Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

Mục lục

Mục lục 4

I Lập trình 5

1 Giới thiệu 6

1.1 Các vấn đề lập trình 6

1.1.1 Introduction 6

1.1.2 Data Structure 7

1.1.3 OOP 7

1.1.4 Networking 7

1.1.5 Sample Project Ideas 7

1.2 How to ask a question 8

1.3 Các vấn đề lập trình 8

1.4 Các mô hình lập trình 8

1.5 Testing 10

1.6 Logging 12

1.7 Lập trình hàm 13

1.8 Lập trình song song 13

1.9 IDE 14

2 Python 16

2.1 Giới thiệu 16

2.2 Cài đặt 17

2.3 Cơ bản 18

2.4 Cú pháp cơ bản 18

2.5 Yield and Generators 20

2.6 Cấu trúc dữ liệu 24

2.6.1 Number 24

2.6.2 Collection 25

2.6.3 String 27

2.6.4 Datetime 28

2.6.5 Object 29

2.7 Object Oriented Programming 29

2.7.1 Metaclasses 31

2.7.2 Design Patterns 34

2.8 File System IO 35

2.9 Operating System 36

2.10 Networking . . . 36
2.11 Concurrency and Parallelism . . . 37
2.12 Event Based Programming . . . 39
2.13 Web Development . . . 40
2.14 Logging . . . 42
2.15 Configuration . . . 42
2.16 Command Line . . . 43
2.17 Testing . . . 43
2.18 IDE Debugging . . . 44
2.19 Package Manager . . . 46
2.20 Environment . . . 46
2.20.1 Create . . . 48
2.20.2 Clone . . . 48
2.20.3 List . . . 48
2.20.4 Remove . . . 48
2.20.5 Share . . . 48
2.21 Module . . . 49
2.22 Production . . . 50
2.23 Quản lý gói với Anaconda . . . 51
2.24 Test với python . . . 51
2.25 Xây dựng docs với readthedocs và sphinx . . . 52
2.26 Pycharm Pycharm . . . 55
2.27 Vì sao lại code python? . . . 55

3 C++ 56
3.1 Get Started . . . 56
3.2 Basic Syntax . . . 57
3.3 Cấu trúc dữ liệu . . . 57
3.4 Lập trình hướng đối tượng . . . 59
3.5 Cơ sở dữ liệu . . . 59
3.6 Testing . . . 60
3.7 IDE Debugging . . . 61

4 Javascript 62
4.1 Installation . . . 62
4.2 IDE . . . 62
4.3 Basic Syntax . . . 62
4.4 Data Structure . . . 64
4.4.1 Number . . . 64
4.4.2 String . . . 65
4.4.3 Collection . . . 67
4.4.4 Datetime . . . 67
4.4.5 Boolean . . . 67
4.4.6 Object . . . 67
4.5 OOP . . . 67
4.6 Networking . . . 69
4.7 Logging . . . 69
4.8 Documentation . . . 69
4.9 Error Handling . . . 70
4.10 Testing . . . 72

4.11 Package Manager . . . 72
4.12 Build Tool . . . 73
4.13 Make Module . . . 73

5 Java **74**
5.1 Get Started . . . 74
5.2 Basic Syntax . . . 74
5.3 Data Structure . . . 79
5.4 OOP . . . 79
5.4.1 Classes . . . 79
5.4.2 Encapsulation . . . 83
5.4.3 Inheritance . . . 83
5.4.4 Polymorphism . . . 85
5.4.5 Abstraction . . . 87
5.5 File System IO . . . 89
5.6 Error Handling . . . 93
5.7 Logging . . . 98
5.8 IDE . . . 99
5.9 Package Manager . . . 99
5.10 Build Tool . . . 100
5.11 Production . . . 100

6 PHP **101**

7 R **103**
7.1 R Courses . . . 103
7.2 Everything you need to know about R . . . 104

8 Scala **106**
8.1 Installation . . . 106
8.2 IDE . . . 106
8.3 Basic Syntax . . . 106

9 NodeJS **108**
9.1 Get Started . . . 108
9.2 Basic Syntax . . . 109
9.3 File System IO . . . 109
9.4 Package Manager . . . 114
9.5 Command Line . . . 115

10 Octave **117**
10.1 Matrix . . . 117

11 Toolbox **119**
11.1 Vim . . . 120
11.2 Virtual Box . . . 122
11.3 VMWare . . . 123

II Khoa học dữ liệu **126**

12 Học sâu **127**
12.1 Deep Feedforward Networks 127
12.2 Tài liệu Deep Learning 129
12.3 Các layer trong deep learning 129
12.3.1 Sparse Layers 129
12.3.2 Convolution Layers 129
12.4 Recurrent Neural Networks 130
12.4.1 What are RNNs? 130
12.4.2 What can RNNs do? 131
12.4.3 Training RNNs 132

13 Pytorch **134**
13.1 So sánh giữa Tensorflow và Pytorch? 134

14 Tensorflow **136**
14.1 Cài đặt Tensorflow trong Windows 136
14.2 Cài đặt tensorflow trong Ubuntu 16.04 137

Tài liệu **138**

Chỉ mục **139**

Ghi chú **140**

Phần I

Lập trình

Chương 1

Giới thiệu

1.1 Các vấn đề lập trình

I will to do crazy and dummy things, I will rewrite article for basic languages (which tutorialpoints do very goods)
Each language I will cover these concepts:
Table of content

```
code/
            1. introduction
            2. syntax
            3. data structure
            4. oop
            5. networking
            6. os
            7. parallel
            8. event based
            9. error handling
            10. logging
            11. configuration
            12. documentation
            13. test
            14. ui
            15. web
            16. database
            17. ide
            18. package manager
            19. build tool
            20. make module
            21. production (docker)
```

1.1.1 Introduction

Installation (environment, IDE)
Hello world
Courses

Resources

Syntax

variables and expressions
conditional
loops and Iteration
functions
define, use
parameters
scope of variables
anonymous functions
callbacks
self-invoking functions, inner functions
functions that return functions, functions that redefined themselves
closures
naming convention
comment convention

1.1.2 Data Structure

Number
String
Collection
DateTime
Boolean
Object

1.1.3 OOP

Classes Objects
Inheritance
Encapsulation
Abstraction
Polymorphism
For OOP Example: see Python: OOP

1.1.4 Networking

REST (example with chat app sender, receiver, message)

1.1.5 Sample Project Ideas

Guess My Number Game
Create Analog Clock
Create Pong Game
Create flappy bird

1.2 How to ask a question

Focus on questions about an actual problem you have faced. Include details about what you have tried and exactly what you are trying to do.
Ask about...
Specific programming problems
Software algorithms
Coding techniques
Software development tools
Not all questions work well in our format. Avoid questions that are primarily opinion-based, or that are likely to generate discussion rather than answers.
Don't ask about...
Questions you haven't tried to find an answer for (show your work!)
Product or service recommendations or comparisons
Requests for lists of things, polls, opinions, discussions, etc.
Anything not directly related to writing computer programs

1.3 Các vấn đề lập trình

Generic
KISS (Keep It Simple Stupid)
YAGNI
Do The Simplest Thing That Could Possibly Work
Keep Things DRY
Code For The Maintainer
Avoid Premature Optimization
Inter-Module/Class
Minimise Coupling
Law of Demeter
Composition Over Inheritance
Orthogonality
Module/Class
Maximise Cohesion
Liskov Substitution Principle
Open/Closed Principle
Single Responsibility Principle
Hide Implementation Details
Curly's Law
Software Quality Laws
First Law of Software Quality

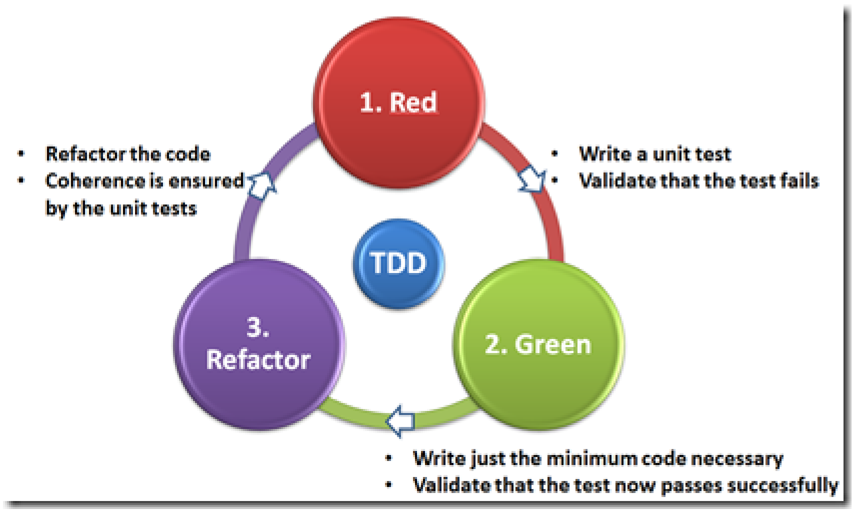
1.4 Các mô hình lập trình

Main paradigm approaches 1
1. Imperative
Description:
Computation as statements that directly change a program state (datafields)
Main Characteristics:
Direct assignments, common data structures, global variables

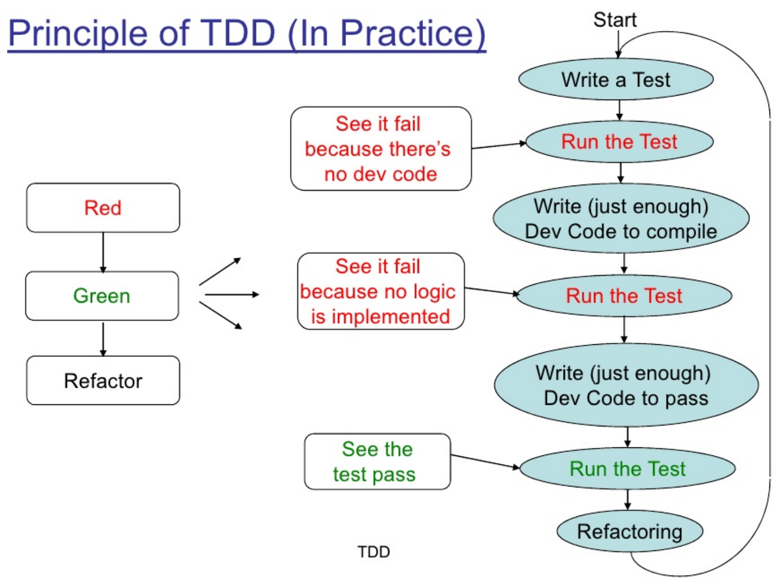
Critics: Edsger W. Dijkstra, Michael A. Jackson
Examples: Assembly, C, C++, Java, PHP, Python
2. Structured
Description:
A style of imperative programming with more logical program structure
Main Characteristics:
Structograms, indentation, either no, or limited use of, goto statements
Examples: C, C++, Java, Python
3. Procedural
Description:
Derived from structured programming, based on the concept of modular programming or the procedure call
Main Characteristics:
Local variables, sequence, selection, iteration, and modularization
Examples: C, C++, Lisp, PHP, Python
4. Functional
Description:
Treats computation as the evaluation of mathematical functions avoiding state and mutable data
Main Characteristics:
Lambda calculus, compositionality, formula, recursion, referential transparency, no side effects
Examples: Clojure, Coffeescript, Elixir, Erlang, F, Haskell, Lisp, Python, Scala, SequenceL, SML
5. Event-driven including time driven
Description:
Program flow is determined mainly by events, such as mouse clicks or interrupts including timer
Main Characteristics:
Main loop, event handlers, asynchronous processes
Examples: Javascript, ActionScript, Visual Basic
6. Object-oriented
Description:
Treats datafields as objects manipulated through pre-defined methods only
Main Characteristics:
Objects, methods, message passing, information hiding, data abstraction, encapsulation, polymorphism, inheritance, serialization-marshalling
Examples: Common Lisp, C++, C, Eiffel, Java, PHP, Python, Ruby, Scala
7. Declarative
Description:
Defines computation logic without defining its detailed control flow
Main Characteristics:
4GLs, spreadsheets, report program generators
Examples: SQL, regular expressions, CSS, Prolog
8. Automata-based programming
Description:
Treats programs as a model of a finite state machine or any other formal automata
Main Characteristics:

State enumeration, control variable, state changes, isomorphism, state transition table
Examples: AsmL

1.5 Testing

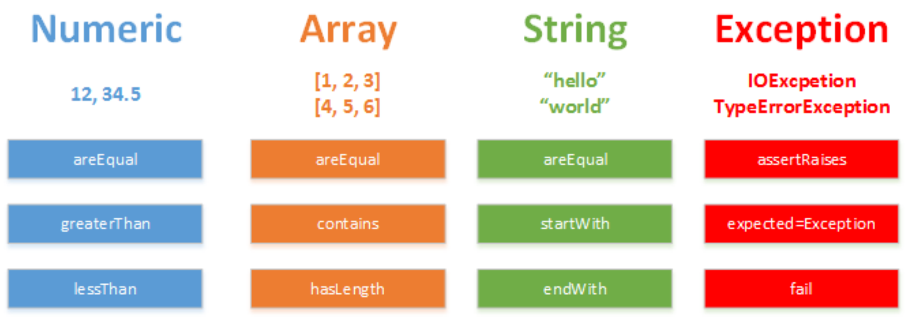

1. Red
2. Green
3. Refactor
TDD
• Write a unit test
• Validate that the test fails
• Write just the minimum code necessary
• Validate that the test now passes successfully
• Refactor the code
• Coherence is ensured by the unit tests

