String Slicing

Ethans
Learn from experts

```
>>> # Slicing Syntax
>>> # string[START:STOP:STEP]
>>>
>>> name[0:5:1]
'Ethan'
>>> name[0:]
'Ethans'
>>> name[:]
'Ethans'
>>> name[:9]
'Ethans'
>>> name[::2]
'Ehn'
```

```
>>> name[-1:-5]
''
>>> name[-1:-5:-1]
'snah'
>>> name[::-1]
'snahtE'
>>> name[::-2]
'sat'
```


String Formatting

Ethans
Learn from experts

```
>>> print "We at %s turning %d today" %('Ethans', 2)
We at Ethans turning 2 today
>>> name, age = 'Ethans', 2
>>> print "We at %s turning %d today" %(name, age)
We at Ethans turning 2 today
>>> print "We at %s turning %s today" %(name, age)
We at Ethans turning 2 today
>>> print "We at %r turning %r today" %(name, age)
We at 'Ethans' turning 2 today
>>>
>>> print 'This is my File name: C:\name\test'
This is my File name: C:
ame        est
>>> print r'This is my File name: C:\name\test'
This is my File name: C:\name\test
```

List

Ethans
Learn from experts

The list is another datatype in Python which can be written with comma-separated values (items) in a square brackets.

Items in a list should be of same type or different type.

```
>>> names = ['Ethans', 'Ethan', 'Ethan Tech']
>>> names[0]
'Ethans'
>>> names[-1]
'Ethan Tech'
>>> names[-2]
'Ethan'
>>> type(names)
<type 'list'>
```


List Functions

Ethans
Learn from experts

```
>>> names
['Ethans', 'Ethan', 'Ethan Tech']
>>> len(names)
3
>>> any(names)
True
>>> all(names)
True
>>> names + names
['Ethans', 'Ethan', 'Ethan Tech', 'Ethans', 'Ethan', 'Ethan Tech']
>>> names * 2
['Ethans', 'Ethan', 'Ethan Tech', 'Ethans', 'Ethan', 'Ethan Tech']
```

List Slicing and range function

Ethans
Learn from experts

```
>>> names
['Ethans', 'Ethan', 'Ethan Tech']
>>> # names[START:STOP:STEP]
>>>
>>> names[0:]
['Ethans', 'Ethan', 'Ethan Tech']
>>> names[0:2]
['Ethans', 'Ethan']
>>> names[-1:-3]
[]
>>> names[-1:-3:-1]
['Ethan Tech', 'Ethan']
```

```
>>> # range(START:STOP:STEP)
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1,10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1,10,2)
[1, 3, 5, 7, 9]
```


List Data Structure

Ethans
Learn from experts

```
>>> # List of List, 2 dimensional
>>> names_age = [ ['Jatin', 35], ['Rahul',16], ['Vijay', 30]]
>>> names_age[0]
['Jatin', 35]
>>> names_age[0][0]
'Jatin'
>>> names_age[-1]
['Vijay', 30]
>>> names_age[-1][1]
30
```

Tuple

Ethans
Learn From experts

Tuple is another datatype in Python which can be written with comma-separated values (items) in a brackets.

Items in a tuple should be of same type or different type doesn't matter.

```
>>> names = ('Ethans', 'Ethan', 'Ethan Tech')
>>> type(names)
<type 'tuple'>
>>> names = ('Ethans')
>>> type(names)
<type 'str'>
>>> names = ('Ethans',)
>>> type(names)
<type 'tuple'>
```


Difference between Tuple and List

Ethans
Learn From experts

```
>>> names_tuple = ('Ethans', 'Ethan', 'Ethan Tech')
>>> names_list = ['Ethans', 'Ethan', 'Ethan Tech']
>>>
>>> names_list[0] = 'ETHANS'
>>> names_tuple[0] = 'ETHANS'

Traceback (most recent call last):
  File "<pyshell#225>", line 1, in <module>
    names_tuple[0] = 'ETHANS'
TypeError: 'tuple' object does not support item assignment
>>>
>>> # Mutable and immutable property of an object
```

