Python Fundamentals

```
        mirror_mod.use_x = False
        mirror_mod.use_y = True
        mirror_mod.use_z = False
    elif _operation == "MIRROR_Z":
        mirror_mod.use_x = False
        mirror_mod.use_y = False
        mirror_mod.use_z = True

        #selection at the end -add back the deselected mirror modifier object
    mirror_ob.select= 1
    modifier_ob.select=1
    bpy.context.scene.objects.active = modifier_ob
    print("Selected" + str(modifier_ob)) # modifier ob is the active ob
        #mirror_ob.select = 0
```

python™

ROOMAN

INNOVATE INTEGRATE EMPOWER

https://rooman.com/courses/

info@rooman.net

Python Fundamentals

Prerequisites

Any one with a desire to learn

Who Can Attend?

Web masters, Programmers, Professional Software Developers, Big Data Scientists, Analysts, Entrepreneurs, students and anyone with a passion to learn Python will find value in attending this course.

What you will get

- 100+ hours of online live classes
- 200+ coding assignments
- Dedicated mentors
- Talks from industry experts
- Real time projects
- 100% placement assistance

Master the fundamentals of Python.

Building a solid foundation to explore the fields of web development, game development, data science & artificial intelligence. Become Industry 4.0 ready with this comprehensive Python course.

1. **Introduction**
   - Compilation v/s Interpretation
   - Script mode and Interactive mode
   - Command Line Arguments
2. **Data Types**
   - Basic Data types
     a. Numbers (int, float, complex)
     b. Strings
     c. Bool
   - Advance Data types (List, tuple, set, dictionary)
   - Type casting
     a. Implicit
     b. Explicit
3. **Functions**
   - Types of Functions
     a. User Defined Functions
     b. Built-in Functions
     c. Lambda Function
        i. Filter
        ii. Reduce
        iii. Map
     d. Recursive Function
   - Doc String
   - Types of Arguments
     a. Positional arguments
     b. Default arguments
     c. Keyword arguments
     d. Variable length arguments
     e. Variable length Keyword argument

4. Modules in Python

- Importing a Module using alias
- Importing using from keyword
- Input()
- __name__() and __main__()
- Turtle Module
- Math module

5. List

- Creation of lists
- Accessing list elements
- List slicing
- List replication
- Appending two list
  - a. append()
  - b. extend()
  - c. using '+' operator
- Removing an element from a list
  - a. pop()
  - b. del keyword
- Reference Type Assignment
- Copy Operation using memory map
  - a. Shallow copy
  - b. Deep copy
- List Comprehension
  - a. Using for loop
  - b. Using list comprehension
  - c. List comprehension using single if condition
  - d. List comprehension with multiple if conditions
  - e. List comprehension using else condition
- Accessing list
  - a. Using for loop
  - b. Using range()
  - c. Accessing elements present within nested list
- Reversing a list
- List Comparison
- List Sorting

a. Ascending order
b. Descending order
· Membership Check of List

6. Tuples

· Membership Check of List
a. Creation of tuple
b. Creation of singleton tuple
c. Packing and Unpacking
d. Unpacking using disposable variable
e. Accessing elements within a tuple
f. Tuple Slicing
g. Copy operation in tuple
· List and tuple Comparison

7. Set

· Creation of set
· Set operations
a. Union
b. Intersection
c. Difference
d. Symmetric Difference
e. Subset
f. Super set
g. Disjoint set
· Set methods
a. add()
b. discard()
c. remove()
· Frozen set
· Set Comprehension
a. Using for loop
b. Using set comprehension
c. set comprehension using single if condition
d. set comprehension with multiple if conditions
e. set comprehension using else condition
· All and Any function
· Internal Implementation of List

- List performance analysis
- When to use a List.
- Internal Implementation of tuple
- Performance Analysis
- Difference between list and tuple
- Internal Implements of set
- Performance analysis of set
- Difference between list and set
- Difference between tuple and set