1. Definition 1 2
Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle:

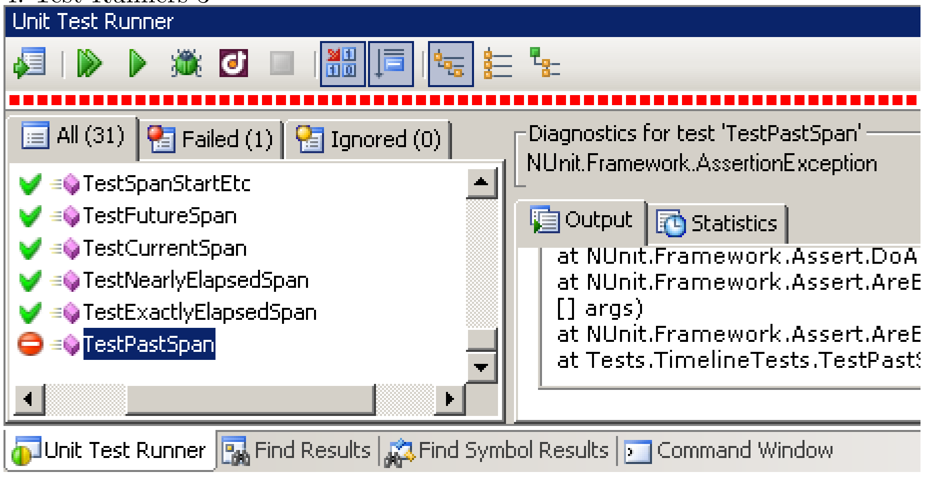

Principle of TDD (In Practice)
Start
Write a Test
Run the Test
Write (just enough) Dev Code to compile
Run the Test
Write (just enough) Dev Code to pass
Run the Test
Refactoring
See it fail because there's no dev code
See it fail because no logic is implemented
See the test pass
Red
Green
Refactor
TDD

Step 1: First the developer writes an (initially failing) automated test case that

defines a desired improvement or new function,
Step 2: Then produces the minimum amount of code to pass that test,
Step 3: Finally refactors the new code to acceptable standards.
Kent Beck, who is credited with having developed or 'rediscovered' the technique, stated in 2003 that TDD encourages simple designs and inspires confidence.
2. Principles 2
Kent Beck defines
Never with a single line of code unless you have a failing automated test. Eliminate duplication Red: (Automated test fail) Green (Automated test pass because dev code has been written) Refactor (Eliminate duplication, Clean the code)
3. Assertions Assert Framework



Assert that the expected results have occurred. [code lang="java"] @Test public void test() assertEquals(2, 1 + 1); [/code]
4. Test Runners 3



```
TestSpanStartEtc
TestFutureSpan
TestCurrentSpan
TestNearlyElapsedSpan
TestExactlyElapsedSpan
TestPastSpan
```

Diagnostics for test 'TestPastSpan'
NUnit.Framework.AssertionException

```
at NUnit.Framework.Assert.DoA
at NUnit.Framework.Assert.AreE
[] args)
at NUnit.Framework.Assert.AreE
at Tests.TimelineTests.TestPastS
```

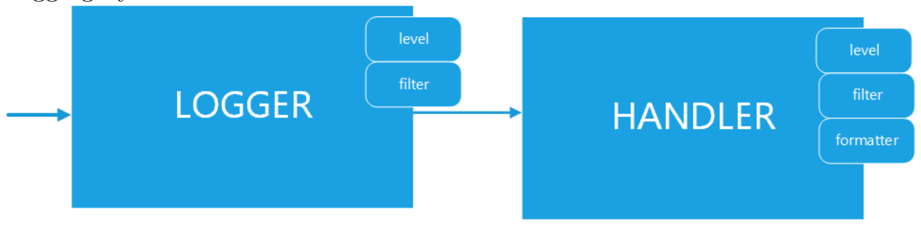
When testing a large real-world web app there may be tens or hundreds of test cases, and we certainly don't want to run each one manually. In such as scenario we need to use a test runner to find and execute the tests for us, and in this article we'll explore just that.
A test runner provides the a good basis for a real testing framework. A test runner is designed to run tests, tag tests with attributes (annotations), and provide reporting and other features.
In general you break your tests up into 3 standard sections; setUp(), tests, and tearDown(), typical for a test runner setup.

The setUp() and tearDown() methods are run automatically for every test, and contain respectively:
The setup steps you need to take before running the test, such as unlocking the screen and killing open apps. The cooldown steps you need to run after the test, such as closing the Marionette session.
5. Test Frameworks
Language Test Frameworks C++/VisualStudio C++: Test Web Service rest-assured Web UI SeleniumHQ

1.6 Logging

Logging is the process of recording application actions and state to a secondary interface.

Logging System



Levels
Level When it's used DEBUG Detailed information, typically of interest only when diagnosing problems. INFO Confirmation that things are working as expected. WARNING An indication that something unexpected happened, or indicative of some problem in the near future (e.g. ‘disk space low’). The software is still working as expected.
ERROR

Due to a more serious problem, the software has not been able to perform some function. CRITICAL A serious error, indicating that the program itself may be unable to continue running. Best Practices 2 4 5 Logging should always be considered when handling an exception but should never take the place of a real handler. Keep all logging code in your production code. Have an ability to enable more/less detailed logging in production, preferably per subsystem and without restarting your program. Make logs easy to parse by grep and by eye. Stick to several common fields at the beginning of each line. Identify time, severity, and subsystem in every line. Clearly formulate the message. Make every log message easy to map to its source code line. If an error happens, try to collect and log as much information as possible. It may take long but it's OK because normal processing has failed anyway. Not having to wait when the same condition happens in production with a debugger attached is priceless.

1.7 Lập trình hàm

Functional Without mutable variables, assignment, conditional
Advantages 1 Most functional languages provide a nice, protected environment, somewhat like JavaLanguage. It's good to be able to catch exceptions instead of having CoreDumps in stability-critical applications. FP encourages safe ways of programming. I've never seen an OffByOne mistake in a functional program, for example... I've seen one. Adding two lengths to get an index but one of them was zero-indexed. Easy to discover though. – AnonymousDonor Functional programs tend to be much more terse than their ImperativeLanguage counterparts. Often this leads to enhanced programmer productivity. FP encourages quick prototyping. As such, I think it is the best software design paradigm for ExtremeProgrammers... but what do I know. FP is modular in the dimension of functionality, where ObjectOrientedProgramming is modular in the dimension of different components. Generic routines (also provided by CeePlusPlus) with easy syntax. ParametricPolymorphism The ability to have your cake and eat it. Imagine you have a complex OO system processing messages - every component might make state changes depending on the message and then forward the message to some objects it has links to. Wouldn't it be just too cool to be able to easily roll back every change if some object deep in the call hierarchy decided the message is flawed? How about having a history of different states? Many housekeeping tasks made for you: deconstructing data structures (PatternMatching), storing variable bindings (LexicalScope with closures), strong typing (TypeInference), * GarbageCollection, storage allocation, whether to use boxed (pointer-to-value) or unboxed (value directly) representation... Safe multithreading! Immutable data structures are not subject to data race conditions, and consequently don't have to be protected by locks. If you are always allocating new objects, rather than destructively manipulating existing ones, the locking can be hidden in the allocation and GarbageCollection system.

1.8 Lập trình song song

Paralell/Concurrency Programming 1. Callback Pattern 2 Callback functions are derived from a programming paradigm known as functional programming.

At a fundamental level, functional programming specifies the use of functions as arguments. Functional programming was—and still is, though to a much lesser extent today—seen as an esoteric technique of specially trained, master programmers.
Fortunately, the techniques of functional programming have been elucidated so that mere mortals like you and me can understand and use them with ease. One of the chief techniques in functional programming happens to be callback functions. As you will read shortly, implementing callback functions is as easy as passing regular variables as arguments. This technique is so simple that I wonder why it is mostly covered in advanced JavaScript topics.
[code lang="javascript"] function getN() return 10;
var n = getN();
function getAsyncN(callback) setTimeout(function() callback(10); , 1000);
function afterGetAsyncN(result) var n = 10; console.log(n);
getAsyncN(afterGetAsyncN); [/code]
2. Promise Pattern 1 3 What is a promise? The core idea behind promises is that a promise represents the result of an asynchronous operation.
A promise is in one of three different states:
pending - The initial state of a promise. fulfilled - The state of a promise representing a successful operation. rejected - The state of a promise representing a failed operation. Once a promise is fulfilled or rejected, it is immutable (i.e. it can never change again).

```
function aPromise(message){
  return new Promise(function( fulfill ,  reject ){
    if (message == "success"){
       fulfill ("it  is  a  success  Promise");
    }  if (message == "fail"){
       reject ("it  is  a  fail  Promise");
    }
  });
}
```

Usage:

```
aPromise("success").then(function(successMessage){
  console.log(successMessage) }, function(failMessage){
  // it is a success Promise
  console.log(failMessage)
})

aPromise("fail").then(function(successMessage){
  console.log(successMessage) }, function(failMessage){
  console.log(failMessage)
}) // it is a fail Promise
```

1.9 IDE

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software devel-

opment. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.
1. Navigation
Word Navigation Line Navigation File Navigation
2. Editing
Auto Complete Code Complete Multicursor Template (Snippets)
3. Formatting
Debugging Custom Rendering for Object

Chương 2

Python

Hướng dẫn online tại http://magizbox.com/training/python/site/

2.1 Giới thiệu

'Python' is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.

The language provides constructs intended to enable clear programs on both a small and large scale.
Python Tutorial Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.
Python is Interpreted
Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
Python is Interactive
You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
Python is Object-Oriented
Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
Python is Beginner Friendly
Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.
Audience This tutorial is designed for software programmers who need to learn Python programming language from scratch.
Sách
Tập hợp các sách python

Khoá học
Tập hợp các khóa học python
Tham khảo
Top 10 Python Libraries Of 2015

2.2 Cài đặt

Get Started Welcome! This tutorial details how to get started with Python.
For Windows Anaconda 4.3.0 Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.
1. Download the installer 2. Optional: Verify data integrity with MD5 or SHA-256 3. Double-click the .exe file to install Anaconda and follow the instructions on the screen Python 3.6 version 64-BIT INSTALLER Python 2.7 version 64-BIT INSTALLER Step 2. Discover the Map
https://docs.python.org/2/library/index.html
For CentOS Developer tools The Development tools will allow you to build and compile software from source code. Tools for building RPMs are also included, as well as source code management tools like Git, SVN, and CVS.

```
yum groupinstall "Development tools"
yum install zlib-devel
yum install bzip2-devel
yum install openssl-devel
yum install ncurses-devel
yum install sqlite-devel
```

Python Anaconda Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

```
cd /opt
wget --no-check-certificate https://www.python.org/ftp/python/2.7.6/
    ↪ Python-2.7.6.tar.xz
tar xf Python-2.7.6.tar.xz
cd Python-2.7.6
./configure --prefix=/usr/local
make && make altinstall
## link
ln -s /usr/local/bin/python2.7 /usr/local/bin/python
# final check
which python
python -V
# install Anaconda
cd ~/Downloads
wget https://repo.continuum.io/archive/Anaconda-2.3.0-Linux-x86_64
    ↪ .sh
bash ~/Downloads/Anaconda-2.3.0-Linux-x86_64.sh
```

2.3 Cơ bản

2.4 Cú pháp cơ bản

Print, print

```
print "Hello World"
```

Conditional

```
if you_smart:
    print "learn python"
else:
    print "go away"
```

Loop
In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.
Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement
Python programming language provides following types of loops to handle looping requirements.
while loop Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. for loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. nested loops You can use one or more loop inside any another while, for or do..while loop. While Loop A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.
Syntax
The syntax of a while loop in Python programming language is

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

Example

```
count = 0
while count < 9:
   print 'The count is:', count
   count += 1
print "Good bye!"
```

For Loop
It 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)
```

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable iterating$_var.Next, thestatementsblockisexecuted.Eachiter$
Example

```
for i in range(10):
    print "hello", i

for letter in 'Python':
   print 'Current letter :', letter

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

print "Good bye!"
```

Yield and Generator
Yield is a keyword that is used like return, except the function will return a generator.

