# Try-except

Ethans
Learn from experts

➢ If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```





# About Try-except

Ethans
Learn from experts

➢A single try statement can have multiple except statements. This is useful when the try block contains statements that may throw different types of exceptions.

➢You can also provide a generic except clause, which handles any exception.

➢After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception.

➢The else-block is a good place for code that does not need the try: block's protection

➢exc_info is the method in the sys module, provide the explanation of the exception

# Else with try

Ethans
Learn from experts

## try...else

- You can use an else block with a try block
- Else will execute when no exception is raised by try block.

**Syntax**

```
try:
   piece of code will be written here
   ......................
else:
   This blcok will execute when there is no
exception
   ......................
```





# Finally with try

Ethans
Learn from experts

## try...finally

- You can use an finally block with a try block
- Finally will execute, in both the cases

**Syntax**

```
try:
   piece of code will be written here
   ......................
finally:
   This block will execute when there is exception or no excp
   ......................
```

# Raise exceptions

Ethans
Learn from experts

## Raise Exception

You can also raise exceptions by using the raise statement.

### Syntax

```
raise [Exception [, args]]
```

- Exception is the type of exception
- args is a value for the exception argument. It is an optional argument; if not supplied, the exception argument is None.
- args will catch while create an object of Exception class, shown in demo





# Raise exceptions

Ethans
Learn from experts

➢ The raise statement allows the programmer to force a specified exception to occur. For example:

```
>>> raise NameError('HiThere')
Traceback (most recent call last): File "<stdin>", line 1, in ?
NameError: HiThere

>>> try: ...
        raise NameError('HiThere')
    ... except NameError: ...
        print 'An exception flew by!'

>>> try: ...
        raise Exception('spam', 'eggs') ...
    except Exception as inst:
      print type(inst) # the exception instance
      print inst.args # arguments stored in .args
```

# User Defined Excp

Ethans
Learn from experts

Programmers can also have their own exceptions by creating a new exception class Exceptions should typically be derived from the Exception class, either directly or indirectly.

For Example:

```
>>> class MyError(Exception):
        ... def __init__(self, value):
            ... self.value = value

>>> try:
    ... raise MyError(2*2)
    except MyError as e:
      print 'My exception occurred, value:', e.value ...
My exception occurred, value: 4
```





# OOPS Concept

Ethans
Learn from experts

→Computer languages are called as object oriented programming language when they follow the object oriented principles. These principles are the pillars of object oriented programming.

**Encapsulation:**

It can also be called as Data hiding principle, where user do not need to bother about how the things

implemented rather than knowing how to use them. Encapsulation is the packaging of data and functions

together. It also allow selective hiding of attributes in an object to protect the code from accidental corruption.

**Inheritance:**

It is the principle of code re-usability, when the object or class inherit/borrow the properties from other class to

minimize the implementation of similar property in a class.

**Polymorphism:**

This principle refers to a programming language's ability to process objects differently depending on their data type or class.

# OOPS Concept

Ethans
Learn from experts

**What is object?**

An object refers as an instance of the class, which is used to manipulate the attributes of the class.

An object can characterized as the real world object which has same characteristics, state and behavior. Cars have state (name, color, model) and behavior (turning, reverse moving, moving at particular speed). So any car model in the real world is the object of car class. As we can have many models of car, similar there can be many object of the car class.