Tuple Data Structure

Ethans
Learn From experts

```
>>> names_tuple = (('Ethans', 'Ethan'), ('Ethan Tech',))
>>> len(names_tuple)
2
>>> names_tuple[0]
('Ethans', 'Ethan')
>>> names_tuple[0][1]
'Ethan'
```

```
>>> names_tuple_list = (['Ethans', 'Ethan'], ['Ethan Tech'])
>>> names_tuple[0][1]
'Ethan'
>>> names_tuple[0][1] = 'ETHANS'

Traceback (most recent call last):
  File "<pyshell#241>", line 1, in <module>
    names_tuple[0][1] = 'ETHANS'
TypeError: 'tuple' object does not support item assignment
>>>
>>> names_tuple_list[0][1] = 'ETHANS'
```


Dictionary

Ethans
Learn From experts

Another builtin datatype in Python where each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces.

```
>>> details = {'Name':'Ethans', 'Age': 2, 'Location':'Pune'}
>>> details[0]

Traceback (most recent call last):
  File "<pyshell#250>", line 1, in <module>
    details[0]
KeyError: 0
>>> details['Name']
'Ethans'
>>> print details # Random order
{'Age': 2, 'Name': 'Ethans', 'Location': 'Pune'}
```

Add and Remove Dictionary Item

Ethans
Learn from experts

```
>>> details
{'Age': 2, 'Name': 'Ethans', 'Location': 'Pune'}
>>> details['Name'] = 'Ethans Tech'
>>> details['Technology'] = 'Python' # Adding New Element
```

```
>>> details
{'Age': 2, 'Technology': 'Python', 'Name': 'Ethans Tech', 'Location': 'Pune'}
>>> del details['Age']
>>> details
{'Technology': 'Python', 'Name': 'Ethans Tech', 'Location': 'Pune'}
```


Dictionaries and List, Tuple

Ethans
Learn from experts

```
>>> print tuple(details)
('Technology', 'Name', 'Location')
>>> print list(details)
['Technology', 'Name', 'Location']
>>> a = dict(one=1, two=2, three=3)
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict([('two', 2), ('one', 1), ('three', 3)])
>>> a == b == c
True
```

Dictionaries data Structure

Ethans
Learn from experts

```
>>> emp = {'Names':['Ethan', 'Ethans'], 'Location':['Pune', 'Bangalore']} # DoL
>>> emp = {'Names':('Ethan', 'Ethans'), 'Location':('Pune', 'Bangalore')} # DoT
>>> emp = {'Names':{'1':'Ethan', 2:'Ethans'}, 'Location':{1:'Pune',2:'Bangalore'}} # DoD
>>> emp
{'Names': {'1': 'Ethan', 2: 'Ethans'}, 'Location': {1: 'Pune', 2: 'Bangalore'}}
>>> emp['Names']['1']
'Ethan'
>>> emp['Location'][1]
'Pune'
```


Set Object

Ethans
Learn from experts

```
>>> managers = set(['Aakash', 'Rahul', 'Bob'])
>>> engineers = set(['Rahul', 'Vijay'])
>>> type(managers)
<type 'set'>
>>> # Get the intersection
>>> managers & engineers
set(['Rahul'])
>>> managers - engineers # elements available in managers not in engin
eers
set(['Bob', 'Aakash'])
>>> managers | engineers # elements available in both
set(['Bob', 'Vijay', 'Aakash', 'Rahul'])
```

Boolean and None Object

Ethans
Learn from experts

```
>>> yes = True
>>> type(yes)
<type 'bool'>
>>> no = False
>>> type(no)
<type 'bool'>
>>>
>>> Null = None
>>> type(Null)
<type 'NoneType'>
```

True and False are the pre defined objects in Python, when comparing the value or doing comparison operation Python returns either True or False.

None is another object in Python.
It is similar as NULL in database.

