



IIT Madras

ONLINE DEGREE


Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
Omkar Joshi
Course Instructor
Online Degree Programme
Indian Institute of Technology, Bangalore
Course Summary

(Refer Slide Time: 0:16)

Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
1. print() 2. Variables 3. input() 4. Data types 5. Operators 6. Expressions	1. Strings 2. Types of quotes and Escape characters 3. if-elif-else 4. Types of import statements	1. while loop 2. for loop 3. Formatted printing 4. Nested loops 5. break, continue, pass	1. Lists 2. Nested lists 3. Obvious sort 4. Matrix operations	1. Functions 2. Types of functions 3. Types of function arguments 4. Scope of variable	1. Lists 2. Tuples 3. Dictionaries 4. Sets

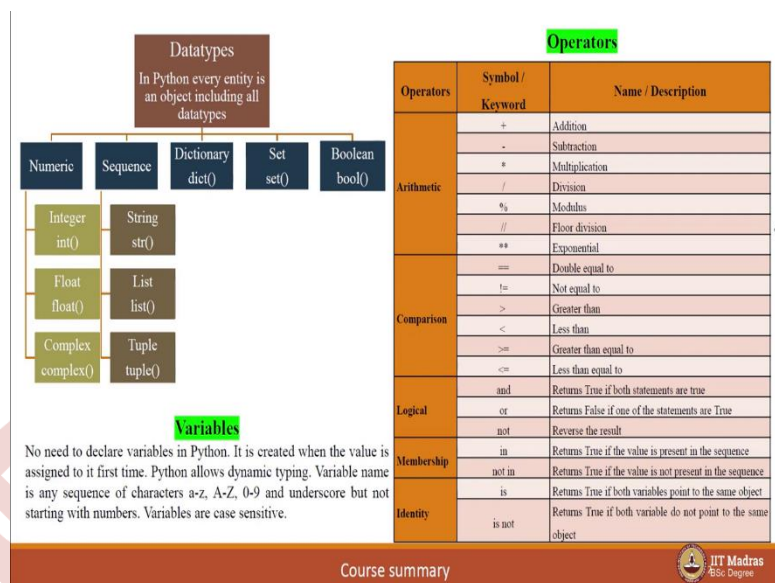
Week 8	Week 9	Week 10	Week 11
1. Recursion 2. Binary search	1. File handling 2. Pandas library	1. Object Oriented Programming 2. Numpy library 3. Matplotlib library	1. Exception handling 2. Functional programming

Course summary

 IIT Madras
BSc. Degree

Hello Python students. In this lecture, we will try to summarize all the contents which we have covered so far in this course. As you can see, in first 6 weeks, we discussed about some core concepts involved in Python language. And then in last 4 weeks, we talked about some little bit advanced concepts related to Python. Now, in this lecture, we will revise all these concepts quickly by summarizing them in small modules. So, let us start that.

(Refer Slide Time: 0:53)



First is datatypes. Now, after learning object-oriented programming, we can comfortably say, every entity in Python is an object, including datatypes. Majorly, there are 5 categories of data types, numeric, sequence, dictionary, set and Boolean. Whereas numeric category is divided into 3 sub datatypes, integer, float and complex numbers.

Similarly, sequence category is also divided into 3 sub datatypes, string, list, and tuple. And as you can observe, in each those box, which represents individual datatype has 2 lines. First line represents the name of the datatype, or the class name of the datatype. Whereas the second line represents the way through which we can create that datatype, as in that second line represents the constructor of that specific class.

After that, we talked about variables. As we know, in Python, we do not have to declare variables in advance. Whenever we assign a value against a variable, that variable gets initialized by default. On top of that, Python also allows dynamic typing, which means datatype of a particular variable can be converted from one form to another. For example, first if we store one integer value inside a variable, then that same variable can be later on used to store a value of different datatype as well. And that is what the dynamic typing is.

After that, we also talked about the rules which we have to follow while naming a variable. Variable name can be any sequence of characters from A to Z in lowercase, uppercase, numbers from 0 to 9, and one special character which is underscore. And in such a sequence, there is only one constraint which says the name of the variable cannot start with a number and variable names are case sensitive.