**What is class?**

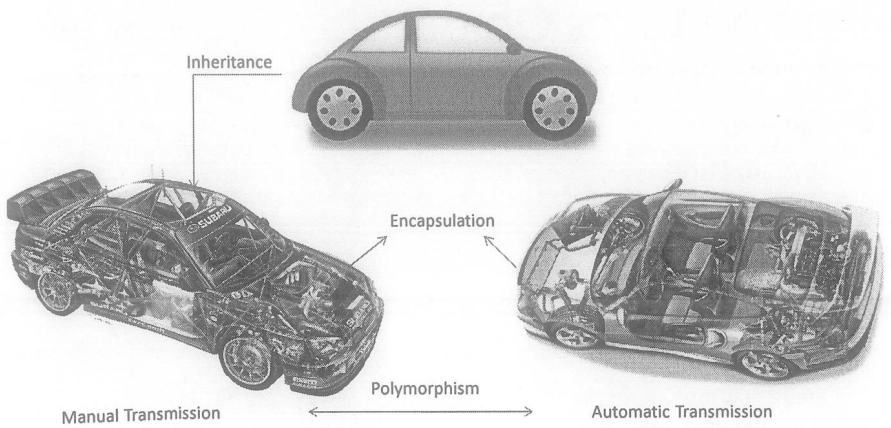
A class is the prototype or the blueprint from which objects are created. In the real world, you often find the many objects of the single class. Every class has the attributes and functions, like car class has the function of manual gear. Car will move back if the reverse gear start operating. These attributes are access with the object of the class.





# OOPS Concept

Ethans
Learn from experts



# Python Class

Ethans
Learn From experts

The simplest form of python class definition looks like:

**Syntax:**

class ClassName:
    <statement-1>
    . . . <statement-N>

**To Note:**

Similar as function definitions (def statements) class definition must be executed before they have any effect.





# Python Class Object

Ethans
Learn From experts

Can be used for:

Initialization

Attribute reference.

**Initialization:**

x = MyClass()

It creates a new *instance* of the class (or new object) and assigns this object to the local variable x. Though it created an empty variable.

**Attribute reference:**

x.f()
x.FunctionName()

# Class Example

Ethans
Learn From experts

**Creating Class**

```
class exampleClass:
... firstVar = 'This is the firstvar in the class'
... secondVar = 'This is the secondVar in the class'

    def classMethod(self)
    ...  print 'I am in the classMethod'
```

**Creating Object**

```
classObj = exampleClass()
classObj.firstVar
classObj. classMethod()
```





# Getter and Setter

Ethans
Learn From experts

**Creating Class**

```
class exampleClass:
... firstVar = 'This is the firstvar in the class'
    def setName(self, name)
    ...  self.name=name
    def getName(self):
    ...  self.name
```

**Creating Object**

```
classObj = exampleClass()
classObj.firstVar
classObj. setName('Ethans')
classObj. getName()
```

# Class Constructor

Ethans
Learn from experts

Constructor is the special method in the class which perform certain actions when you create an object.

**Creating Class with Constructor**

```
class exampleClass:
... firstVar = 'This is the firstvar in the class'
    def __init__(self, name)
    ... print "Let's say I am in initialization phase"
    ... self.name=name
```

**Creating Object**

```
classObj1 = exampleClass('Jatin')
classObj2 = exampleClass('Educum')

classObj1.name
```





# Class and instance variable

Ethans
Learn from experts

```
class Dog:

    kind = 'canine'         # class variable shared by all instances

    def __init__(self, name):
        self.name = name    # instance variable unique to each instance

>>> d = Dog('Fido')
>>> e = Dog('Buddy')
>>> d.kind                  # shared by all dogs
'canine'
>>> e.kind                  # shared by all dogs
'canine'
>>> d.name                  # unique to d
'Fido'
>>> e.name                  # unique to e
'Buddy'
```

Code Snippet:

Kind is class variable use by all object. But cannot be used inside the method.

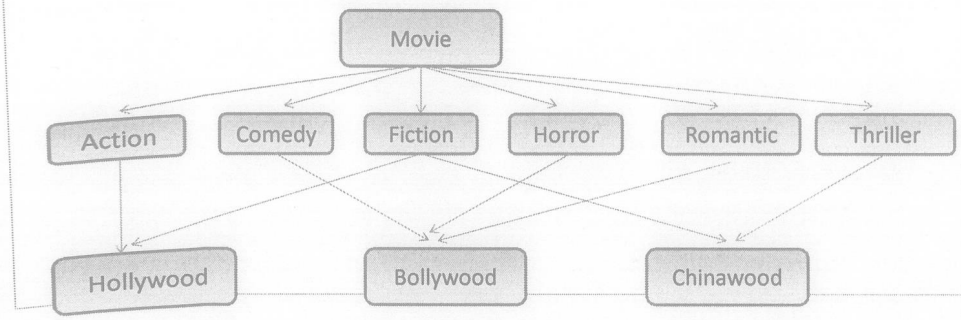
Name is instance variable.