```
def createGenerator():
    yield 1
    yield 2
    yield 3
mygenerator = createGenerator() # create a generator
print(mygenerator) # mygenerator is an object!
# <generator object createGenerator at 0xb7555c34>
for i in mygenerator:
    print(i)
# 1
# 2
# 3
```

Visit Yield and Generator explained for more information
Functions
Variable-length arguments

```
def functionname([formal_args,] *var_args_tuple ):
   "function_docstring"
   function_suite
   return [expression]
```

Example

```
#!/usr/bin/python

# Function definition is here
def printinfo( arg1, *vartuple ):
   "This prints a variable passed arguments"
   print "Output is: "
   print arg1
   for var in vartuple:
      print var
```

```
    return;

# Now you can call printinfo function
printinfo( 10 )
printinfo( 70, 60, 50 )
```

Coding Convention Code layout Indentation: 4 spaces
Suggest Readings
"Python Functions". www.tutorialspoint.com "Python Loops". www.tutorialspoint.com
"What does the “yield” keyword do?". stackoverflow.com "Improve Your Python: 'yield' and Generators Explained". jeffknupp.com
Vấn đề với mảng

Random Sampling [1] - sinh ra một mảng ngẫu nhiên trong khoảng (0, 1), mảng ngẫu nhiên số nguyên trong khoảng (x, y), mảng ngẫu nhiên là permutation của số từ 1 đến n

2.5 Yield and Generators

Coroutines and Subroutines When we call a normal Python function, execution starts at function's first line and continues until a return statement, exception, or the end of the function (which is seen as an implicit return None) is encountered. Once a function returns control to its caller, that's it. Any work done by the function and stored in local variables is lost. A new call to the function creates everything from scratch.
This is all very standard when discussing functions (more generally referred to as subroutines) in computer programming. There are times, though, when it's beneficial to have the ability to create a "function" which, instead of simply returning a single value, is able to yield a series of values. To do so, such a function would need to be able to "save its work," so to speak.
I said, "yield a series of values" because our hypothetical function doesn't "return" in the normal sense. return implies that the function is returning control of execution to the point where the function was called. "Yield," however, implies that the transfer of control is temporary and voluntary, and our function expects to regain it in the future.
In Python, "functions" with these capabilities are called generators, and they're incredibly useful. generators (and the yield statement) were initially introduced to give programmers a more straightforward way to write code responsible for producing a series of values. Previously, creating something like a random number generator required a class or module that both generated values and kept track of state between calls. With the introduction of generators, this became much simpler.
To better understand the problem generators solve, let's take a look at an example. Throughout the example, keep in mind the core problem being solved: generating a series of values.
Note: Outside of Python, all but the simplest generators would be referred to as coroutines. I'll use the latter term later in the post. The important thing

[1] tham khảo [pytorch](http://pytorch.org/docs/master/torch.html?highlight=randntorch.randn), [numpy](https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.random.html))

to remember is, in Python, everything described here as a coroutine is still a generator. Python formally defines the term generator; coroutine is used in discussion but has no formal definition in the language.
Example: Fun With Prime Numbers Suppose our boss asks us to write a function that takes a list of ints and returns some Iterable containing the elements which are prime1 numbers.
Remember, an Iterable is just an object capable of returning its members one at a time.
"Simple," we say, and we write the following:

```
def get_primes(input_list):
    result_list = list()
    for element in input_list:
        if is_prime(element):
            result_list .append()

    return result_list
```

or better yet...

```
def get_primes(input_list):
    return (element for element in input_list if is_prime(element))

# not germane to the example, but here's a possible implementation of
# is_prime...

def is_prime(number):
    if number > 1:
        if number == 2:
            return True
        if number % 2 == 0:
            return False
        for current in range(3, int(math.sqrt(number) + 1), 2):
            if number % current == 0:
                return False
        return True
    return False
```

Either get$_{p}rimesimplementationabovefulfillstherequirements, sowetellourbosswe'redone.Shereportsour f$
Dealing With Infinite Sequences Well, not quite exactly. A few days later, our boss comes back and tells us she's run into a small problem: she wants to use our get$_{p}rimesfunctiononaverylargelistofnumbers.Infact, thelistissolargethatmerelycreatingitwouldconsume$
Once we think about this new requirement, it becomes clear that it requires more than a simple change to get$_{p}rimes.Clearly, wecan'treturnalistofalltheprimenumbersfromstarttoinfinity($
Before we give up, let's determine the core obstacle preventing us from writing a function that satisfies our boss's new requirements. Thinking about it, we arrive at the following: functions only get one chance to return results, and thus must return all results at once. It seems pointless to make such an obvious statement; "functions just work that way," we think. The real value lies in asking, "but what if they didn't?"
Imagine what we could do if get$_{p}rimescouldsimplyreturnthenextvalueinsteadofallthevaluesatonce.Itwould$

Unfortunately, this doesn't seem possible. Even if we had a magical function that allowed us to iterate from n to infinity, we'd get stuck after returning the first value:
def get$_{p}rimes(start) : forelementinmagical_{i}nfinite_{r}ange(start) : ifis_{p}rime(element) :$
$returnelementImagineget_{p}rimesiscalledlikeso :$
def solve$_{n}umber_{1}0() : She * is * workingonProjectEuler10, Iknewit!total =$
$2fornext_{p}rimeinget_{p}rimes(3) : ifnext_{p}rime < 2000000 : total+ = next_{p}rimeelse :$
$print(total)returnClearly, inget_{p}rimes, wewouldimmediatelyhitthecasewherenumber =$
$3andreturnatline4.Insteadofreturn, weneedawaytogenerateavalueand, whenaskedforthenextone, pickupw$
Functions, though, can't do this. When they return, they're done for good. Even if we could guarantee a function would be called again, we have no way of saying, "OK, now, instead of starting at the first line like we normally do, start up where we left off at line 4." Functions have a single entry point: the first line.
Enter the Generator This sort of problem is so common that a new construct was added to Python to solve it: the generator. A generator "generates" values. Creating generators was made as straightforward as possible through the concept of generator functions, introduced simultaneously.
A generator function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a generator function (even if it also contains a return statement). There's nothing else we need to do to create one.
generator functions create generator iterators. That's the last time you'll see the term generator iterator, though, since they're almost always referred to as "generators". Just remember that a generator is a special type of iterator. To be considered an iterator, generators must define a few methods, one of which is next(). To get the next value from a generator, we use the same built-in function as for iterators: next().
This point bears repeating: to get the next value from a generator, we use the same built-in function as for iterators: next().
(next() takes care of calling the generator's next() method). Since a generator is a type of iterator, it can be used in a for loop.
So whenever next() is called on a generator, the generator is responsible for passing back a value to whomever called next(). It does so by calling yield along with the value to be passed back (e.g. yield 7). The easiest way to remember what yield does is to think of it as return (plus a little magic) for generator functions.**
Again, this bears repeating: yield is just return (plus a little magic) for generator functions.
Here's a simple generator function:
»> def simple$_{g}enerator_{f}unction() :>>> yield1 >>> yield2 >>> yield3Andherearetwosimplewaystouseit$
»> for value in simple$_{g}enerator_{f}unction() :>>> print(value)123 >>> our_{g}enerator =$
$simple_{g}enerator_{f}unction() >>> next(our_{g}enerator)1 >>> next(our_{g}enerator)2 >>>$
$next(our_{g}enerator)3Magic?What'sthemagicpart?Gladyouasked!Whenageneratorfunctioncallsyield, the"$
Let's rewrite get$_{p}rimesasageneratorfunction.Noticethatwenolongerneedthemagical_{i}nfinite_{r}angefunction$
def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : yieldnumbernumber+ =$
$1Ifageneratorfunctioncallsreturnorreachestheenditsdefinition, aStopIterationexceptionisraised.Thissig$
$loopispresentinget_{p}rimes.Ifitweren't, thefirsttimenext()wascalledwewouldcheckifthenumberisprimeand$
»> our$_{g}enerator = simple_{g}enerator_{f}unction() >>> forvalueinour_{g}enerator :>>>$
$print(value)$

»> our$_{g}eneratorhasbeenexhausted... >>> print(next(our_{g}enerator))Traceback(mostrecentcalllast):$ $File" < ipython - input - 13 - 7e48a609051a > ", line1, in < module >$ $next(our_{g}enerator)StopIteration$
»> however, we can always create a new generator »> by calling the generator function again...
»> new$_{g}enerator = simple_{g}enerator_{f}unction() >>> print(next(new_{g}enerator))perfectlyvalid1Thus, theu$
Visualizing the flow Let's go back to the code that was calling get$_{p}rimes :$ $solve_{n}umber_{1}0.$
def solve$_{n}umber_{1}0() : She * is * workingonProjectEuler10, Iknewit!total =$ $2fornext_{p}rimeinget_{p}rimes(3) : ifnext_{p}rime < 2000000 : total+ = next_{p}rimeelse :$ $print(total)returnIt'shelpfultovisualizehowthefirstfewelementsarecreatedwhenwecallget_{p}rimesinsolve_{n}$
We enter the while loop on line 3 The if condition holds (3 is prime) We yield the value 3 and control to solve$_{n}umber_{1}0.Then, backinsolve_{n}umber_{1}0 :$
The value 3 is passed back to the for loop The for loop assigns next$_{p}rimetothisvaluenext_{p}rimeisaddedtototalT$
def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : yieldnumbernumber+ =$ $1 <<<<<<<<<< Mostimportantly, numberstillhasthesamevalueitdidwhenwecalledyield(i.e.3).Rememb$
Moar Power In PEP 342, support was added for passing values into generators. PEP 342 gave generators the power to yield a value (as before), receive a value, or both yield a value and receive a (possibly different) value in a single statement. To illustrate how values are sent to a generator, let's return to our prime number example. This time, instead of simply printing every prime number greater than number, we'll find the smallest prime number greater than successive powers of a number (i.e. for 10, we want the smallest prime greater than 10, then 100, then 1000, etc.). We start in the same way as get$_{p}rimes :$
def print$_{s}uccessive_{p}rimes(iterations, base = 10) : likenormalfunctions, ageneratorfunctioncanbeassigned$
prime$_{g}enerator = get_{p}rimes(base)missingcode...forpowerinrange(iterations) :$ $missingcode...$
def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : ...whatgoeshere?Thenextlineofget_{p}rimestakesa$ $yieldfoomeans, "yieldfooand, whenavalueissenttome, setothertothatvalue."Youcan"send"valuestoagenera$
def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : number = yieldnumbernumber+ =$ $1Inthisway, wecansetnumbertoadifferentvalueeachtimethegeneratoryields.Wecannowfillinthemissingc$
def print$_{s}uccessive_{p}rimes(iterations, base = 10) : prime_{g}enerator = get_{p}rimes(base)prime_{g}enerator.send($ $print(prime_{g}enerator.send(base**power))Twothingstonotehere : First, we'reprintingtheresultofgenerato$
Second, notice the prime$_{g}enerator.send(None)line.Whenyou'reusingsendto"start"agenerator(thatis, execu$
Round-up In the second half of this series, we'll discuss the various ways in which generators have been enhanced and the power they gained as a result. yield has become one of the most powerful keywords in Python. Now that we've built a solid understanding of how yield works, we have the knowledge necessary to understand some of the more "mind-bending" things that yield can be used for.
Believe it or not, we've barely scratched the surface of the power of yield. For example, while send does work as described above, it's almost never used when generating simple sequences like our example. Below, I've pasted a small demonstration of one common way send is used. I'll not say any more about it as figuring out how and why it works will be a good warm-up for part two.

```
import random

def get_data():
    """Return 3 random integers between 0 and 9"""
    return random.sample(range(10), 3)

def consume():
    """Displays a running average across lists of integers sent to it"""
    running_sum = 0
    data_items_seen = 0

    while True:
        data = yield
        data_items_seen += len(data)
        running_sum += sum(data)
        print('The running average is {}'.format(running_sum / float(
            ↪ data_items_seen)))

def produce(consumer):
    """Produces a set of values and forwards them to the pre-defined
        ↪ consumer
    function"""
    while True:
        data = get_data()
        print('Produced {}'.format(data))
        consumer.send(data)
        yield

if __name__ == '__main__':
    consumer = consume()
    consumer.send(None)
    producer = produce(consumer)

    for _ in range(10):
        print('Producing...')
        next(producer)
```

Remember... There are a few key ideas I hope you take away from this discussion: generators are used to generate a series of values yield is like the return of generator functions The only other thing yield does is save the "state" of a generator function A generator is just a special type of iterator Like iterators, we can get the next value from a generator using next() for gets values by calling next() implicitly

2.6 Cấu trúc dữ liệu

2.6.1 Number

Basic Operation