After that, we talked about operators in Python, majorly, there are 5 categories of operators in Python, namely arithmetic, comparison, logical, membership and identity. We use arithmetic operators in order to perform mathematical operations. Whereas, comparison operators are used mostly with conditional statements or loops. Logical operators are used in expressions, whereas, we use membership operators along with data structures to check the membership of an element. Whereas, identity operators are used majorly with objects to check the identity of individual entity.

(Refer Slide Time: 4:35)

Formatted printing

It allows programmer to take more control over printing output rather than simply printing space separated values.

1. **f-string:**

```
name = input()
print(f'Hi, {name}!')
```
2. **format():**

```
name = input()
print('Hi, {}'.format(name))
```

Types of import statements

1. **import math**
It will import the math library
2. **from math import ***
It will import the entire contents of math library
3. **from math import pi**
It will import only variable pi from math library
4. **import calendar as cal**
It will import the calendar library and it can be accessed as cal
5. **from calendar import month as m**
It will import only month method from calendar library and it can be accessed as m

Escape characters

Backslash '\' is considered as escape character in Python. It is used to insert characters in strings which are illegal otherwise.

Escape character	Result
\\	Backslash
'\'	Single quote
'\"'	Double quotes
\n	New line
\t	Tab

Useful external libraries

[Pandas](#), [NumPy](#), [Matplotlib](#)

Course summary

After this, we talked about formatted printing, where we focused majorly on 2 different types of functions called f string and format. f string allows us to modify our output using print statement in a certain way. Whereas, format also allows us to print output on the console in a specific manner. Using either of these methods as a programmer, we can take more control over how an output is going to get printed on the console.

In addition to that, we also use something called as escape characters, which allows us to use special characters as part of string, which was not possible earlier. Like backslash, single quotes, double quotes, newline character or even the tab. After that, we also talked about importance of libraries and different ways to import a specific library in Python. As you can see, there are 5 different ways using which you can import a library in Python.

First way will import the library. Whereas second method will import all the contents of that library. Third method will import a specific variable or a specific method from that library. Fourth method is similar to first method, but this time, we can import a library with a specific

name. And then fifth method is similar to third method. But once again, the fifth method will allow us to import a specific method or a variable using specific name.

Once we learned about import statements, we also have to talk about the external libraries, which we discussed in this course, which are Pandas, NumPy, and Matplotlib. And then we briefly saw a few examples using these 3 libraries. As we all know, these are external libraries. Hence, they all have their own detailed documentation available over internet. Hence, we have added the links to those specific documentations in this particular slide, you can simply click on any of this library and it will take you to the detailed documentation of that specific library.

(Refer Slide Time: 7:27)

Conditional statements

These conditional statements are used when some kind of decision making is required. Comparison operators are most useful when we use these conditional statements.

if block:
The goal of this statement is to check if the given condition is True or False. If it is True then Python will execute the following indented lines of code.

```
if a > b:  
    print('a is greater than b')
```

if-elif-else blocks:
If all the conditions given in if and/or elif evaluates to be False then Python executes else block.

```
if a > b:  
    print('a is greater than b')  
elif a < b:  
    print('a is less than b')  
else:  
    print('a and b are equal')
```

Inline if-elif-else (Ternary operator):
The above code can be written in a single line as well.

```
print('a is greater than b' if a > b else  
      'a is less than b' if a < b else 'a and b  
are equal')
```

Nested conditional statements:
Any of the above mentioned conditional blocks can be written inside any other conditional blocks and such kind of structure is referred as nested if-else.

Course summary

IIT Madras
BSc Degree

Moving on, next is conditional statements, conditional statements are nothing but those if else blocks, we talked about if, we talked about elseif, we also talked about else block. And we all know how the execution of if elseif and else works together. After that, we also talked about something called as inline if-else, or popularly known as ternary operator, which allows you to write the entire if else score block into a single line.

And if you remember, this will not improve the efficiency of the code. But it will certainly help you while writing some simple if else blocks. Instead of writing that code in number of lines, we can instead replace it with a single line. After that, we also saw there are all sorts of nesting is possible with respect to this conditional statement, you can have if else inside if, you can have if else inside else if, inside else, and so on. So, all sorts of different combinations of this nesting is possible with conditional statements.