Avoid class variables as much as you can.

## Inheritance

Ethan's Learn From experts

→One of the principle of objected oriented language is inheritance.
→The following diagram show how the inheritance works.




Slide 177

www.ethans.co.in

## Inheritance

Ethan's Learn From experts

**Syntax:**

```
class childClass(parentClass):
    <statement-1>
    . . . <statement-N>
```

Here child class is derived from the parent class. So all the parameters of parent class belongs to the child class. So if the child object try to access the parameter which is not available in child class it goes to parent class.

If the parameter is available at child class, child object access the child parameter. In this case parent object is overridden.

Slide 178

www.ethans.co.in

# Inheritance ex

Ethans
Learn from experts

```
class parentClass:
...    firstVar = 'This is my firstVar in the parent Class'
...    secondVar = 'This is my secondVar in the parent Class'

class childClass(parentClass):
...    pass

Pobj= parentClass()
Cobj= childClass()

Pobj.firstVar
Cobj.firstVar
```





# Inheritance with multiple classes ex

Ethans
Learn from experts

```
class parentClass1:
...    firstVar = 'This is my firstVar in the parent Class'
...    secondVar = 'This is my secondVar in the parent Class'

class parentClass2:
...    thirdVar = 'This is my thirdVar in the parent Class'
...    secondVar = 'This is my secondVar in the parent Class2'

class childClass(parentClass1, parentClass2):
...    pass

Cobj= childClass()
Cobj.firstVar
Cobj.secondVar
```

## New Style Classes

Ethans
Learn from experts

```
class parentClass1(object):
...     def func(self):
            print 'This is my func function in the parent Class1'

class parentClass2(object):
...     def func(self):
            print 'This is my func function in the parent Class2'

class childClass(parentClass1, parentClass2):
...         def func(self):
                super(childClass, self).func()

Cobj= childClass()
Cobj.func()
```





## Class Functions

Ethans
Learn from experts

- The getattr(obj, name[, default]) : to access the attribute of object
- The hasattr(obj,name) : to check if an attribute exists or not.
- The setattr(obj,name,value) : to set an attribute. If attribute does not exist, then it would be created.
- The delattr(obj, name) : to delete an attribute.

**For example:**

```
hasattr(emp1, 'age')    # Returns true if 'age' attribute exists
getattr(emp1, 'age')    # Returns value of 'age' attribute
setattr(emp1, 'age', 8) # Set attribute 'age' at 8
delattr(empl, 'age')    # Delete attribute 'age'
```

## Function Overloading

Ethans
Learn from experts

```
class overloading():

  def __init__(self, var1, var2):
    self.var1 = var1
    self.var2 = var2
    print "Value of var1 and var2 is %d and %d" %(var1, var2)


  def __str__(self):
    return 'Returning a string from it'

  def __len__(self):
    return self.var1

  def __add__(self, var):
    self.var3 = self.var1 + var
    return 'Sum of var1 and var2 is: %r' %self.var3

obj = overloading(10, 20)
print obj
print len(obj)
print obj + 20
```





## Function Overloading

Ethans
Learn from experts

```
class overloading():

  def __init__(self, var1, var2):
    self.var1 = var1
    self.var2 = var2
    print "Value of var1 and var2 is %d and %d" %(var1, var2)


  def __str__(self):
    return 'Returning a string from it'

  def __len__(self):
    return self.var1

  def __add__(self, var):
    self.var3 = self.var1 + var
    return 'Sum of var1 and var2 is: %r' %self.var3

obj = overloading(10, 20)
print obj
print len(obj)
print obj + 20
```

## Objective – Module 8

Ethans
Learn from experts

**Debugging, Framework & Regular expression**

- Debug Python programs using pdb debugger
- Pycharm Debugger
- Assert statement for debugging
- Testing with Python using UnitTest Framework
- What are regular expressions?
- The match and search Function
- Compile and matching
- Matching vs searching
- Search and Replace feature using RE
- Extended Regular Expressions
- Wildcard characters and work with them



