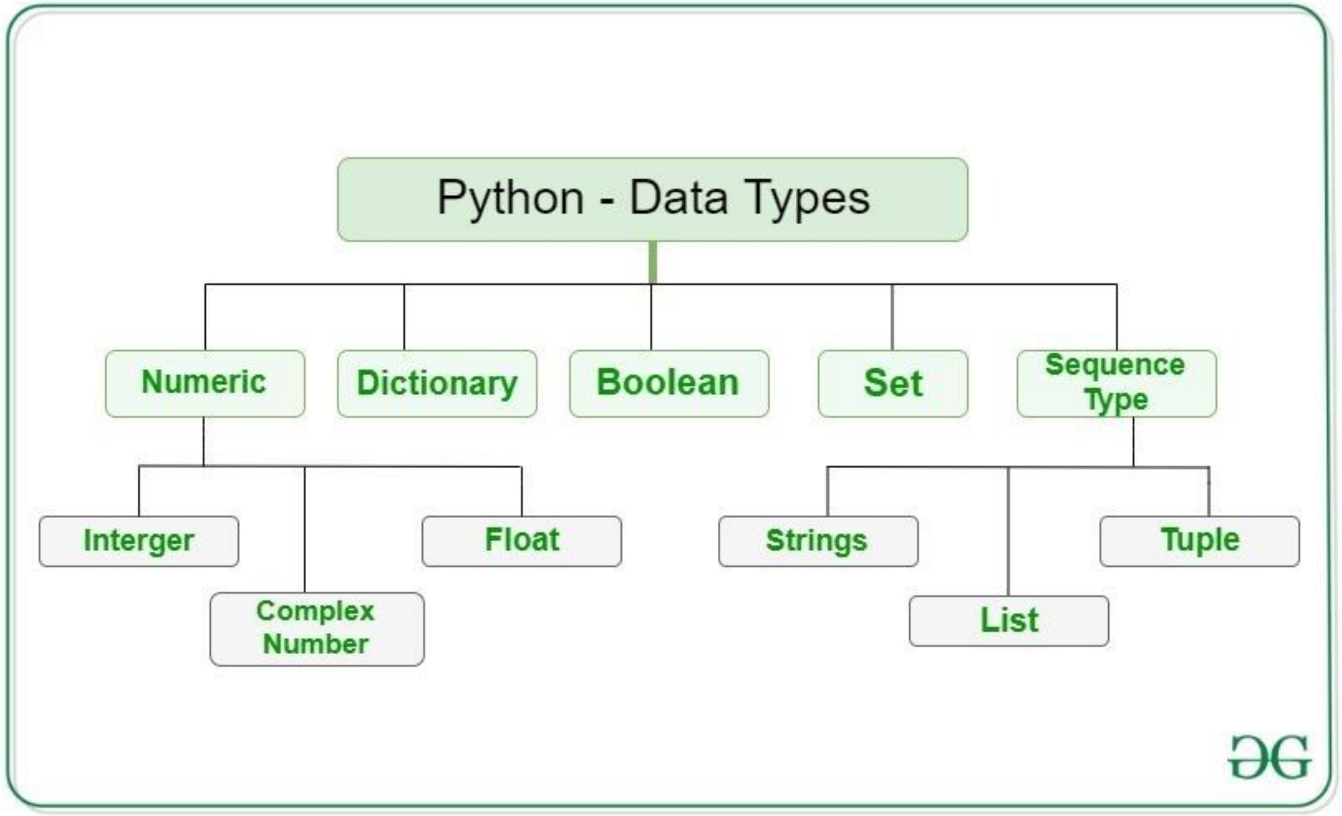
OUTPUT

- print(value1, value2, …, sep=' ', end='\n')
- Can print most of the data types directly

INPUT

- input(prompt)
- Always return a string. Returns a whole line until enter is pressed.

DATA TYPES



Extra type – None
Use type() function to check data type of a variable

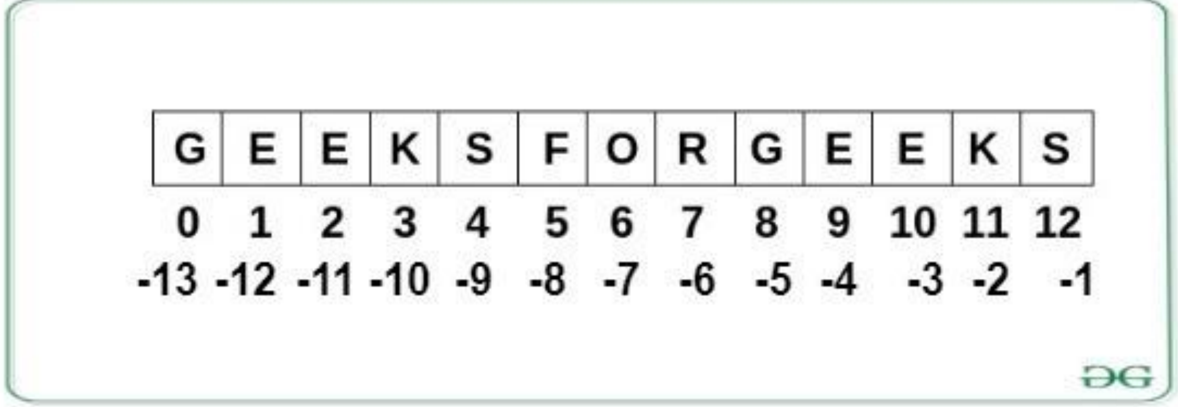
NUMERIC

- In Python, numeric data type represent the data which has numeric value. Numeric value can be interger, floating number or even complex numbers. These values are defined as `int`, `float` and `complex` class in Python.
- Use int(), float() and complex() to convert string and other numeric data types into int, float and complex respectively

BOOLEAN

- Booleans are data type with one of the two built-in values, `True` or `False`. It is denoted by the class bool.
- The two valid boolean values are True and False (notice that the first letter is capitalised).