(Refer Slide Time: 8:50)

Loops

while loop:
It enables you to execute set of statements as long as the given condition is True.

```
i = 0
while i < 10:
    print(i)
    i += 1
```

This code will print numbers from 0 to 9.

for loop:

1. Using range():

```
for i in range(10):
    print(i)
```

This code is equivalent to the above while loop and it will also print numbers from 0 to 9.
2. Without using range():

```
for i in 'Hello all, Welcome to Python':
    print(i)
```

This code will print the given string one character at a time in every iteration.

Nested loops:
Any of the above mentioned loops (while and 2 versions of for) can be written inside any of these loops and such kind of structure is referred as nested iterations/loops.

Loop control statements:
Loops are used to automate repetitive statements. But sometimes there may arise a condition where we may want to terminate the loop, skip an iteration or ignore that condition. In such conditions we use loop control statements like **break**, **continue** and **pass** respectively.


range(start, end, step):
It is an in-built function which returns a sequence of numbers based on the values of start, end and step.

start: An integer number specifying at which position to start.
It is an optional argument. Default value is 0.

end: An integer number specifying at which position to stop (non-inclusive).

step: An integer number specifying the incrementation/decrementation.
It is an optional argument. Default value is 1.

Course summary



IIT Madras
BSc Degree

After conditional statements, the most useful component of any programming language is loops. In Python, we saw two different types of loops while and for. In for loop, we also talked about two different variations of for loop, one using a range function, whereas other one without using range function. And this second category of for loop is also referred as for each, both types of for loops, and while loops can be used inside each other.

Hence, once again, all sorts of different nesting is possible and allowed by Python language. Loop allows us to automate repetitive steps. But sometimes due to some situation in the specific program. We may prefer to have control over the way the iterations are happening in a specific loop. And that is the place where we use these three loop control statements like break, continue and pass.

Break allows us to terminate the loop, continue allows us to skip an iteration, whereas pass allows us to ignore a specific condition. And as we were discussing about for loop using the range function, hence this range function becomes very important with respect to Python programming language. So, let us look at that range function in detail. This range function can take maximum three arguments.

The first one is start, second is end and third is step. Start represents the starting point for that particular range. It is an optional argument and the default value is 0. End specifies the endpoint for the sequence and this endpoint is non inclusive. And this particular argument is a mandatory argument is the right range function with only one number.

As you can see over here, in the first example of for loop, that number is considered as the N value. And the third argument is step. Once again, it is an integer value, where we can specify with how much margin you want to increment or decrement the values in the sequence, it is an optional argument, and the default value is 1.

(Refer Slide Time: 11:48)

In-built functions

- `print()`: Prints argument(s) on to the console
- `input()`: Takes input from the console as a string
- `type()`: Returns the type of an object
- `len()`: Returns the length of an object
- `max()`: Returns the largest value in the iterable
- `min()`: Returns the smallest value in the iterable
- `sum()`: Sums the elements in an iterator
- `sorted()`: Returns the sorted list
- `iter()`: Returns an iterator object
- `next()`: Returns the next element in the iterator object
- `enumerate()`: Returns an enumerate object by adding counter to an iterable
- `zip()`: Returns an iterator after coupling two or more iterators
- `map()`: Returns the iterator after applying specific function on each element in it
- `filter()`: Returns an iterator after applying specific filtering function

User defined functions

Along with in-built functions, Python allows you to define your own functions called user defined functions. Function is a block of code which executes only when it is called. Function can have parameter(s) which is/are used to store passed argument(s) from the call. It can return a value back to its call as a result. Function is defined using `def` keyword.