## Python Debugger

Ethans
Learn from experts

- Pdb is the interactive source code debugger for Python
- It is by default available along with the Python interpreter.
- It is very common for IDLE or command line user who do not use any specific debugger tool and Python IDE.
- Pdb is the name of the module and pdb.set_trace is the name of the function which enable debugging mode.

```
import sys
import os
import pdb

pdb.set_trace()

def sumOfTwoNumbers(a, b):
    c = a + b
    return c

a = int(sys.argv[1])
b = int(sys.argv[2])

sum = sumOfTwoNumbers(a, b)
print sum
```

## Generic commands

Ethans
Learn from experts

| Debugger Commands | Description |
|---|---|
| h | Help |
| n | Execute the next statement |
| Enter | Repeating the last debugging command |
| q | Quitting debugger |
| p | Print the variables |
| c | Turning off the (pdb) prompt |
| l | Seeing where you are |
| w | Print a stack trace, with the most recent frame at the bottom |
| d | Move the current frame one level down in the stack trace |
| u | Move the current frame one level up in the stack trace |

Slide 187

www.ethans.co.in

## Generic commands

Ethans
Learn from experts

- n (Execute the next statement with 'n') - Lower case n key is used to execute the next statement (can be said line by line or complete block), Ultimately it will come to the end of the program and return you the normal prompt.
- Enter – Execute the last statement (Repeating the last command)
- q (Quit from the debugger) - Lower case q is used to abruptly quitting the debugger that's why you see an exception while execution of the program.
- p (Printing the value of the variable during debugging) - During the execution user can print the variable value, and NameError will be raised if the variable is not defined.
- c (continue the execution the program with c command )

Slide 188

www.ethans.co.in

## Generic commands

Ethans
Learn from experts

- l (List the location of the program) - "l" shows you, on the screen, the general area of your program's souce code that you are executing. By default, it lists 11 (eleven) lines of code
- s (step into functions) - With the help command you will step inside the function in python
- r (return from the function) - Same as c command but for functions
- b (set the breakpoint) - b number will set the breakpoint and b directly return all the breakpoint set in the program.

- Assign value to the variable
- !Var = 10





## PDB Demonstration

Ethans
Learn from experts

PDB Demonstration

## Pycharm

Ethans
Learn from experts

PyCharm is an Integrated Development Environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django. PyCharm is developed by the Czech company JetBrains.

It is cross-platform working on Windows, Mac OS X and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm Community Edition is less extensive than the Professional Edition.

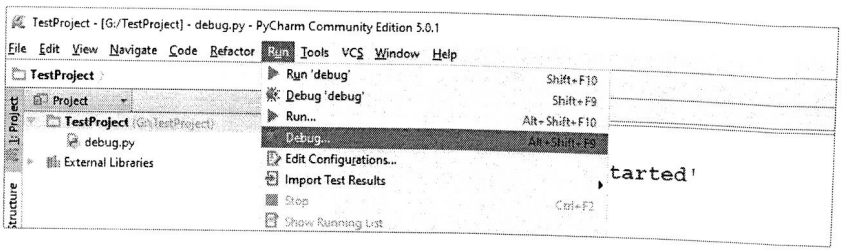
Source - Wikipedia





## Pycharm Debugger

Ethans
Learn from experts



## Pycharm Debugger Demonstration

Ethans
Learn from experts

Pycharm Debugger Demonstration





## UNIT Test Framework

Ethans
Learn from experts

- Unittest Framework is xUnit style framework for Python.
- The unittest module used to be called PyUnit.
- The framework implemented by unittest supports fixtures, test suites, and a test runner to enable automated testing for your code.

- The standard workflow is:
  1. You define your own class derived from unittest.TestCase.
  2. Then you fill it with functions that start with 'test_'.
  3. You run the tests by placing unittest.main() in your file, usually at the bottom.