```
1
1.2
1 + 2
abs(-5)
```

2.6.2 Collection

In this post I will cover 4 most popular data types in python list, tuple, set, dictionary
List The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth.
The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.
Usage
A list keeps order, dict and set don't: when you care about order, therefore, you must use list (if your choice of containers is limited to these three, of course)
Most Popular Operations
Create a list a = ["a", "b", 3] Access values in list a[1] Updated List a[0] = 5 Delete list elements del a[1] Reverse a list a[::-1] Itertools [a + b for (a, b) in itertools.product(x, y)] Select random elements in list random.choice(x) random.sample(x, 3) Create a list a = [1, 2, 3] [1, 2, 3] Access values in list list1 = ['physics', 'chemistry', 1997, 2000] list2 = [1, 2, 3, 4, 5, 6, 7]
print list1[0] physics
print list2[1:5] [2, 3, 4, 5] Updated lists list = ['physics', 'chemistry', 1997, 2000]
print list[2] 1997
list[2] = 2001 print list[2] 2001 Delete list elements list1 = ['physics', 'chemistry', 1997, 2000];
print list1 ['physics', 'chemistry', 1997, 2000]
del list1[2]
print list1 ['physics', 'chemistry', 2000] Reverse a list [1, 3, 2][::-1] [2, 3, 1]
Itertools import itertools
x = [1, 2, 3] y = [2, 4, 5]
[a + b for (a, b) in itertools.product(x, y)] [3, 5, 6, 4, 6, 7, 5, 7, 8] Select random elements in list import random
x = [13, 23, 14, 52, 6, 23]
random.choice(x) 52
random.sample(x, 3) [23, 14, 52] Tuples A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
Usage
Tuples have structure, lists have order Tuples being immutable there is also a semantic distinction that should guide their usage. Tuples are heterogeneous data structures (i.e., their entries have different meanings), while lists are homogeneous sequences Most Popular Operations
Create a tuple t = ("a", 1, 2) Accessing Values in Tuples t[0], t[1:] Updating Tuples Not allowed Create a tuple tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5); tup3 = "a", "b", "c", "d"; tup4 = () tup5 = (50,)
Accessing Values in Tuples !/usr/bin/python
tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5, 6, 7);
tup1[0] physics
tup2[1:5] [2, 3, 4, 5] Updating Tuples Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take por-

tions of existing tuples to create new tuples as the following example demonstrates
tup1 = (12, 34.56); tup2 = ('abc', 'xyz');
Following action is not valid for tuples tup1[0] = 100;
So let's create a new tuple as follows tup3 = tup1 + tup2; print tup3 Set Sets are lists with no duplicate entries.
The sets module provides classes for constructing and manipulating unordered collections of unique elements. Common uses include membership testing, removing duplicates from a sequence, and computing standard math operations on sets such as intersection, union, difference, and symmetric difference.
Usage
set forbids duplicates, list does not: also a crucial distinction. Most Popular Operations
Create a set x = set(["Postcard", "Radio", "Telegram"]) Add elements to a set x.add("Mobile") Remove elements to a set x.remove("Radio") Subset y.issubset(x) Intersection x.intersection(y) Difference between two sets x.difference(y) Create a set x = set(["Postcard", "Radio", "Telegram"]) x set(['Postcard', 'Telegram', 'Radio']) Add elements to a set x = set(["Postcard", "Radio", "Telegram"]) x.add("Mobile") x set(['Postcard', 'Telegram', 'Mobile', 'Radio']) Remove elements to a set x = set(["Postcard", "Radio", "Telegram"]) x.remove("Radio") x set(['Postcard', 'Telegram']) Subset x = set(["a","b","c","d"]) y = set(["c","d"]) y.issubset(x) True Intersection x = set(["a","b","c","d"]) y = set(["c","d"]) x.intersection(y) set(['c', 'd']) Difference between two sets x = set(["Postcard", "Radio", "Telegram"]) y = set(["Radio","Television"]) x.difference(y) set(['Postcard', 'Telegram']) Dictionary 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. An empty dictionary without any items is written with just two curly braces, like this: .
Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.
Usage
dict associates with each key a value, while list and set just contain values: very different use cases, obviously. Most Popular Operations
Create a dictionary d = "a": 1, "b": 2, "c": 3 Update dictionary d["a"] = 4 Delete dictionary elements del d["a"] Create a dictionary dict = 'Name': 'Zara', 'Age': 7, 'Class': 'First'
print "dict['Name']: ", dict['Name'] print "dict['Age']: ", dict['Age'] Update dictionary dict = 'Name': 'Zara', 'Age': 7, 'Class': 'First'
dict['Age'] = 8; update existing entry dict['School'] = "DPS School"; Add new entry
print "dict['Age']: ", dict['Age'] print "dict['School']: ", dict['School'] Delete dictionary elements dict = 'Name': 'Zara', 'Age': 7, 'Class': 'First'
del dict['Name']; remove entry with key 'Name' dict.clear(); remove all entries in dict del dict ; delete entire dictionary
print "dict['Age']: ", dict['Age'] print "dict['School']: ", dict['School'] Related Readings Python Lists, tutorialspoint.com Python Dictionary, tutorialspoint.com Python Dictionary Methods, guru99 In Python, when to use a Dictionary, List or Set?, stackoverflow What's the difference between lists and tuples?, stackoverflow

2.6.3 String

Format '0, 1, 2'.format('a', 'b', 'c') 'a, b, c' Regular Expressions The aim of this chapter of our Python tutorial is to present a detailed led and descriptive introduction into regular expressions. This introduction will explain the theoretical aspects of regular expressions and will show you how to use them in Python scripts.
Regular Expressions are used in programming languages to filter texts or textstrings.
It's possible to check, if a text or a string matches a regular expression.
There is an aspect of regular expressions which shouldn't go unmentioned: The syntax of regular expressions is the same for all programming and script languages, e.g. Python, Perl, Java, SED, AWK and even X.
Functions match function This function attempts to match RE pattern to string with optional flags.
re.match(pattern, string, flags=0) Example
import re
line = "Cats are smarter than dogs"
matched$_object = re.match(r'(.*)are(.*?).*', line, re.M|re.I)$
if matched$_object : print"matched_object.group() : ", matched_object.group()print"matched_object.group(1) :$
$", matched_object.group(1)print"matched_object.group(2) : ", matched_object.group(2)else :$
$print"Nomatch!!"Whenthecodeisexecuted, itproducesfollowingresults$
matched$_object.group() : Catsaresmarterthandogsmatched_object.group(1) : Catsmatched_object.group(2) :$
$smartersearchfunctionThisfunctionsearchesforfirstoccurrenceofREpatternwithinstirngwithoptionalf$
re.search(pattern, string, flags=0) Example
!/usr/bin/python import re
line = "Cats are smarter than dogs"
search$_object = re.search(r'dogs', line, re.M|re.I)ifsearch_object : print"search-$
$->search_object.group() : ", search_object.group()else : print"Nothingfound!!"Whenthecodeisexecuted, it$
search -> search$_object.group() : dogssubfunctionThismethodreplacesalloccurrencesoftheREpatterninstri$
re.sub(pattern, repl, string, max=0) Example
!/usr/bin/python import re
phone = "2004-959-559 This is Phone Number"
Delete Python-style comments num = re.sub(r'.*$', "", phone)print"PhoneNum :$
$", num$
Remove anything other than digits num = re.sub(r", "", phone) print "Phone Num : ", num When the code is executed, it produces following results
Phone Num : 2004-959-559 Phone Num : 2004959559 Tokens Cheatsheet Character Classes . any character except newline /go.gle/ google goggle gogle word, digit, whitespace // AaYyz09 ?! // 012345 aZ? // 0123456789 abcd?/ §not word, digit, whitespace // abcded 1234 ?> // abc 12345 ?<. /§/ abc 123? <. [abc] any of a, b or c /analy[sz]e/ analyse analyze analyxe $[^abc]nota, borc/analy[^sz]e/analyseanalyzeanalyxe[a-$
$g]characterbetweenag/[2-4]/demo1demo2demo3demo4demo5QuantifiersAlternationa*$
$a+a?0ormore, 1ormore, 0or1/go*gle/goglegoglegooglegoooooglehgle/go+gle/gglegoglegooglegoooooglehgle/$
start / end of the string /$^abc/$ abc /$^abc/abcabc/abc/$ abc abc word, not-word boundary // This island is beautiful. // cat certificate Escaped characters
escaped special characters // username@exampe.com 300.000 USD // abc@/
/ abc@ tab, linefeed, carriage return // abc def /ab/ ab // abc@00A9 unicode escaped © /00A9/ Copyright©2017 - All rights reserved Groups and Lockaround
(abc) capture group /(demo|example)[0-9]/ demo1example4demo backreference

to group 1 /(abc|def)=/ abc=abc def=defabc=def (?:abc) non-capturing group /(?:abc)3/ abcabcabc abcabc (?=abc) positive lookahead /t(?=s)/ tttsssttss (?!abc) negative lookahead /t(?!s)/ tttsssttss (?<=abc) positive lookbehind /(?<=foo)bar/ foobar fuubar (?<!abc) negative lookbehind /(?<!foo)bar/ foobar fuubar Related Readings
Online regex tester and debugger: PHP, PCRE, Python, Golang and JavaScript, regex101.com RegExr: Learn, Build, Test RegEx, regexr.com

2.6.4 Datetime

Print current time
from datetime import datetime datetime.now().strftime(' '2015-12-29 14:02:27'
Get current time
import datetime datetime.datetime.now() datetime(2009, 1, 6, 15, 8, 24, 78915)
Unixtime
import time int(time.time()) Measure time elapsed
import time
start = time.time() print("hello") end = time.time() print(end - start) Moment
Dealing with dates in Python shouldn't have to suck.
Installation
pip install moment Usage
import moment from datetime import datetime
Create a moment from a string moment.date("12-18-2012")
Create a moment with a specified strftime format moment.date("12-18-2012", "
Moment uses the awesome dateparser library behind the scenes moment.date("2012-12-18")
Create a moment with words in it moment.date("December 18, 2012")
Create a moment that would normally be pretty hard to do moment.date("2 weeks ago")
Create a future moment that would otherwise be really difficult moment.date("2 weeks from now")
Create a moment from the current datetime moment.now()
The moment can also be UTC-based moment.utcnow()
Create a moment with the UTC time zone moment.utc("2012-12-18")
Create a moment from a Unix timestamp moment.unix(1355875153626)
Create a moment from a Unix UTC timestamp moment.unix(1355875153626, utc=True)
Return a datetime instance moment.date(2012, 12, 18).date
We can do the same thing with the UTC method moment.utc(2012, 12, 18).date
Create and format a moment using Moment.js semantics moment.now().format("YYYY-M-D")
Create and format a moment with strftime semantics moment.date(2012, 12, 18).strftime("
Update your moment's time zone moment.date(datetime(2012, 12, 18)).locale("US/Central").date
Alter the moment's UTC time zone to a different time zone moment.utcnow().timezone("US/Eastern").date
Set and update your moment's time zone. For instance, I'm on the west coast, but want NYC's current time. moment.now().locale("US/Pacific").timezone("US/Eastern")
In order to manipulate time zones, a locale must always be set or you must be using UTC. moment.utcnow().timezone("US/Eastern").date

You can also clone a moment, so the original stays unaltered now = moment.utcnow().timezone("US/Pacific") future = now.clone().add(weeks=2) Related Readings How to get current time in Python, stackoverflow Does Python's time.time() return the local or UTC timestamp?, stackoverflow Measure time elapsed in Python?, stackoverflow momnet, https://github.com/zachwill/moment

2.6.5 Object

Convert dict to object Elegant way to convert a normal Python dict with some nested dicts to an object
class Struct: def $_{init_{(self,**entries):self.}{}_{dict_{.update(entries)Then,youcanuse}}}$
> args = 'a': 1, 'b': 2 > s = Struct(**args) > s < $_{main_{.Structinstanceat0x01D6A738>>s.a1>s.b2RelatedReadings}}$
stackoverflow, Convert Python dict to object?

2.7 Object Oriented Programming

Object Oriented Programming Python has been an object-oriented language since it existed. Because of this, creating and using classes and objects are downright easy. This chapter helps you become an expert in using Python's object-oriented programming support.
If you do not have any previous experience with object-oriented (OO) programming, you may want to consult an introductory course on it or at least a tutorial of some sort so that you have a grasp of the basic concepts.
Classes and Objects Classes can be thought of as blueprints for creating objects. When I define a BankAccount class using the class keyword, I haven't actually created a bank account. Instead, what I've created is a sort of instruction manual for constructing "bank account" objects. Let's look at the following example code:
class BankAccount: id = None balance = 0
def $_{init_{(self,id,balance=0):self.id=idself.balance=balance}}$
def $_{get_balance(self):returnself.balance}$
def withdraw(self, amount): self.balance = self.balance - amount
def deposit(self, amount): self.balance = self.balance + amount
john = BankAccount(1, 1000.0) john.withdraw(100.0) The class BankAccount line does not create a new bank account. That is, just because we've defined a BankAcount doesn't mean we've created on; we've merely outlined the blueprint to create a BankAccount object. To do so, we call the class's
$^{init}{}_{methodwiththepropernumberofarguments(minusself,whichwe'llgettoinamoment)}$
So, to use the "blueprint" that we crated by defining the class BankAccount (which is used to create BankAccount objects), we call the class name almost as if it were a function: john = BankAccount(1, 1000.0). This line simple say "use the BankAccount blueprint to create me a new object, which I'll refer to as john".
The john object, known as an instance, is the realized version of the BankAccount class. Before we called BankAccount(), no BankAccount object existed. We can, of course, create as many BankAccount objects as we'd like. There is still, however, only one BankAccount class, regardless of how many instances of the class we create.

self So what's with that self parameter to all of the BankAccount methods? What is it? Why, it's the instance, of course! Put another way, a method like withdraw defines the instructions for withdrawing money from some abstract customer's account. Calling john.withdraw(100) puts those instructions to use on the john instance.
So when we say def withdraw(self, amount):, we're saying, "here's how you withdraw money from a BankAccount object (which we'll call self) and a dollar figure (which we'll call amount). self is the instance of the BankAccount that withdraw is being called on. That's not me making analogies, either. john.withdraw(100.0) is just shorthand for BankAccount.withdraw(john, 100.0), which is perfectly valid (if not often seen) code.
Constructors: $_{init_{selfmaymakesenseforothermethods,butwhatabout}{}_{init}?Whenwecall_{init},we'reintheprocessofcreatinganobject,sohowcanth}$