Membership Operators

Ethans
Learn from experts

Python membership operators test for membership in a sequence or an iterable, such as strings, lists, or tuples.

`in`

Evaluates to `true` if it finds a variable in the specified sequence and `false` otherwise.

`not in`

Evaluates to `true` if it does not finds a variable in the specified sequence and `false` otherwise.

```
>>> 'is' in 'This is a string'
True
>>> 'IS' not in 'This is a string'
True
>>> 1 in range(10)
True
>>> 10 not in range(10)
True
```

Python Identity Operator

Ethans
Learn from experts

`is`

Evaluates to `true` if the variables on either side of the operator point to the same object and `false` otherwise.

`is not`

Evaluates to `false` if the variables on either side of the operator point to the same object and `true` otherwise.

```
>>> number1 = 10
>>> number2 = 300
>>>
>>> number1 is number2
False
>>> number1 is not number2
True
>>> number1 is 10
True
>>> number2 is 300
False
```


Python Identity Operator

Ethans
Learn from experts

`is`

Evaluates to `true` if the variables on either side of the operator point to the same object and `false` otherwise.

`is not`

Evaluates to `false` if the variables on either side of the operator point to the same object and `true` otherwise.

```
>>> name = 'Ethan'
>>> name is 'Ethan'
True
>>> names = ['Ethans', 'Ethan', 'Ethan Tech']
>>> names2 = names
>>> name is names
False
>>> names2[0] = 'ETHANS'
>>> names
['ETHANS', 'Ethan', 'Ethan Tech']
```

Quiz:

Ethans Learn from experts

1 – What will be the output of below program?

```
>>> sum([1,2,3,4,5])
```

2 – What will be the output of below program?

```
>>> name = 'Ethans Technologies'
>>> name[0:6]
```

3 – What will be the output of below program?

```
>>> name = 'Ethans Technologies‘
>>> name[-1:0]
```


Interesting Fact

Ethans Learn from experts

Raspberry Pi is a card-sized, inexpensive microcomputer that is being used for a surprising range of exciting do-it-yourself stuff such as robots, remote-controlled cars, and video game consoles.

With Python as its main programming language, the Raspberry Pi is being used even by programmers to build radios, cameras, arcade machines, and pet feeders!

With Raspberry Pi mania on the uptrend, there are countless DIY projects, tutorials, and books to choose from online.

These will help you branch out from your "hello world" starter programs to something you can truly be proud of.

Objective – Module 3

Ethan's Learn from experts

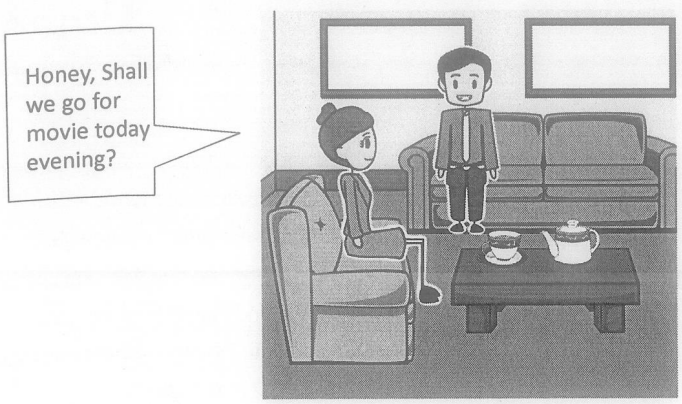
Conditional Statements and Loops

- What are conditional statements?
- How to use the indentations for defining if, else, elif block
- What are loops?
- How to control the loops
- How to iterate through the various object
- Sequence and iterable objects