## Automating Unit test case - 1

Ethans
Learn From experts

```
import unittest

def checkArguments(a,b):
    return True if a == b else False

class EqualityTestCases(unittest.TestCase):

    def test_check_if_4_equals_4(self):
        self.assertTrue(checkArguments(4,4))

if __name__ == '__main__':
    unittest.main()
```

Slide 195

www.ethans.co.in

## Automating Unit test case - 2

Ethans
Learn From experts

```
import unittest

def checkArguments(a,b):
    return True if a == b else False

class EqualityTestCases(unittest.TestCase):

    def setUp(self):
        print 'This method will call before every test case called!'

    def test_check_if_4_equals_4(self):
        self.assertTrue(checkArguments(4,4))

    def test_check_string_multiplication(self):
        self.assertEqual('a' * 3, 'aaa')

if __name__ == '__main__':
    unittest.main()
```

Slide 196

www.ethans.co.in

## Execute Test Cases

Ethan's
Learn From experts

```
C:\Users\jatin\Dropbox\Edureka GE Batch\Day6>python testcases2.py -v
test_check_if_4_equals_4 (__main__.EqualityTestCases) ... This method will ca
 before every test case called!
ok
test_check_string_multiplication (__main__.EqualityTestCases) ... This method
ill call before every test case called!
ok

----------------------------------------------------------------------
Ran 2 tests in 0.014s

OK
```





## Execute Specific Test case

Ethan's
Learn From experts

```
C:\Users\jatin\Dropbox\Edureka GE Batch\Day6>python
-m unittest testcases2.EqualityTestCases.test_check_
string_multiplication
This method will call before every test case called!

.
----------------------------------------------------------------------
Ran 1 test in 0.003s

OK
```

## UNIT Test Framework Demo

Ethans
Learn From experts

UNIT Test Framework Demo





## What is Regular Expression?

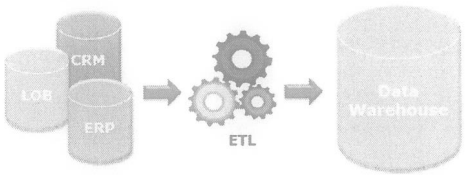
Ethans
Learn From experts

- Regular expression is a set of characters together form the search pattern.
- Main use of regular expression is to matches pattern in any string forms.
- The other use of regular expression to provide 'find and replace feature' in the programming languages
- Many language provide regular expression capabilities, some language have it inbuilt and other are having regular expression libraries.
- Regular expression is also known by regex or regexp.
- Regular expression forms the generic pattern for the string matching with the help of pre-defined wildcard characters.
- In Python, regular expression is supported by **re module,** which comes along with the installation of Python interpreter.

## Traditional Uses of Regular expression

Ethans
Learn from experts

- Online Ad Targeting
- Data filtrations
- Text Processing
- Big Data analysis
- Data science
- ETL processing
- Internet Search
- Web Scraping
- Data handling
- Data substitutions
- And much More...








## Match Function

Ethans
Learn from experts

- Match function is available in re module.
- It attempts to match an RE pattern to a string with optional flags at the beginning of string.
- The re.match function returns a match object on success, None object on failure.
- Help on match function:

```
>>> import re
>>> help(re.match)
```

| Parameter | Description |
|---|---|
| pattern | The regex/pattern which need to match |
| string | This is the string, where re need to be matches |
| flags | Modifiers or Flags to support more functionality to re |

## Match Function - Ex

Ethans
Learn from experts

```
import re

line = "this is the string contain sentence"
matchObj = re.match("This is the string Match", line, re.I)

if matchObj:
   print "\n matchObj.group() : \n", matchObj.group()
   print "\n matchObj.group(1) : \n", matchObj.group(1)
else:
   print "No match!!"
```





## Search Function

Ethans
Learn from experts

- It scan the complete string and for the first occurrence of a given pattern within a string
- The re.search function returns a match object on success, None on failure.

- Help on match function:

```
>>> import re
>>> help(re.match)
```

| Parameter | Description |
|---|---|
| pattern | The regex/pattern which need to match |
| string | This is the string, where re need to be matches |
| flags | Modifiers or Flags to support more functionality to re |