This is why when we call $_{init},weinitializeobjectsbysayingthingslikeself.id=id.Remember,sinceselfistheinstance,thisisequivalenttosayin$
Be careful what you $_{init_{After}{}_{init_{h}}asfinished,thecallercanrightlyassumethattheobjectisreadytouse.Thatis,afterjohn=BankAccount(1,10}$
Inheritance While Object-oriented Programming is useful as a modeling tool, it truly gains power when the concept of inheritance is introduced. Inheritance is the process by which a "child" class derives the data and behavior of a "parent" class. An example will definitely help us here.
Imagine we run a car dealership. We sell all types of vehicles, from motorcycles to trucks. We set ourselves apart from the competition by our prices. Specifically, how we determine the price of a vehicle on our lot: $5,000 x number of wheels a vehicle has. We love buying back our vehicles as well. We offer a flat rate - 10% of the miles driven on the vehicle. For trucks, that rate is $10,000. For cars, $8,000. For motorcycles, $4,000.
If we wanted to create a sales system for our dealership using Object-oriented techniques, how would we do so? What would the objects be? We might have a Sale class, a Customer class, an Inventory class, and so forth, but we'd almost certainly have a Car, Truck, and Motorcycle class.
What would these classes look like? Using what we've learned, here's a possible implementation of the Car class:
class Car(object): def $_{init_{(self,wheels,miles,make,model,year,sold_on):self.wheels=wheelsself.miles=milesself.make=makeself.model=m}}$
def sale$_price(self) : ifself.sold_onisnotNone : return0.0Alreadysoldreturn5000.0*$
$self.wheels$
def purchase$_price(self) : ifself.sold_onisNone : return0.0Notyetsoldreturn8000-$
$(.10*self.miles)OK, thatlooksprettyreasonable.Ofcourse, wewouldlikelyhaveanumberofothermethodsonth$
$sale_priceandpurchase_price.We'llseewhytheseareimportantinabit.$
Now that we've got the Car class, perhaps we should create a Truck class? Let's follow the same pattern we did for car:
class Truck(object): def $_{init_{(self,wheels,miles,make,model,year,sold_on):self.wheels=wheelsself.miles=milesself.make=makeself.model}}$
def sale$_price(self) : ifself.sold_onisnotNone : return0.0Alreadysoldreturn5000.0*$
$self.wheels$
def purchase$_price(self) : ifself.sold_onisNone : return0.0Notyetsoldreturn10000-$
$(.10*self.miles)Wow.That'salmostidenticaltothecarclass.Oneofthemostimportantrulesofprogramming(i$
So what gives? Where did we go wrong? Our main problem is that we raced straight to the concrete: Car and Truck are real things, tangible objects that make intuitive sense as classes. However, they share so much data and functionality in common that it seems there must be an abstraction we can introduce here. Indeed there is: the notion of Vehicle.

Abstract Classes A Vehicle is not a real-world object. Rather, it is a concept that some real-world objects (like cars, trucks, and motorcycles) embody. We would like to use the fact that each of these objects can be considered a vehicle to remove repeated code. We can do that by creating a Vehicle class:
class Vehicle(object): base$_{s}ale_{p}rice = 0$
def $_{i}nit_{(}self,wheels,miles,make,model,year,sold_{o}n):self.wheels=wheelsself.miles=milesself.make=makeself.model=modelself.year=yearself.$
def sale$_{p}rice(self) : ifself.sold_{o}nisnotNone : return0.0Alreadysoldreturn5000.0*$
$self.wheels$
def purchase$_{p}rice(self) : ifself.sold_{o}nisNone : return0.0Notyetsoldreturnself.base_{s}ale_{p}rice-$
$(.10*self.miles)NowwecanmaketheCarandTruckclassinheritfromtheVehicleclassbyreplacingobjectinthel$
We can now define Car and Truck in a very straightforward way:
class Car(Vehicle):
def $_{i}nit_{(}self,wheels,miles,make,model,year,sold_{o}n):self.wheels=wheelsself.miles=milesself.make=makeself.model=modelself.year=yearself.$
class Truck(Vehicle):
def $_{i}nit_{(}self,wheels,miles,make,model,year,sold_{o}n):self.wheels=wheelsself.miles=milesself.make=makeself.model=modelself.year=yearself.$
class Struct: def $_{i}nit_{(}self,**entries):self._{d}ict_{.}update(entries)Then,youcanuse$
$>$ args = 'a': 1, 'b': 2 $>$ s = Struct(**args) $>$ s $<$ $_{m}ain_{.}Structinstanceat0x01D6A738>>s.a1>s.b2SuggestedReadingsImproveYour$

2.7.1 Metaclasses

Metaclasses Python, Classes, and Objects Most readers are aware that Python is an object-oriented language. By object-oriented, we mean that Python can define classes, which bundle data and functionality into one entity. For example, we may create a class IntContainer which stores an integer and allows certain operations to be performed:
class IntContainer(object): def $_{i}nit_{(}self,i):self.i=int(i)$
def add$_{o}ne(self) : self.i+ = 1ic = IntContainer(2)ic.add_{o}ne()print(ic.i)3Thisisabitofasillyexample, butsh$
$theirabilitytobundledataandoperationsintoasingleobject, whichleadstocleaner, moremanageable, andmorea$
$orientedapproachtoprogrammingcanbeveryintuitiveandpowerful.$
What many do not realize, though, is that quite literally everything in the Python language is an object.
For example, integers are simply instances of the built-in int type:
print type(1) <type 'int'> To emphasize that the int type really is an object, let's derive from it and specialize the $_{a}dd_{m}ethod(whichisthemachineryunderneaththe+operator):$
(Note: We'll used the super syntax to call methods from the parent class: if you're unfamiliar with this, take a look at this StackOverflow question).
class MyInt(int): def $_{a}dd_{(}self,other):print"specializingaddition"returnsuper(MyInt,self)._{a}dd_{(}other)$
i = MyInt(2) print(i + 2) specializing addition 4 Using the + operator on our derived type goes through our $_{a}dd_{m}ethod,asexpected.Weseethatintreallyisanobjectthatcanbesubclassedandextendedjustlikeuser-definedc$
Down the Rabbit Hole: Classes as Objects We said above that everything in python is an object: it turns out that this is true of classes themselves. Let's look at an example.
We'll start by defining a class that does nothing
class DoNothing(object): pass If we instantiate this, we can use the type operator to see the type of object that it is:
d = DoNothing() type(d) $_{m}ain_{.}DoNothingWeseethatourvariabledisaninstanceoftheclass_{m}ain_{.}DoNothing.$
We can do this similarly for built-in types:
L = [1, 2, 3] type(L) list A list is, as you may expect, an object of type list.

But let's take this a step further: what is the type of DoNothing itself?
type(DoNothing) type The type of DoNothing is type. This tells us that the class DoNothing is itself an object, and that object is of type type.
It turns out that this is the same for built-in datatypes:
type(tuple), type(list), type(int), type(float) (type, type, type, type) What this shows is that in Python, classes are objects, and they are objects of type type. type is a metaclass: a class which instantiates classes. All new-style classes in Python are instances of the type metaclass, including type itself:
type(type) type Yes, you read that correctly: the type of type is type. In other words, type is an instance of itself. This sort of circularity cannot (to my knowledge) be duplicated in pure Python, and the behavior is created through a bit of a hack at the implementation level of Python.
Metaprogramming: Creating Classes on the Fly Now that we've stepped back and considered the fact that classes in Python are simply objects like everything else, we can think about what is known as metaprogramming. You're probably used to creating functions which return objects. We can think of these functions as an object factory: they take some arguments, create an object, and return it.
Here is a simple example of a function which creates an int object:
def int$_{f}actory(s) : i = int(s)returni$
i = int$_{f}actory('100')print(i)100Thisisoverly-simplistic, butanyfunctionyouwriteinthecourseofanormalpr$
$takesomearguments, dosomeoperations, andcreatereturnanobject.Withtheabovediscussioninmind, though,$
$-thisisametafunction :$
def class$_{f}actory() : classFoo(object) : passreturnFoo$
F = class$_{f}actory()f = F()print(type(f)) < class'_{main_{.Foo'>Justasthefunctionint_{f}actoryconstructsanreturnsaninstanceofin}}$
But the above construction is a bit awkward: especially if we were going to do some more complicated logic when constructing Foo, it would be nice to avoid all the nested indentations and define the class in a more dynamic way. We can accomplish this by instantiating Foo from type directly:
def class$_{f}actory() : returntype('Foo', (),)$
F = class$_{f}actory()f = F()print(type(f)) < class'_{main_{.Foo'>Infact,theconstruct}}$
class MyClass(object): pass is identical to the construct
MyClass = type('MyClass', (),) MyClass is an instance of type type, and that can be seen explicitly in the second version of the definition. A potential confusion arises from the more common use of type as a function to determine the type of an object, but you should strive to separate these two uses of the keyword in your mind: here type is a class (more precisely, a metaclass), and MyClass is an instance of type.
The arguments to the type constructor are: type(name, bases, dct) - name is a string giving the name of the class to be constructed - bases is a tuple giving the parent classes of the class to be constructed - dct is a dictionary of the attributes and methods of the class to be constructed
So, for example, the following two pieces of code have identical results:
class Foo(object): i = 4
class Bar(Foo): def get$_i(self) : returnself.i$
b = Bar() print(b.get$_i())4Foo = type('Foo', (), dict(i = 4))$
Bar = type('Bar', (Foo,), dict(get$_i = lambdaself : self.i))$
b = Bar() print(b.get$_i())4Thisperhapsseemsabitover-complicatedinthecaseofthiscontrivedexample, butitc$
$the - fly.$

Making Things Interesting: Custom Metaclasses Now things get really fun. Just as we can inherit from and extend a class we've created, we can also inherit from and extend the type metaclass, and create custom behavior in our metaclass.
Example 1: Modifying Attributes Let's use a simple example where we want to create an API in which the user can create a set of interfaces which contain a file object. Each interface should have a unique string ID, and contain an open file object. The user could then write specialized methods to accomplish certain tasks. There are certainly good ways to do this without delving into metaclasses, but such a simple example will (hopefully) elucidate what's going on.
First we'll create our interface meta class, deriving from type:
class InterfaceMeta(type): def $_{new}{}_{(cls,name,parents,dct):createaclass_idifit'snotspecifiedif'class_id'notindct:dct['class_id']=name.lowe}$
open the specified file for writing if 'file' in dct: filename = dct['file'] dct['file'] = open(filename, 'w')
we need to call type.$_{new}{}_{tocompletetheinitializationreturnsuper(InterfaceMeta,cls).}{}_{new}{}_{(cls,name,parents,dct)Noticethatwe'vemodified}$
Now we'll use our InterfaceMeta class to construct and instantiate an Interface object:
Interface = InterfaceMeta('Interface', (), dict(file='tmp.txt'))
print(Interface.class$_i d)print(Interface.file)interface < openfile'tmp.txt', mode'w'at0x21b8810 >$
$Thisbehavesaswe'dexpect : theclass_idclassvariableiscreated, andthefileclassvariableisreplacedwithanopen$
class Interface(object): $_{m}etaclass_{=InterfaceMetafile='tmp.txt'}$
print(Interface.class$_i d)print(Interface.file)interface < openfile'tmp.txt', mode'w'at0x21b8ae0 >$
$bydefiningthe_{m}etaclass_{attributeoftheclass,we'vetoldtheclassthatitshouldbeconstructedusingInterfaceMetaratherthanusingtype.Tomake}$
type(Interface) $_{m}ain_{.InterfaceMetaFurthermore,anyclassderivedfromInterfacewillnowbeconstructedusingthesamemetaclass:}$
class UserInterface(Interface): file = 'foo.txt'
print(UserInterface.file) print(UserInterface.class$_i d) < openfile'foo.txt', mode'w'at0x21b8c00 >$
$userinterfaceThissimpleexampleshowshowmetaclassescanbeusedtocreatepowerfulandflexibleAPIsforpr$
Example 2: Registering Subclasses Another possible use of a metaclass is to automatically register all subclasses derived from a given base class. For example, you may have a basic interface to a database and wish for the user to be able to define their own interfaces, which are automatically stored in a master registry.
You might proceed this way:
class DBInterfaceMeta(type): we use $_{i}nit_{ratherthan}{}_{n}ew_{herebecausewewanttomodifyattributesoftheclass*after*theyhavebeencreat}$

super(DBInterfaceMeta, cls).$_{i}nit_{(name,bases,dct)Ourmetaclasssimplyaddsaregistrydictionaryifit'snotalreadypresent,andaddsthenew}$
class DBInterface(object): $_{m}etaclass_{=DBInterfaceMeta}$
print(DBInterface.registry) Now let's create some subclasses, and double-check that they're added to the registry:
class FirstInterface(DBInterface): pass
class SecondInterface(DBInterface): pass
class SecondInterfaceModified(SecondInterface): pass
print(DBInterface.registry) 'firstinterface': <class '$_{m}ain_{.FirstInterface'>,'secondinterface':<class'}{}_{m}ain_{.SecondInterface'>,'}$

Conclusion: When Should You Use Metaclasses? I've gone through some examples of what metaclasses are, and some ideas about how they might be used to create very powerful and flexible APIs. Although metaclasses are in the background of everything you do in Python, the average coder rarely has to think about them.
But the question remains: when should you think about using custom metaclasses in your project? It's a complicated question, but there's a quotation

floating around the web that addresses it quite succinctly:
Metaclasses are deeper magic than 99% of users should ever worry about. If you wonder whether you need them, you don't (the people who actually need them know with certainty that they need them, and don't need an explanation about why).
Tim Peters
In a way, this is a very unsatisfying answer: it's a bit reminiscent of the wistful and cliched explanation of the border between attraction and love: "well, you just... know!"
But I think Tim is right: in general, I've found that most tasks in Python that can be accomplished through use of custom metaclasses can also be accomplished more cleanly and with more clarity by other means. As programmers, we should always be careful to avoid being clever for the sake of cleverness alone, though it is admittedly an ever-present temptation.
I personally spent six years doing science with Python, writing code nearly on a daily basis, before I found a problem for which metaclasses were the natural solution. And it turns out Tim was right:
I just knew.

2.7.2 Design Patterns

Design Patterns Singleton Non-thread-safe Paul Manta's implementation of singletons