Slide 61

Choices in Real life

Ethan's Learn from experts

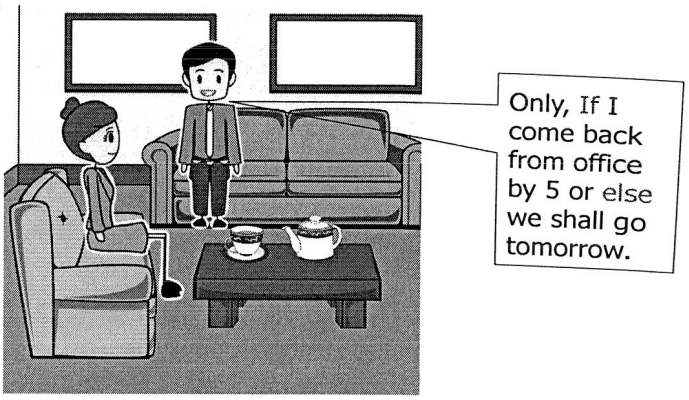


Slide 62

www.ethans.co.in

Condition statements

Ethans
Learn from experts

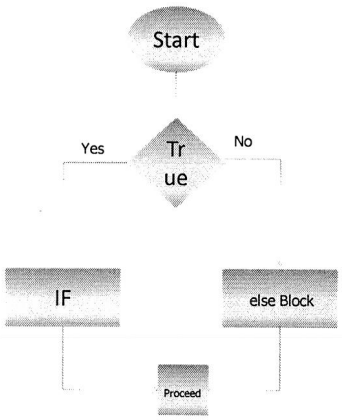


Slide 63

www.ethans.co.in

If else Condition statements

Ethans
Learn from experts



→Variable alone doesn't support conditions.
→If-elif-else like clauses used to make conditions based on some pre-conditions.
→In Python, we have the reserved keywords like if, else, elif and unless for conditions.
→The statements to be executed are enclosed within braces indentation block
→All defined objects like numbers, string, list, tuple, dict etc returns true when they are defined with true value.

Slide 64

www.ethans.co.in

Operators

Ethans
Learn from experts

In module 2, we have listed the condition operators.

In Python, we have following operators which evaluate either True or False

Type	Operators
Conditional Operator	== (equals), != (Not equal), > (greater than), < (less than), <= (less then or equals), >= (greater than or equals)
Membership Operator	In, not in
Identity Operator	Is, is not

If - Else Example

Ethans
Learn from experts

```
number = 10
if number: # Not condition, return True or False based on Value
   print "Got a true expression value ",
   print number

number = 0
if number: # Check the indentation of the block
   print "2 - Got a true expression value ",
   print number
else:
   print "2 - Got a False expression value ", number
```

False Value

Ethans
Learn From experts

Python conditions is either True or False.

Following are certain example of evaluated False value in the conditions.

- 0, 0.0, 0L - Numbers
- '' – An Empty String
- [] – An Empty List
- {} - An Empty dictinoary
- () - An Empty tuple
- None
- False
- Set() – An Empty set

If elif else statement

Ethans
Learn From experts

```
number = 10
if number == 100:
   print "1 - The number is equals to 100 ",
   print number
elif number == 150:
   print "2 - The number is equals to 150 ",
   print number
elif number == 200:
   print "3 - The number is equals to 200 ",
   print number
else:
   print "4 - Got a different value"
```

Logical Operator

Ethan's Learn from experts

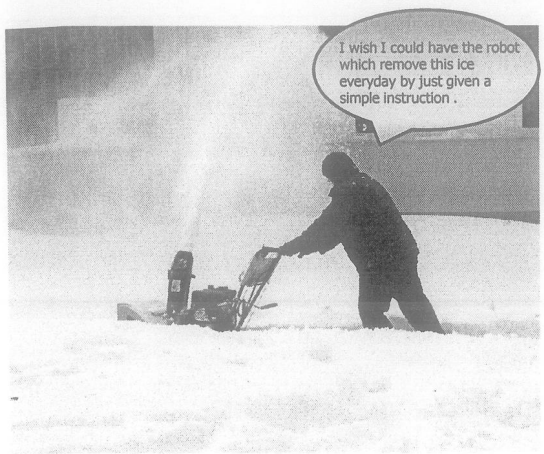
```
number1, number2, number3 = 10, 20, 0

if number1 and number2: # If both number1 and number2 are True
   print "number1 and number2 are True"
else:
   print "Either number1 is not true or number2 is not true"

if number1 or number2:
   print "Either number1 is true or number2 is true or both are True"
else:
   print "Both number1 and number2 are False"
```


Repeated Action

Ethan's Learn from experts



Loops

Ethans
Learn From experts

Loops are the repetitive action/s to perform any task.