STRINGS



- A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by `str` class. Strings in Python can be created using single quotes or double quotes or even triple quotes.
- Elements of a string can be accessed by positive and negative indices.
- Strings are immutable (individual elements can not be deleted/updated). But whole string can be deleted using **del** statement.
- Escape sequences same as in C/C++ (eg. \n, \t, \', \", \\ etc..).
- Strings can be sliced (creating substrings) as follows: string[start:end], where elements at [start, end) are included in the sliced string. (Negative indices allowed)
- String formating can be done by the format() method
- Size of string or any other iterable can be found by len(iterable).

LISTS

- Lists are just like the arrays. A single list may contain DataTypes like Integers, Strings, as well as Objects. The elements in a list are indexed according to a definite sequence and the indexing of a list is done with 0 being the first index. It is represented by `list` class.
- Empty lists are created by [ ] or list(). A list with values is created by [value1, value2, value3, ….] or list(iterable).
- **Adding Elements to a List:** Using `append()`, `insert()` and `extend()`.
- Lists are mutable
- **Accessing elements from the List –**
- Use the index operator [ ] to access an item in a list. In Python, negative sequence indexes represent positions from the end of the array. Instead of having to compute the offset as in `List[len(List)-3]`, it is enough to just write `List[-3]`.
- **Removing Elements from the List:** Using `remove()` and `pop()`
- `Lists can be sliced in the same manner as strings.`
- `List(string) gives list of characters in the string.`
- `split() can be used to split a string into parts based on a parameter`
- `map() can be used to create a list of numbers from a list of strings.`

TUPLES

- Tuple is an ordered collection of Python objects much like a list. The important difference between a list and a tuple is that tuples are immutable. It is represented by `tuple` class. In Python, tuples are created by placing a sequence of values separated by 'comma' with or without the use of parentheses for grouping of the data sequence.
- Tuples are immutable.
- Empty tuple can be created by ( ) or tuple(). Tuple with values can be created by (value1, value2, …) with or without paranthesis or tuple(iterable)
  - In the first case for a single value tuple a trailing comma is necessary else it won't be considered as a tuple.
- Accessing tuple elements and slicing of tuples is the same as that of lists.
- Deletion of tuples same as strings.
- Tuple unpacking is also a very useful feature.

SETS

- In Python, **Set** is an unordered collection of data type that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.
- Sets can be created by using the built-in **`set()`** function with an iterable object or a sequence by placing the sequence inside curly braces, separated by 'comma'.
- Elements can be added to the Set by using built-in **`add()`** function. Only one element at a time can be added to the set by using `add()` method
- Set items cannot be accessed by referring to an index, since sets are unordered the items has no index. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.
- Elements can be removed from the Set by using built-in `remove()` function but a KeyError arises if element doesn't exist in the set. To remove elements from a set without KeyError, use `discard()`, if the element doesn't exist in the set, it remains unchanged. To remove all the elements from the set, `clear()` function is used.
- **Frozen sets** in Python are immutable objects that only support methods and operators that produce a result without affecting the frozen set or sets to which they are applied. While elements of a set can be modified at any time, elements of the frozen set remain the same after creation.
  If no parameters are passed, it returns an empty frozenset.

DICTIONARY

- Dictionary in Python is an unordered collection of data values, used to store data values like a map. Dictionary holds `key:value` pair. Each key-value pair in a Dictionary is separated by a colon `:`, whereas each key is separated by a 'comma'
- Empty dictionary can be created by { } or dict().
- Keys can have any data type and immutable sequence.Values can have any data type.
- Key are case-sensitive and should be unique.
- One value at a time can be added to a Dictionary by defining value along with the key e.g. `Dict[Key] = 'Value'.` While adding a value, if the key value already exists, the value gets updated otherwise a new Key with the value is added to the Dictionary.
- In order to access the items of a dictionary refer to its key name or use `get()` method.
- **Removing Elements from Dictionary:** Using `pop()` and `popitem()`

DECISION MAKING

COMPARISON BETWEEN DECISION MAKING IN C/C++ AND PYTHON

C/C++

- if(condition) { … }
- else { . . . }
- else if(condition) { . . . }
- switch case can be used
- Indentation not mandatory

Python

- if condition:

  . . .
- else:

  . . .
- elif condition:

  . . .
- Equivalent of switch case not present
- Indentation mandatory (also for single statement)

CONTROL FLOW

LOOPS

WHILE & WHILE-ELSE LOOP

while loop

- Syntax:-
- while condition:

      . . .

- Working same a while loop in C/C++

while else loop

- Syntax:-
- while condition:

      . . .

  else:

      . . .

- Else part is executed when the condition becomes false, except when the loop terminates using break

LOOP CONTROL STATEMENTS

- The statements **break** and **continue** work in the same way as in C/C++
- The **pass** statement is equivalent of **;** statement in C/C++.
- Pass statement is used to write empty loops. Pass is also used for empty control statement, function and classes.

FOR & FOR-ELSE LOOP

THANK YOU