```
@Singleton
class Foo:
   def __init__(self):
       print 'Foo created'

f = Foo() # Error, this isn't how you get the instance of a singleton

f = Foo.Instance() # Good. Being explicit is in line with the Python
    ↪ Zen
g = Foo.Instance() # Returns already created instance

print f is g # True

class Singleton:
    """
    A non-thread-safe helper class to ease implementing singletons.
    This should be used as a decorator -- not a metaclass -- to the
    class that should be a singleton.

    The decorated class can define one `__init__` function that
    takes only the `self` argument. Also, the decorated class cannot be
    inherited from. Other than that, there are no restrictions that
        ↪ apply
    to the decorated class.

    To get the singleton instance, use the `Instance` method. Trying
    to use `__call__` will result in a `TypeError` being raised.

    """

    def __init__(self, decorated):
        self._decorated = decorated

    def Instance(self):
        """
        Returns the singleton instance. Upon its first call, it creates
            ↪ a
        new instance of the decorated class and calls its `__init__`
            ↪ method.
        On all subsequent calls, the already created instance is
            ↪ returned.

        """
        try:
            return self._instance
        except AttributeError:
            self._instance = self._decorated()
            return self._instance

    def __call__(self):
        raise TypeError('Singletons must be accessed through `Instance()
            ↪ `.')

    def __instancecheck__(self, inst):
        return isinstance(inst, self._decorated)
```

Thread safe
weredriver's implementation of singletons. A thread safe implementation of singleton pattern in Python. Based on tornado.ioloop.IOLoop.instance() approach.
import threading
Based on tornado.ioloop.IOLoop.instance() approach. See https://github.com/facebook/tornado
class SingletonMixin(object): $_{singleton_lock=threading.Lock()_{singleton_instance=None}}$
@classmethod def instance(cls): if not cls.$_{singleton_instance:withcls._{singleton_lock:ifnotcls._{singleton_instance:cls._{singleton_instan}}}}$

class A(SingletonMixin): pass
class B(SingletonMixin): pass
if $_{name}{}_{=='}{}_{main}{}_{':a,a2=A.instance(),A.instance()b,b2=B.instance(),B.instance()}$
assert a is a2 assert b is b2 assert a is not b
print('a: print('b: Suggested Readings Is there a simple, elegant way to define singletons?

2.8 File System IO

JSON Write json file with pretty format and unicode
import json import io

data = "menu": "header": "Sample Menu", "items": ["id": "Open", "id": "OpenNew", "label": "Open New", None, "id": "Help", "id": "About", "label": "About Adobe CVG Viewer..."]
with io.open("sample$_json.json", "w", encoding = "utf8")asf : content = json.dumps(data, indent =$ 4, $sort_keys = True, ensure_ascii = False)f.write(unicode(content))Result$
"menu": "header": "Sample Menu", "items": ["id": "Open" , "id": "OpenNew", "label": "Open New" , null, "id": "Help" , "id": "About", "label": "About Adobe CVG Viewer..."] Read json file
import json from pprint import pprint
with open('sample$_json.json')asdata_file : data = json.load(data_file)$
pprint(data) Result
u'menu': u'header': u'Sample Menu', u'items': [u'id': u'Open', u'id': u'OpenNew', u'label': u'Open New', None, u'id': u'Help', u'id': u'About', u'label': u'About Adobe CVG Viewer...'] Related Reading
Parsing values from a JSON file in Python, stackoverflow How do I write JSON data to a file in Python?, stackoverflow XML Write xml file with lxml package
import lxml.etree as ET root declaration root = ET.Element('catalog') insert comment comment = ET.Comment(' this is a xml sample file ') root.insert(1, comment) book element book = ET.SubElement(root, 'book', id="bk001") book data author = ET.SubElement(book, 'author') author.text = "Gambardella, Matthew" title = ET.SubElement(book, 'title') title.text = "XML Developer's Guide" write xml to file tree = ET.ElementTree(root) tree.write("sample$_book.xml", pretty_print =$ $True, xml_declaration = True, encoding =' utf - 8')Result$
<?xml version='1.0' encoding='UTF-8'?> <catalog> <!– this is a xml sample file –> <book id="bk001"> <author>Gambardella, Matthew</author> <title>XML Developer's Guide</title> </book> </catalog> Read xml file with lxml package
from lxml import etree as ET
tree = ET.parse("sample$_book.xml")root = tree.getroot()book = root.find('book')print"BookInformation"p$
$", book.attrib["id"]print"Author : ", book.find('author').textprint"Title : ", book.find('title').textResult$
Book Information ID : bk001 Author : Gambardella, Matthew Title : XML Developer's Guide

2.9 Operating System

File Operations Copy folder 1
import shutil shutil.copyfile("src", "dst") CLI shutil - High-level file operations

2.10 Networking

REST JSON 1 2 GET
import requests url = "http://localhost:8080/messages" response = requests.get(url) data = response.json() POST 3
import requests import json
url = "http://localhost:8080/messages" data = 'sender': 'Alice', 'receiver': 'Bob', 'message': 'Hello!' headers = 'Content-type': 'application/json', 'Accept': 'application/json' r = requests.post(url, data=json.dumps(data), headers=headers)

2.11 Concurrency and Parallelism

Running several threads is similar to running several different programs concurrently, but with the following benefits
Multiple threads within a process share the same data space with the main thread and can therefore share information or communicate with each other more easily than if they were separate processes. Threads sometimes called light-weight processes and they do not require much memory overhead; they are cheaper than processes. A thread has a beginning, an execution sequence, and a conclusion. It has an instruction pointer that keeps track of where within its context it is currently running.
It can be pre-empted (interrupted) It can temporarily be put on hold (also known as sleeping) while other threads are running - this is called yielding. Starting a New Thread To spawn another thread, you need to call following method available in thread module:
thread.start$_{n}ew_{t}hread(function, args[, kwargs])Thismethodcallenablesafastandefficientwaytocreatenew$
The method call returns immediately and the child thread starts and calls function with the passed list of args. When function returns, the thread terminates. Here, args is a tuple of arguments; use an empty tuple to call function without passing any arguments. kwargs is an optional dictionary of keyword arguments.
Example
!/usr/bin/python
import thread import time
Define a function for the thread def print$_{t}ime(threadName, delay) : count = 0 while count < 5 : time.sleep(delay) count += 1 print$”
Create two threads as follows try: thread.start$_{n}ew_{t}hread(print_{t}ime, (''Thread- 1'', 2,)) thread.start_{n}ew_{t}hread(print_{t}ime, (''Thread-2'', 4,)) except : print''Error : unabletostartthread$”
while 1: pass When the above code is executed, it produces the following result Thread-1: Thu Jan 22 15:42:17 2009 Thread-1: Thu Jan 22 15:42:19 2009 Thread-2: Thu Jan 22 15:42:19 2009 Thread-1: Thu Jan 22 15:42:21 2009 Thread-2: Thu Jan 22 15:42:23 2009 Thread-1: Thu Jan 22 15:42:23 2009 Thread-1: Thu Jan 22 15:42:25 2009 Thread-2: Thu Jan 22 15:42:27 2009 Thread-2: Thu Jan 22 15:42:31 2009 Thread-2: Thu Jan 22 15:42:35 2009 Although it is very effective for low-level threading, but the thread module is very limited compared to the newer threading module.
The Threading Module The newer threading module included with Python 2.4 provides much more powerful, high-level support for threads than the thread module discussed in the previous section.
The threading module exposes all the methods of the thread module and provides some additional methods:
threading.activeCount(): Returns the number of thread objects that are active. threading.currentThread(): Returns the number of thread objects in the caller's thread control. threading.enumerate(): Returns a list of all thread objects that are currently active. In addition to the methods, the threading module has the Thread class that implements threading. The methods provided by the Thread class are as follows:
run(): The run() method is the entry point for a thread. start(): The start() method starts a thread by calling the run method. join([time]): The join() waits for threads to terminate. isAlive(): The isAlive() method checks whether

a thread is still executing. getName(): The getName() method returns the name of a thread. setName(): The setName() method sets the name of a thread. Creating Thread Using Threading Module To implement a new thread using the threading module, you have to do the following
Define a new subclass of the Thread class. Override the init(self [,args]) method to add additional arguments. Then, override the run(self [,args]) method to implement what the thread should do when started. Once you have created the new Thread subclass, you can create an instance of it and then start a new thread by invoking the start(), which in turn calls run() method.
Example
!/usr/bin/python
import threading import time
exitFlag = 0
class myThread (threading.Thread): def ${}_{i}nit_{(self,threadID,name,counter):threading.Thread.{}_{i}nit_{(self)self.threadID=threadIDsel}}$
def print$_{t}ime(threadName, delay, counter) : whilecounter : ifexitFlag : threadName.exit()time.sleep(dela$
1
Create new threads thread1 = myThread(1, "Thread-1", 1) thread2 = myThread(2, "Thread-2", 2)
Start new Threads thread1.start() thread2.start()
print "Exiting Main Thread" When the above code is executed, it produces the following result
Starting Thread-1 Starting Thread-2 Exiting Main Thread Thread-1: Thu Mar 21 09:10:03 2013 Thread-1: Thu Mar 21 09:10:04 2013 Thread-2: Thu Mar 21 09:10:04 2013 Thread-1: Thu Mar 21 09:10:05 2013 Thread-1: Thu Mar 21 09:10:06 2013 Thread-2: Thu Mar 21 09:10:06 2013 Thread-1: Thu Mar 21 09:10:07 2013 Exiting Thread-1 Thread-2: Thu Mar 21 09:10:08 2013 Thread-2: Thu Mar 21 09:10:10 2013 Thread-2: Thu Mar 21 09:10:12 2013 Exiting Thread-2 Synchronizing Threads The threading module provided with Python includes a simple-to-implement locking mechanism that allows you to synchronize threads. A new lock is created by calling the Lock() method, which returns the new lock. The acquire(blocking) method of the new lock object is used to force threads to run synchronously. The optional blocking parameter enables you to control whether the thread waits to acquire the lock.
If blocking is set to 0, the thread returns immediately with a 0 value if the lock cannot be acquired and with a 1 if the lock was acquired. If blocking is set to 1, the thread blocks and wait for the lock to be released.
The release() method of the new lock object is used to release the lock when it is no longer required.
Example
!/usr/bin/python
import threading import time
class myThread (threading.Thread): def ${}_{i}nit_{(self,threadID,name,counter):threading.Thread.{}_{i}nit_{(self)self.threadID=threadIDsel}}$
def print$_{t}ime(threadName, delay, counter) : whilecounter : time.sleep(delay)print" counter- =$
1
threadLock = threading.Lock() threads = []
Create new threads thread1 = myThread(1, "Thread-1", 1) thread2 = myThread(2, "Thread-2", 2)
Start new Threads thread1.start() thread2.start()

Add threads to thread list threads.append(thread1) threads.append(thread2) Wait for all threads to complete for t in threads: t.join() print "Exiting Main Thread" When the above code is executed, it produces the following result Starting Thread-1 Starting Thread-2 Starting Thread-3 Thread-1 processing One Thread-2 processing Two Thread-3 processing Three Thread-1 processing Four Thread-2 processing Five Exiting Thread-3 Exiting Thread-1 Exiting Thread-2 Exiting Main Thread Related Readings "Python Multithreaded Programming". www.tutorialspoint.com. N.p., 2016. Web. 13 Dec. 2016. "An Introduction To Python Concurrency". dabeaz.com. N.p., 2016. Web. 14 Dec. 2016.

2.12 Event Based Programming

Introduction: pydispatcher 1 2 PyDispatcher provides the Python programmer with a multiple-producer-multiple-consumer signal-registration and routing infrastructure for use in multiple contexts. The mechanism of PyDispatcher started life as a highly rated recipe in the Python Cookbook. The project aims to include various enhancements to the recipe developed during use in various applications. It is primarily maintained by Mike Fletcher. A derivative of the project provides the Django web framework's "signal" system.
Used by Django community
Usage 1 To set up a function to receive signals: from pydispatch import dispatcher
SIGNAL = 'my-first-signal'
def handle$_event(sender): """Simpleeventhandler""" print'Signalwassentby', sender$
dispatcher.connect(handle$_event, signal = SIGNAL, sender = dispatcher.Any)$
The use of the Any object allows the handler to listen for messages from any Sender or to listen to Any message being sent. To send messages: first$_sender =$ $object() second_sender =$
def main(): dispatcher.send(signal=SIGNAL, sender=first$_sender) dispatcher.send(signal =$ $SIGNAL, sender = second_sender)$