8. Dictionary

- Internal Implementation of Dictionary
- Creation of Dictionary
- Adding elements to a dictionary
- Accessing elements from a dictionary
- Accessing values from a dictionary using get()
- Different ways of deleting elements from a dictionary
  - a. pop()
  - b. popitem()
  - c. del keyword
  - d. clear()
- Different ways of accessing a dictionary
  - e. keys()
  - f. values()
  - g. items()
- Different ways of iterating over a dictionary
  - a. keys()
  - b. values()
  - c. items()
- Membership check in a dictionary
- Merging of dictionaries
  - a. Using update()
  - b. Using **
- Dictionary Comprehensions
  - a. Using for loop
  - b. Using dictionary
  - c. Dictionary comprehension using single if condition

- d. Dictionary comprehension with multiple if conditions
- e. Dictionary comprehension using else condition

· When to use a Dictionary

· Zip()

- a. Zip() function on list of varying length

· Difference between List, tuple, set and dictionary

9. Collections Module

· Dequeue

· Named Tuple

· Ordered Dictionary

· Default Dictionary

· Chain map

· Counter

10. String

· Different ways of creating a string

· Internal Implementation of String

· String Formatting

- a. Default formatting
- b. Positional formatting
- c. Keyword formatting
- d. Binary formatting
- e. % Formatting Specifier

· Built-in functions in String

- a. lower()
- b. upper()
- c. title()
- d. capitalize()
- e. swapcase()
- f. maketrans()
- g. translate()
- h. split()
- i. startswith()
- j. endswith()

· Accessing individual character of a String

- a. Forward direction
- b. Reverse direction

- String Comparison
  - a. Using values
  - b. Using Reference
  - c. Ignoring case
  - d. Difference between casefold() and lower()
- String Concatenation
  - a. Using '+' Operator
  - b. Using join()
  - c. Using format()
  - d. Using 'f' string literal

11. Regular Expressions

- Quantifiers
- Word Character
- Character class
- Grouping
- Raw String
- Re Module
  - a. match()
  - b. search()
  - c. Difference between match() and search()
  - d. findall()
  - e. sub()
  - f. split()
  - g. start()
  - h. end()
  - i. group()
  - j. compile()
- Flags

12. Exception Handling

- User Defined Exception Handler
- Disadvantage of having single Except block
- Grouping multiple exceptions in a single except block
- Printing the default message of exception by aliasing exception object
- Optional else block
- Propagation of exception object

- Use of 'raise' keyword in exception handling
- Handling the exception using try-except blocks
- Re-throwing an exception
- Valid and in-valid syntax of try-except block
- Difference between exception handling in python and java
- Customized Exception in python
- Exception Hierarchy
- Scope of variable in python
    a. Accessing global variable within the function
    b. Difference between globals() and locals() methods
    c. Nested function scope in python

13. Loggers

- Levels of Loggers
    a. INFO
    b. DEBUG
    c. ERROR
    d. WARNING
    e. CRITICAL
- Formatters in loggers
- Traceback error log file
- Levels for respective file handling
    a. ERROR
    b. INFO
    c. DEBUG

14. File Handling in Python

- Reading a file
- Modes of file
- Closing a file
- Context Managers in python
- Reading file contents line by line
- Reading a single line in a file
- Reading multiple lines in a file
- Reading a character in a line
- readline()
- Cursor Position
    a. tell()

b. seek()

· Writing a file
· File modes in python
· Read
· Write
· Create
· Erase
· Position
· Exclusive Creation
· Exclusive Mode operation
· File to File transfer

15. Os Modules

16. Object Orientation in Python

· Object creation in python
· Creation of instance variable in python
· __new__()
· __init__()
· self keyword in python
· static and non-static methods in python

17. Iterators and Generators

· Iterators
  a. Difference between containers and non-containers
  b. __iter__() and __next__()
  c. Iter tools module
    i. Islice
    ii. cycle
· Examples
· Generators
  a. Difference between normal function and iterator function
  b. Use of 'yield' keyword in python
  c. Control flow diagram of iterators and generators

18. First class functions

19. Closures

20. Decorators

21. Oops Concept

- Encapsulation
- Polymorphism
- Inheritance
- Abstraction

22. Operator overloading and Magic Methods

23. Multi-Threading in Python