How to define a function?

```
def add(x, y):
    return x + y
```

How to call a functions

```
result = add(10, 20)
```

Types of function arguments

- 1. Positional arguments:** Mapping of arguments and parameters happens as per the sequence


```
def myFunction(x, y, z):
    return x + y - z
result = myFunction(10, 20, 30)
```
- 2. Keyword arguments:** Mapping of arguments and parameters is given explicitly


```
def myFunction(z, y, x):
    return x + y - z
result = myFunction(x = 10, z = 30, y = 20)
```
- 3. Default arguments:** Mapping of arguments and parameters is either explicit or positional but if argument is missing then the default value is considered. Default arguments should be assigned at the end of the list of parameters.


```
def myFunction(z, y = 20, x = 10):
    return x + y - z
result = myFunction(30)
```

Course summary

IIT Madras
BSc. Degree

As we were talking about some inbuilt functions like range, there are many other inbuilt functions, which we should learn along with it. Like print, which allows us to print some value on the console, input allows us to accept some value as an input from console, type is a function which allows us to check the type of a specific object, len returns the length of an object, max function gives us the largest value in the iterable, min function gives us the smallest value in the iterable.

sum function gives the sum of the elements in the iterator, sorted function sorts the given input and returns a list, iter returns an iterable object, next is a function which allows us to iterate over any iterator object, enumerator returns an enumerate object by adding counter to an iterable, zip returns an iterator after coupling two or more iterators, map returns iterator after applying specific function on each element inside that particular iterator, filter function returns an iterator after applying specific filtering function on every value inside that iterator.

But as we know these inbuilt functions are not sufficient in most of the cases. Hence, we have to write our own functions. And those functions are referred as user defined functions. User defined functions can also take n number of parameters, they can return values, and so on. And as you can see here, this is how we can define a function or call a function.


These functions can have three different types of arguments first, are purely based on the position or the sequence in which the arguments are written. Hence, they are referred as positional arguments. The second category is referred to as keyword arguments, in this case, along with arguments we also have to specify the parameter variable explicitly. Hence, in this case, the sequence or the position of any argument is not important.

And then the third category is default argument. This allows us to have a default value for a specific argument. If user provides that value, then we override default value. Or if user does not provide a value, we continue the execution of the function with the default value exactly the way we saw in range function. And the only constraint is the default arguments should be assigned at the end of list of parameters.

(Refer Slide Time: 15:17)

Property	List Video link	Tuple Video link	Dictionary Video link	Set Video link
Notation	[]	()	{'Key': 'Value'}	{ }
Creation	list()	tuple()	dict()	set()
Mutability	Mutable	Immutable	Mutable	Mutable
Type of elements which can be stored	Any	Any	Keys: Hashable Values: Any	Hashable
Order of elements	Ordered	Ordered	Unordered*	Unordered
Duplicate elements	Allowed	Allowed	Keys: Not allowed Values: Allowed	Not allowed
Operations	Add, Update, Delete	None	Keys: Add, Delete Values: Add, Update, Delete	Add, Delete
Operations	Indexing, Slicing, Iteration	Indexing, Slicing, Iteration	Iteration	Iteration
Sorting	Possible	Not possible	Possible	Not possible

*Python 3.6 and earlier. Dictionaries are ordered as per Python 3.7 and above.

Course summary 

Moving to next concept, which is collections in Python, as in data structures in Python, we saw four different types of collections in Python list, tuple, dictionary and set. And this particular table provides the brief overview of all these four collections in Python. For more details, we have added the specific video links of these four collections in this particular table itself. These are the same lectures, which you studied earlier in week 6. In fact, this table representing the comparison between these four collections was also available in that particular week.

(Refer Slide Time: 16:11)

Recursive function	Exception handling
<p>When a function calls itself in its definition then it is called as recursive function</p> <pre>def add_N_numbers(n): if n == 1: return n else: return n + add_N_numbers(n - 1) result = add_N_numbers(5)</pre>	<p>Errors which occur during execution of the Python program are termed as exceptions. Exceptions are different from syntax errors. Python provides try - except blocks in order to handle such exceptions.</p> <pre>try: print(a / b) print(myDictionary['xyz']) except ZeroDivisionError: print('Denominator cannot be zero') except KeyError: print('This key is not there in the dictionary')</pre>
<p>lambda function</p> <p>It is an anonymous function which can have any number of arguments but can execute only one expression.</p> <pre>f = lambda a, b, c: a + b - c result = f(10, 20, 5)</pre>	<p>The above code is syntactically correct and will execute successfully if value of variable b is non-zero and 'xyz' is a key in myDictionary. But at the same time it will throw an exception if either variable b is zero or myDictionary does not have the key 'xyz'.</p> <p>Complete list of in-built exceptions.</p>
<p>List comprehension</p> <p>It is an optimized way to create a list in a single line.</p> <pre>myList = [x ** 2 for x in range(10) if x % 2 == 0]</pre>	<p>finally block:</p> <p>It is a special type of code block which always executes after successful execution as well as after termination due to execution. Therefore it is used to deallocate the system resources used in the program, e.g. file pointers.</p>