Which causes the following to be printed:
Signal was sent by <object object at 0x196a090> Signal was sent by Messaging Conda link Docker link Github - pubSubService Github - pubSubClient Pypi link
Python Publish - Subscribe Pattern Implementation:
Step by Step to run PubSub: Step 1: Pull pubsub image from docker hub run it: docker pull hunguyen/pubsub:latest docker run -d -p 8000:8000 hunguyen/pubsub Step 2: To run client first install pyconfiguration from conda conda install -c rain1024 pyconfiguration Step 3: Install pubSubClient package from conda conda install -c hunguyen pubsubclient Step 4: Create config.json file "PUBLISH$_SUBSCRIBE_SERVICE" : "http : //api.service.com" Step5 :$
$Runpubsubclientcreateandregisterorsyncapublisherpublisher = Publisher('P1')createanewtopictopic =$
$Topic('A')createaneventofatopicevent = Event(topic)publisherpublishesaneventpublisher.publish(event)c$
$Subscriber('S1')subscribersubscribestoatopicsubscriber.subscribe(topic)subscribergetallneweventsbytimest$
$subscriber.get_events()pydispatcher$
stackoverflow, Recommended Python publish/subscribe/dispatch module?

2.13 Web Development

Django 1 Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.
Project Folder Structure
project$_f$*older/your$_p$roject$_n$ame/your$_p$roject$_n$ame/static/models.pyserializers.pysettings.pyurls.pyviews.p*
InstalldependenciespipinstalldjangopipinstalldjangorestframeworkpipinstallmarkdownMarkdownsuppo
*filterFilteringsupportpipinstalldjango − cors − headersCORSsupportStep*2 :
*Createprojectdjango−adminstartprojectyour$_p$roject$_n$ameStep*3 : *Configapps*3*Add′your$_p$roject$_n$ame′,′ res*
INSTALLED$_A$*PPS* = (...′*your$_p$roject$_n$ame′′rest$_f$ramework′*,)*Step*4 : *Model*, *View*, *Route*6*Step*4.1 :
CreatemodelandserializerYoucangotoDjango : *Modelfieldreferencepageformorefields.*
Step 4.1.1: Create Task class in your$_p$*roject$_n$ame/models.pyfilefromdjango.dbimportmodels*
class Task(models.Model): content = models.CharField(max$_l$*ength* = 30)*status* =
models.CharField(*max$_l$ength* = 30)*Step*4.1.2 : *CreateTaskSerializerclassinyour$_p$roject$_n$ame/serializers.*
class TaskSerializer(serializers.HyperlinkedModelSerializer): class Meta: model = Task fields = ('id', 'content', 'status') Step 4.1.3: Create table in database 4
python manage.py syncdb With django 1.9
python manage.py makemigrations your$_p$*roject$_n$amepythonmanage.pymigrateStep*4.2 :
CreateTaskViewSetclassinyour$_p$roject$_n$ame/views.pyfilefromyour$_p$roject$_n$ame.modelsimportTaskfrom
class TaskViewSet(viewsets.ModelViewSet): queryset = Task.objects.all() serializer$_c$*lass* =
*TaskSerializerStep*4.3 : *Configroute*5*Changeyour$_p$roject$_n$ame/urls.pyfile*
from django.conf.urls import include, url from django.contrib import admin
from rest$_f$*rameworkimportroutersfromyour$_p$roject$_n$ame.viewsimportTaskViewSet*
router = routers.DefaultRouter() router.register(r'api/tasks', TaskViewSet) admin.autodiscover()
urlpatterns = [url(r'^{a}*dmin/′, include(admin.site.urls)), url(r′′, include(router.urls)), url(r′^{a}pi*−
*auth/′, include(′rest$_f$ramework.urls′, namespace =′ rest$_f$ramework′))]Step*5 :
*RunServerpythonmanage.pyrunserverStep*6.*UseAPIStep*6.1 : *Createanewtaskcurl*−
i − XPOST − H”Content − Type : application/json”http : //localhost :
8000/*api/tasks−d′”content”* : ”*a*”, ”*status*” : ”*INIT*”′*Step*6.2 : *Listalltaskscurlhttp* :
//*localhost* : 8000/*api/tasksStep*6.3 : *Getdetailoftask*1*curlhttp* : //*localhost* :
8000/*api/tasks*/1*Step*6.4 : *Deletetask*1*curl−i−XDELETEhttp* : //*localhost* :
8000/*api/tasks*/1*Step*7 : *CORSKnownError* : *No′Access−Control−Allow*−
Origin′headerispresentontherequestedresource.Origin′null′isthereforenotallowedaccess.
Step 7.1: Install corsheader app Add module corsheaders to your$_p$*roject$_n$ame/settings.py*
INSTALLED$_A$*PPS* = (...′*corsheaders′*, ...)*Step*7.2*AddmiddlewareclassesAddmiddleware$_c$lassestoyour$_p$roj*
MIDDLEWARE$_C$*LASSES* = (...′*corsheaders.middleware.CorsMiddleware′,′ django.middleware.commo*
AllowAll
Add this line to your$_p$*roject$_n$ame/settings.py*
CORS$_O$*RIGIN$_A$LLOW$_A$LL* : *TrueStep*8 : *httpsYoucanusehttps* : //*github.com/teddziuba/django*−
sslserver
Unicode REST$_F$*RAMEWORK* = ′*DEFAULT$_R$ENDERER$_C$LASSES′* : (′*rest$_f$ramework.renderers.JS*
PagingAddthismodulesettingtoyour$_p$roject$_n$ame/settings.py
REST$_F$*RAMEWORK* = ′*DEFAULT$_P$AGINATION$_C$LASS′* :′ *rest$_f$ramework.pagination.LimitOffse*
API:
GET <>/?limit=<limit>offset=<offset>
Step 10: Search by field in import this to your viewsets.py
from rest$_f$*rameworkimportfilters*

add this to your viewsets class
filter$_{b}ackends = (filters.SearchFilter,)search_{f}ields = ('<field>','<field>'$
,)
One-to-Many Relationship 7 from django.db import models
class User(models.Model): name = models.TextField()
def $_{str}{}_{(self):return"-".format(str(self.id),self.name)}$
class Task(models.Model): name = models.TextField() assign = models.ForeignKey(User,
on$_{d}elete = models.CASCADE)StartingwithMysqlAddthisdatabasesettingstoyour_{p}roject_{n}ame/settings.p$
DATABASES = 'default': 'ENGINE': 'django.db.backends.mysql', 'NAME':
'[DB$_{N}AME]','USER' :' [DB_{U}SER]','PASSWORD' :' [PASSWORD]','HOST' :'$
$[HOST]', OranIPAddressthatyourDBishostedon'PORT' :' 3306'$,
Install this module to your virtual environment
conda install mysql-python if you are using virtual environment
pip install mysql-python if you using are root environment
Custom View 8 from rest$_{f}rameworkimportmixins$
class CreateModelMixin(object): """ Create a model instance. """ def create(self, request, *args, **kwargs): event = request.data try: event['time'] = int(time.time()) except Exception, e: print 'Set Time Error' serializer = self.get$_{s}erializer(data = request.data)serializer.is_{v}alid(raise_{e}xception = True)self.perform_{c}reate(serializer)headers = self.get_{s}uccess_{h}eaders(serializer.data)returnResponse(serializer.data, status = status.HTTP_201_{C}REATED, headers = headers)$
def perform$_{c}reate(self, serializer) : serializer.save()$
def get$_{s}uccess_{h}eaders(self, data) : try : return'Location' : data[api_{s}ettings.URL_{F}IELD_{N}AME]except(Ty$
$return$
class YourViewSet(CreateModelMixin, mixins.RetrieveModelMixin, mixins.UpdateModelMixin, mixins.DestroyModelMixin, mixins.ListModelMixin, GenericViewSet): queryset = YourModel.objects.all() serializer$_{c}lass = YourModelSerializerLoggingsettingsHereisanexample, putthi$
LOGGING = 'version': 1, 'disable$_{e}xisting_{l}oggers' : False,'formatters' :$
$'verbose' : 'format' :', 'simple' : 'format' :',,'filters' : 'special' : '()' :' project.logging.SpecialFilter','foo'$
$'console' : 'level' :' INFO','filters' : ['require_{d}ebug_{t}rue'],'class' :' logging.StreamHandler','formatter' :'$
$'django' : 'handlers' : ['console'],'propagate' : True,,'django.request' : 'handlers' : ['mail_{a}dmins'],'level' :'$
Python: Build Python API Client package Step 1: Write document on Swagger Editor1 Step 2: Genenrate Client –> Python –> save python-client.zip Step 3: Extract zip Step 4: Open project in Pycharm rename project directory, project name, swagger$_{c}lientpackageStep5 : 2mkdircondacdcondagitclonehttps : //github.com/hunguyen1702/conda$
$rf.gitREADME.mdStep6 : Editmeta.yamlfileinyour_{p}ackagefolder6.1Followinstructioninsidemeta.yaml$
$build : -python - setuptoolsrun : -pythonwith : requirements : build : -python - setuptools - six - certifi - python - dateutilrun : -python - six - certifi - python - dateutilStep7 : cd..condabuildyour_{p}ackageStep8 : mkdirchannelcdchannelcondaconvert--platformall /anaconda/conda-bld/linux-64/your_{p}ackage_{0}.1.0-py27_{0}.tar.bz2Step9 : Createvirtual-envname : your_{e}nv_{n}amedependencies : -certifi = 2016.2.28 = py27_{0} - openssl = 1.0.2h = 0 - pip = 8.1.2 = py27_{0} - python = 2.7.11 = 0 - python - dateutil = 2.5.3 = py27_{0} - readline = 6.2 = 2 - setuptools = 20.7.0 = py27_{0} - six = 1.10.0 = py27_{0} - tk = 8.5.18 = 0 - wheel = 0.29.0 = py27_{0} - zlib = 1.2.8 = 0 - pip : -urllib3 == 1.15.1Step10 : Install : condainstall - -use - localyour_{p}ackageDjango$
Writing your first Django app, part 1
Django REST framework: Installation
Django: Migrations

Building a Simple REST API for Mobile Applications
Django: Models
How to show object details in Django Rest Framework browseable API?
rest$_{f}ramework : mixins$

2.14 Logging

logging 1 2 3 levels, attributes references
The logging library takes a modular approach and offers several categories of components: loggers, handlers, filters, and formatters.
Loggers expose the interface that application code directly uses. Handlers send the log records (created by loggers) to the appropriate destination. Filters provide a finer grained facility for determining which log records to output. Formatters specify the layout of log records in the final output. Step 0: Project structure
code/ main.py config logging.conf logs app.log Step 1: Create file logging.conf
[loggers] keys=root
[handlers] keys=consoleHandler,fileHandler
[formatters] keys=formatter
[logger$_{r}oot]level = DEBUGhandlers = consoleHandler, fileHandler$
[handler$_{c}onsoleHandler]class = StreamHandlerlevel = DEBUGformatter = formatterargs = (sys.stdout,)$
[handler$_{f}ileHandler]class = FileHandlerlevel = DEBUGformatter = formatterargs = ('logs/app.log',' a')$
[formatter$_{f}ormatter]format = datefmt = Step2 : Loadconfigandcreatelogger$
In main.py
import logging.config
load logging config logging.config.fileConfig('config/logging.conf') Step 3: In your application code
logging.getLogger().debug('debug message') logging.getLogger().info('info message') logging.getLogger().warn('warn message') logging.getLogger().error('error message') logging.getLogger().critical('critical message') More Resources
Introduction to Logging Quick and simple usage of python log Python: Logging module
Python: Logging cookbook
Python: Logging guide

2.15 Configuration

pyconfiguration
Installation conda install -c rain1024 pyconfiguration Usage Step 1: Create config.json file
"SERVICE$_{U}RL" : "http : //api.service.com" Step2 : Addthesecodetomain.pyfile$
from pyconfiguration import Configuration Configuration.load('config.json') print Configuration.SERVICE$_{U}RL$
> http://api.service.com References: What's the best practice using a settings file 1
What's the best practice using a settings file in Python?

2.16 Command Line

Command Line Arguments There are the following modules in the standard library:
The getopt module is similar to GNU getopt. The optparse module offers object-oriented command line option parsing. Here is an example that uses the latter from the docs:
from optparse import OptionParser
parser = OptionParser() parser.add$_option("-f","--file",dest="filename",help="writereporttoFILE",metavar="FILE")parser.add_option("-q","--quiet",action="store_false",dest="verbose",default=True,help="don'tprintstatusmessagestostdout")$
(options, args) = parser.parse$_args()optparsesupports(amongotherthings):$
Multiple options in any order. Short and long options. Default values. Generation of a usage help message. Suggest Reading Command Line Arguments In Python

2.17 Testing

Testing your code is very important.
Getting used to writing testing code and running this code in parallel is now considered a good habit. Used wisely, this method helps you define more precisely your code's intent and have a more decoupled architecture.
Unittest unittest is the batteries-included test module in the Python standard library. Its API will be familiar to anyone who has used any of the JUnit/-nUnit/CppUnit series of tools.
The Basics Creating test cases is accomplished by subclassing unittest.TestCase.
import unittest
def fun(x): return x + 1
class MyTest(unittest.TestCase): def test(self): self.assertEqual(fun(3), 4) Skipping tests Unittest supports skipping individual test methods and even whole classes of tests. In addition, it supports marking a test as an "expected failure," a test that is broken and will fail, but shouldn't be counted as a failure on a .code TestResult.
Skipping a test is simply a matter of using the skip() decorator or one of its conditional variants.
import sys import unittest
class MyTestCase(unittest.TestCase):
@unittest.skip("demonstrating skipping") def test$_nothing(self):self.fail("shouldn'thappen")$
@unittest.skipIf(mylib.$_{version_{<(1,3),"notsupportedinthislibraryversion")deftest_format(self):Teststhatworkforonlyacertainversionof}}$
@unittest.skipUnless(sys.platform.startswith("win"), "requires Windows") def
test$_windows_support(self):windowsspecifictestingcodepassToxtoxaimstoautomateandstandardizetesting$
Tox is a generic virtualenv management and test command line tool you can use for:
checking your package installs correctly with different Python versions and interpreters running your tests in each of the environments, configuring your test tool of choice acting as a frontend to Continuous Integration servers, greatly reducing boilerplate and merging CI and shell-based testing. Installation
You can install tox with pip using the following command