They are also called as iterative constructs statements.

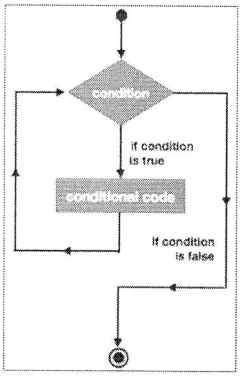
Python provide us below reserved keywords for repetitive actions

- While loop
- For loop

While Loop

Ethans
Learn From experts

- while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.



Syntax

```
while expression:
   statement(s)
```

While Loop - Example

Ethans
Learn from experts

```
count = 0

# Indentation is required for while loop
while count < 9:
   print 'The count is:', count
   count = count + 1
```


while loop – infinite loop

Ethans
Learn from experts

- A loop becomes infinite loop if a condition never becomes false.
- This results in a loop that never ends. Such a loop is called an *infinite loop*.
- Example - client/server programming.

Example

```
var = 1
while var == 1 :  # This constructs an infinite loop
   num = raw_input("Enter a number  :")
   print "You entered: ", num

print "Good bye!"
```

Note - It will go in an infinite loop and use CTRL+C to come out of the program.

Else with while

Ethans
Learn From experts

- Python supports to have an else statement associated with a loop statement.
- When the else statement is used with a while loop, the else statement is executed when the condition becomes false.

```
count = 0

# Indentation is required for while loop
while count < 9:
   print 'The count is:', count
   count = count + 1
else:
    print 'While loop completed'
```

Slide 75

www.ethans.co.in

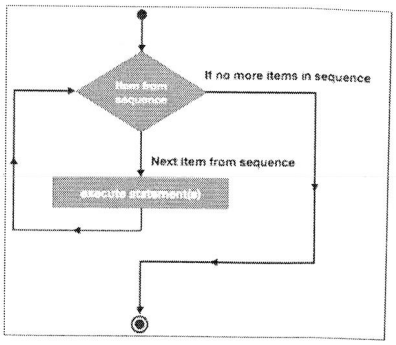
For loop

Ethans
Learn From experts

- The for loop in Python has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

```
for iterating_var in sequence:
   statements(s)
```



Slide 76

www.ethans.co.in

For loop - Example

Ethans
Learn from experts

```
for letter in 'Python':     # First Example
   print 'Current Letter :', letter

fruits = ['banana', 'apple',  'mango']
for fruit in fruits:        # Second Example
   print 'Current fruit :', fruit

for index in range(len(fruits)): # Third Example
   print 'Current fruit :', fruits[index]
```


Controlling Loop

Ethans
Learn from experts

- Loop control statements change execution from its normal sequence.
- When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
- Python supports the following loop control statements –
  - `break` statement
  - `continue` statement
  - `pass` statement

Break, continue and pass

Ethans
Learn from experts

- The `break` statement in Python terminates the current loop and resumes execution at the next statement.
- The `break` statement can be used in both `while` and `for` loops.

- The `continue` statement in Python returns the control to the beginning of the `while` or `for` loop.
- The `continue` statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.

- The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.
- The pass statement is a null operation; nothing happens when it executes.

Did we know?

Ethans
Learn from experts

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. It is built over the Simple DirectMedia Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games (mainly the graphics part) can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

Pygame was built to replace PySDL after its development stalled.[4] Pygame was originally written by Pete Shinners and is released under the open source free software GNU Lesser General Public License. It has been a community project since 2004 or 2005.