Course summary

IIT Madras
BSc Degree

So, instead of spending too much time on this, let us move to the next concept, which is recursive function. Now, from this slide onwards, we will focus on those last 4 weeks, where we discussed about some advanced concepts in Python, like recursive function. Recursive function is a type of a function, which calls itself, as in function calls itself in its own definition. And in such a case, the function is referred as, a recursive function.

After that, we also talked about one more different type of function called lambda function. Lambda function is an anonymous function, which can take any number of arguments, but can execute only one expression. And this is the example of such an anonymous function. With respect to functional programming aspect, we also saw one more component called list comprehension. This allows us to create a list in more optimized manner.

As you can see, here, we are creating a list using iterator and a conditional statement. Even though we are doing all this it is happening in just one line. Along with functional programming, we also talked about one very important aspect of any programming language called exception handling. We also saw exception handling is considered as a standard way to write a program. And as we all know exceptions are different from syntax errors.

Errors, which occurred during execution of a program, are termed as exceptions. And Python provides something called as try except blocks. In order to handle such exceptions. You must also remember, there are a very long list of such inbuilt exceptions available in Python. Hence, once again, we have provided an external link to exception handling documentation provided by Python, where you can see the list of all such exceptions at a one place.

Along with try except blocks, we also discussed about one more block called finally. And this is a special core block, which always executes irrespective of whether the program runs successfully or it terminates with an exception. In either cases, this finally block will always execute. Therefore, it is a good practice to deallocate the system resources used in program, in finally block. For example, closing of a file pointer.

(Refer Slide Time: 19:21)

File handling

We store our entire information on computer systems using files. Hence, there has to be some way to open/read/write files using Python.

How to open/create a file?
It is done using open() function which takes two parameters, first is file name with its extension and second is mode for opening file.

```
f = open('abc.txt', 'r')
```

How to read a file?
There are three major functions through which we can read files.
f.read(): It reads the specified number of bytes from the file. The default/optional argument is -1 which indicates the entire file content.
f.readline(): It reads the file one line at a time.
f.readlines(): It reads the entire file as a list of strings representing individual lines.

How to write a file?
In order to write on to the file, we must open it using either 'w' or 'a' mode. And, then we can use f.write() function to write content.

Operation	Parameter	Description
Create	'x'	Creates the specified file, returns an error if the file exists
Read	'r'	Default value. Opens a file for reading, error if the file does not exist
Append	'a'	Opens a file for appending, creates the file if it does not exist
Write	'w'	Opens a file for writing, creates the file if it does not exist
Text	't'	Default value. Text mode
Binary	'b'	Binary mode (e.g. images)

Course summary

Along with exception handling, we also talked about file handling. We store all our information digitally using files. Hence, there has to be some way through which we can read, write those files using Python. And that is why we studied some specific functions, which are used in order to open or create a file, which is an open function. It takes two parameters.

First is file name and second is the mode in which a file can be opened. This mode can be create, read, append or write. And the type of file can be either text or binary, like an image. Specifically for reading a file, we have three major functions called read, readline, and readlines. And for writing, we use a standard function called write.

(Refer Slide Time: 20:27)

Object Oriented Programming

Python is an object oriented programming language. Hence, every entity in Python is an object with its associated attributes and methods.

How to create class?

```
class myClass:
    def __init__(self):
        pass
```

How to create object?

```
obj = myClass()
```

How to add attributes and methods to class?

```
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def display(self):
        print(self.name, self.age)
```

```
s1 = Student('ABC', 10)
s1.display()
```


How to implement inheritance?

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def display(self):
        print(self.name, self.age)
```

```
class Student(Person):
    def __init__(self, name, age, marks):
        super().__init__(name, age)
        self.marks = marks
    def display(self):
        super().display()
        print(self.marks)
```

```
s1 = Student('ABC', 10, 90)
s1.display()
```

What are the types of inheritance?
Simple, Hierarchical, Multiple, Multilevel and Hybrid.

Course summary 

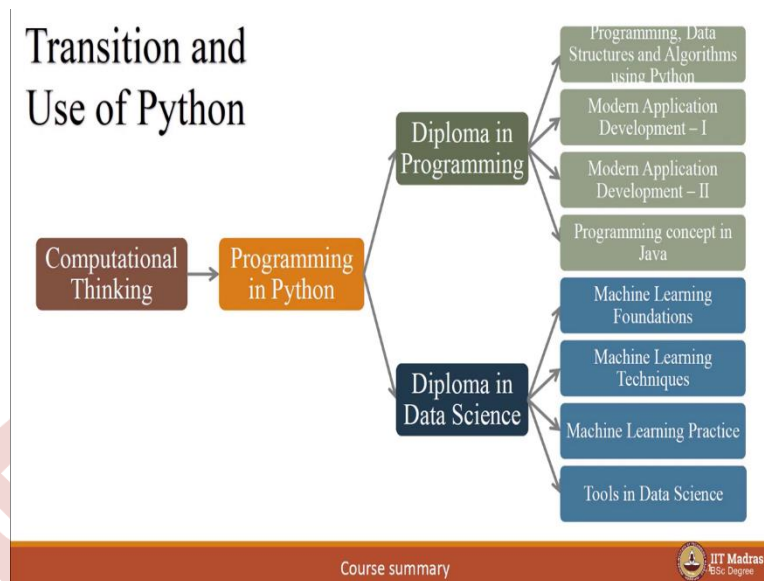
After file handling, we also talked about one different way of approaching a Python program. And that new approach is referred as object-oriented programming. This particular concept allowed us to translate our real time objects and the relations between objects into a Python program. And that is where we introduced the concept of classes and objects.

Class is created using a keyword called class, whereas object can be created using a constructor of a class. Every class can have attributes and methods, these attributes can be initialized using a special method called init. All the functions which are defined inside a class are referred as methods. Along with some default methods of every class, we can also define some user defined methods as well, like this display method over here.

And as I mentioned earlier, object-oriented programming allows us to translate not just object relations between objects as well. And that is when we discuss something called as inheritance, where we discussed how one class can be created by taking help of another class. And in such a case, the original class is referred as parent class or a base class or a superclass.

Whereas, this newly created class is referred as child class or derived class or subclass. We also talked about various different types of inheritance. And these types are simple hierarchical, multiple, multi-level and hybrid and that is all we have studied in this particular course.

(Refer Slide Time: 22:51)



Now, you all must be wondering what after this and the answer is this particular flow chart over here, which shows the transition of your study from computational thinking to programming in Python to two levels of diploma available in this particular program. These are only few courses which belong to that particular diploma. In order to get either of these diplomas, there are some n number of courses which you have to complete. Out of which these are some courses where you will use Python programming language or you will use some concepts which you studied in this language.

For example, if we consider Diploma in programming, then the subjects like programming data structures and algorithm using Python, modern application development 1 and 2 will be taught using Python. Whereas the concepts like exception handling, object-oriented programming, inheritance and so on, you will study once again during programming concept in Java course. Over there, the implementation of that concept might vary, but still a concept which you studied here will remain the same.

On the other side, which is diploma in data science. Once again, the courses like machine learning foundations, techniques, and practice will be taught using Python language. Whereas the course like tools in data science will involve a lot of different tools. And out of those mini tools, one very important tool will be Python language. And as you know, there are many other courses involved in these two diplomas. And even after that, there is one more level which is degree level.

But at this point, let us not talk about that level. Let us go one step at a time, software computational thinking. We studied programming in Python. Now, you have to start thinking

about courses involved in diploma level. And especially those where you can use this Python language in order to study those courses. And with that, I conclude this lecture, and this course. All the best. Thank you for watching. Happy learning.