```
> pip install tox
```

Setup default environment in Windows with conda

```
> conda create −p C:\python27 python=2.7
> conda create −p C:\python34 python=3.4
```

Related Readings Testing Your Code, The Hitchhiker's Guide to Python unittest Unit testing framework, docs.python.org Is it possible to use tox with conda-based Python installations?, stackoverflow

2.18 IDE Debugging

Today, I write some notes about my favorite Python IDE - PyCharm. I believe it's a good one for developing python, which supports git, vim, etc. This list below contains my favorite features.
Pycharm Features Intelligent Editor Navigation Graphical Debugger Refactorings Code Inspections Version Control Integration Scientific Tools Intelligent Editor PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.
Syntax Highlighting
Read your code easier with customizable colors for Python code and Django templates. Choose from several predefined color themes.
Auto-Identation and code formating
Automatic indents are inserted on new line. Indent verification and code reformatting are compliant with project code-style settings.
Configurable code styles
Select a predefined coding style to apply to your code style configuration for various supported languages.
Code completion
Code completion for keywords, classes, variables, etc. as you type or via Ctrl+Space. Editor suggestions are context-aware and offer the most appropriate options.
Keyboard shortcuts: Tab, Alt+Enter
Code selection and comments
Select a block of code and expand it to an expression, to a line, to a logical block of code, and so on with shortcuts. Single keystroke to comment/uncomment the current line or selection.
Code formatter
Code formatter with code style configuration and other features help you write neat code that's easy to support. PyCharm contains built-in PEP-8 for Python and other standards compliant code formatting for supported languages.
Code snippets and templates
Save time using advanced customizable and parametrized live code templates and snippets.
Keyboard shortcuts check.if ENTER
if check: type$_{s}omethingCodefolding$
Code folding, auto-insertion of braces, brackets quotes, matching brace/bracket highlighting, etc.
On-the-fly error highlighting
Errors are shown as you type. The integrated spell-checker verifies your identifiers and comments for misspellings.

Multiple carets and selections
With multiple carets, you can edit several locations in your file at the same time.
Keyboard shortcuts: SHIFT + F6
Code analysis
Numerous code inspections verify Python code as you type and also allow inspecting the whole project for possible errors or code smells.
Quick-fixes
Quick-fixes for most inspections make it easy to fix or improve the code instantly. Alt+Enter shows appropriate options for each inspection.
Keyboard shortcuts: F2
Duplicated code detector
Smart duplicated code detector analyzes your code and searches for copy/pasted code. You'll be presented with a list of candidates for refactoring and with the help of refactorings it's easy to keep your code dry.
Configurable language injections
Natively edit non-Python code embedded into string literals, with code completion, error-highlighting, and other coding assistance features.
Code auto generation
Code auto-generation from usage with quick-fixes; docstrings and the code matching verification, plus autoupdate on refactoring. Automatic generation of a docstring stub (reStructuredText, Epytext, Google, and NumPy).
Intention actions
Intention actions help you apply automated changes to code that is correct, to improve it or to make your coding routine easier.
Searching
Keyboard shortcuts: Double Shift (search everywhere)
Navigation Shortcuts
Keyboard shortcuts: ALT + SHIFT + UP/DOWN (move line up and down)
Graphical Debugger PyCharm provides extensive options for debugging your Python/Django and JavaScript code:
Set breakpoints right inside the editor and define hit conditions Inspect context-relevant local variables and user-defined watches, including arrays and complex objects, and edit values on the fly Set up remote debugging using remote interpreters Evaluate an expression in runtime and collect run-time type statistics for better autocompletion and code inspections Attach to a running process Debug Django templates
Inline Debugger
With an inline debugger, all live debugging data are shown directly in the editor, with variable values integrated into the editor's look-and-feel. Variable values can be viewed in the source code, right next to their usages.
Step into My Code
Use Step into My Code to stay focused on your code: the debugger will only step through your code bypassing any library sources.
Multi-process debugging
PyCharm can debug applications that spawn multiple Python processes, such as Django applications that don't run in –no-reload mode, or applications using many other Web frameworks that use a similar approach to code auto-reloading.
Run/Debug configurations

Every script/test or debugger execution creates a special 'Run/Debug Configuration' that can be edited and used later. Run/Debug Configurations can be shared with project settings for use by the whole team.
Workspace Custom Scheme Go to File - Settings... then Editor - Colors Fonts
Now you can change your scheme, I like Darcular
https://confluence.jetbrains.com/download/attachments/51945983/appearance3.png?version=1modification
IPython Support PyCharm supports usage of IPython magic commands.
http://i.stack.imgur.com/aTEW2.png
Vim Support You can configure PyCharm to work as a Vim editor
https://confluence.jetbrains.com/download/attachments/51946537/vim4.png?version=1modificationDate=1
Keyboard Shortcuts: Ctrl+Shift+V (paste)

2.19 Package Manager

py2exe py2exe is a Python Distutils extension which converts Python scripts into executable Windows programs, able to run without requiring a Python installation.Spice
Installation py2exe conda install -c https://conda.anaconda.org/clinicalgraphics cg-py2exe Build 1 python setup.py py2exe build PyQT python setup.py py2exe –includes sip Known Issues Error: Microsoft Visual C++ 10.0 is required (Unable to find vcvarsall.bat) (link)
How to fix
Step 1: Install Visual Studio 2015
Step 2:
set VS100COMNTOOLS=

2.20 Environment

Similar to pip, conda is an open source package and environment management system 1. Anaconda is a data science platform that comes with a lot of packages. It uses conda at the core. Unlike Anaconda, Miniconda doesn't come with any installed packages by default. Note that for miniconda, everytime you open up a terminal, conda won't automatically be available. Run the command below to use conda within miniconda.
Conda Let's first start by checking if conda is installed.

```
> conda --version

conda 4.2.12
To see the full documentation for any command, type the command
    ↪ followed by --help. For example, to learn about the conda
    ↪ update command:

> conda update --help
Once it has been confirmed that conda has been installed, we will now
    ↪ make sure that it is up to date.

> conda update conda
```

```
Using Anaconda Cloud api site https://api.anaconda.org
Fetching package metadata: ....
.Solving package specifications :    .........

Package plan for installation in environment //anaconda:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
       ↪
    conda-env-2.6.0            |                0          601 B
    ruamel_yaml-0.11.14        |          py27_0         184 KB
    conda-4.2.12               |          py27_0         376 KB
    ------------------------------------------------------------
       ↪
                                          Total:         560 KB

The following NEW packages will be INSTALLED:

    ruamel_yaml: 0.11.14-py27_0

The following packages will be UPDATED:

    conda:      4.0.7-py27_0 --> 4.2.12-py27_0
    conda-env:  2.4.5-py27_0 --> 2.6.0-0
    python:     2.7.11-0     --> 2.7.12-1
     sqlite :   3.9.2-0      --> 3.13.0-0

Proceed ([y]/n)? y

Fetching packages ...
conda-env-2.6. 100% |#
    ↪ ###############################| Time:
    ↪ 0:00:00 360.78 kB/s
ruamel_yaml-0. 100% |#
    ↪ ###############################| Time:
    ↪ 0:00:00   5.53 MB/s
conda-4.2.12-p 100% |#
    ↪ ###############################| Time:
    ↪ 0:00:00   5.84 MB/s
Extracting packages ...
[      COMPLETE      ]|#
    ↪ #################################################|
    ↪  100%
Unlinking packages ...
[      COMPLETE      ]|#
    ↪ #################################################|
    ↪  100%
Linking packages ...
```

```
[        COMPLETE  ]| #
  ↪ ################################################|
  ↪  100%
Environments
```

2.20.1 Create

In order to manage environments, we need to create at least two so you can move or switch between them. To create a new environment, use the conda create command, followed by any name you wish to call it:

```
# create new environment
conda create −n <your_environment> python=2.7.11
```

2.20.2 Clone

Make an exact copy of an environment by creating a clone of it. Here we will clone snowflakes to create an exact copy named flowers:

```
conda create −−name flowers −−clone snowflakes
```

2.20.3 List

List all environments
Now you can use conda to see which environments you have installed so far. Use the conda environment info command to find out

```
> conda info −e

conda environments:
snowflakes              /home/username/miniconda/envs/snowflakes
bunnies                 /home/username/miniconda/envs/bunnies
```

Verify current environment
Which environment are you using right now - snowflakes or bunnies? To find out, type the command:

```
conda info −−envs
```

2.20.4 Remove

If you didn't really want an environment named flowers, just remove it as follows:

```
conda remove −−name flowers −−all
```

2.20.5 Share

You may want to share your environment with another person, for example, so they can re-create a test that you have done. To allow them to quickly reproduce your environment, with all of its packages and versions, you can give them a copy of your environment.yml file.

Export the environment file
To enable another person to create an exact copy of your environment, you will export the active environment file.

```
conda env export > environment.yml
```

Use environment from file
Create a copy of another developer's environment from their environment.yml file:

```
conda env create −f environment.yml
# remove environment
conda remove −n <your_environemnt> −−all
```

2.21 Module

Create Public Module conda, pypi, github
Step 0/4: Check your package name Go to https://pypi.python.org/pypi/your$_{p}ackage_{n}ametoseeyourpackage$
Step 1/4: Make your module 1 1.1 pip install cookiecutter
1.2 cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git
1.3 Fill all necessary information
full$_{n}ame[AudreyRoyGreenfeld] : email[aroy@alum.mit.edu] : github_{u}sername[audreyr] : project_{n}ame[PythonBoilerplate] : project_{s}lug[] : project_{s}hort_{d}escription : release_{d}ate[] : pypi_{u}sername[] : year[2016] : version[0.1.0] : use_{p}ypi_{d}eployment_{w}ith_{t}ravis[y] : Itwillcreateadirectory$
|- LICENSE |- README.md |- TODO.md |- docs | |– conf.py | |– generated | |– index.rst | |– installation.rst | |– modules.rst | |– quickstart.rst | |– sandman.rst |- requirements.txt |- your$_{p}ackage||--_{init.py||--your_{p}ackage.py||--test||--models.py||--test_{y}our_{p}ackage.py|-setup.pyStep2/4:Gi}$
2. Create a .pypirc configuration file in $HOMEdirectory$
[distutils] index-servers = pypi
[pypi] repository=https://pypi.python.org/pypi username=your$_{u}sernamepassword = your_{p}assword3.ChangeyourMANIFEST.in$
recursive-include project$_{f}older * 4.UploadyourpackagetoPyPI$
python setup.py register -r pypi python setup.py sdist upload -r pypi Step 4/4: Conda 2 1. Install conda tools
conda install conda-build conda install anaconda-client 2. Build a simple package with conda skeleton pypi
cd your$_{p}ackage_{f}oldermkdircondacdcondacondaskeletonpypiyour_{p}ackageThiscreatesadirectorynamedyour_{p}$
|- your$_{p}ackage|| - -bld.bat|| - -meta.yaml|| - -build.sh3.Buildyourpackage$
conda build your$_{p}ackage$
convert to all platform conda convert -f –platform all C:-bld-64$_{p}ackage - 0.1.1 - py27_{0}.tar.bz24.UploadpackagestoAnaconda$
anaconda login anaconda upload linux-32/your$_{p}ackage.tar.bz2anacondauploadlinux-64/your_{p}ackage.tar.bz2anacondauploadwin-32/your_{p}ackage.tar.bz2anacondauploadwin-64/your_{p}ackage.tar.bz2CreatePrivateModuleStep1 : Makeyourmodule11.1pipinstallcookiecutter$
1.2 cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git
1.3 Fill all necessary information
full$_{n}ame[AudreyRoyGreenfeld] : email[aroy@alum.mit.edu] : github_{u}sername[audreyr] : project_{n}ame[PythonBoilerplate] : project_{s}lug[] : project_{s}hort_{d}escription :$

$release_date[] : pypi_username[] : year[2016] : version[0.1.0] : use_pypi_deployment_with_travis[y] :$
$Step2 : BuildyourmoduleChangeyourMANIFEST.in$
recursive-include project$_folder * Buildyourmodulewithsetup.py$
cd your$_project_folder$
build local python setup.py build > It will create a new folder in $> PYTHON_HOME/Lib/sites-$
$packages/your_project_name - 0.1.0 - py2.7.egg$
build distribution python setup.py sdist > It will create a zip file in $PROJECT_FOLDER/distStep3 :$
$UsageyourmoduleInthesamemachine$
import your$_project_nameInothermachine$
Python: Build Install Local Package with Conda Here is a step by step tutorial
about building a local module package install it from a custom channel 1
Step 1: Make a setup folder for your package with cookkiecutter on terminal:
mkdir build cd build pip install cookiecutter cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git
Fill all necessary information
full$_name[AudreyRoyGreenfeld] : email[aroy@alum.mit.edu] : github_username[audreyr] :$
$project_name[PythonBoilerplate] : project_slug[] : project_short_description :$
$release_date[] : pypi_username[] : year[2016] : version[0.1.0] : use_pypi_deployment_with_travis[y] :$
$Itwillcreateadirectory$
|- LICENSE |- README.md |- TODO.md |- docs | |– conf.py | |– generated | |–
index.rst | |– installation.rst | |– modules.rst | |– quickstart.rst | |– sandman.rst |-
requirements.txt |- your$_package||--_{init.py||--your_package.py||--test||--models.py||--test_your_package.py|-setup.pyCopyyourr}$
Add this line to MANIFEST.in
recursive-include project$_folder*Step2 : Buildcondapackagemkdircondacdcondamkdirchannelgitclonehttps :$
$//github.com/hunguyen1702/condaBuildLocalTemplate.gitmvcondaBuildLocalTemplateyour_package_nam$
$rf.gitREADME.mdEditthefilemeta.yamlwiththeinstructioninsideitcd..condabuildyour_package_nameStep$
$Createcustomchannelandinstallfromlocalpackage Createachanneldirectory$
cd channel Convert your$_packageyou'vebuilttoallplatform$
conda convert –platform all /anaconda/conda-bld/linux-64/your$_package_0.1.0-$
$py27_0.tar.bz2andthiswillcreate :$
channel/ linux-64/ package-1.0-0.tar.bz2 linux-32/ package-1.0-0.tar.bz2 osx-64/ package-1.0-0.tar.bz2 win-64/ package-1.0-0.tar.bz2 win-32/ package-1.0-0.tar.bz2 Register your package to your new channel
cd .. conda index channel/linux-64 channel/osx-64 channel/win-64 Veriy your new channel
conda search -c file://path/to/channel/ –override-channels If you see your$_package'sappearance, soit'sworked$
After that if you want to install that package from local, run this command:
conda install –use-local your$_package$
and when you want to create environment with local package from file, you just have export environment to .yml file and add this channels section before the dependencies section:
channels: - file://path/to/your/channel/

2.22 Production

Production with docker Base Image: magizbox/conda2.7/
Docker Folder
your$_app/appconfigmain.pyDockerfilerun.shDockerfile$
FROM magizbox/conda2.7:4.0

ADD ./app /app ADD ./run.sh /run.sh
RUN conda env create -f environment.yml run.sh
source activate your$_environment$
cd /app
python main.py Compose
service: build: ./service-app command: 'bash run.sh' Note: an other python conda with lower version (such as 3.5), will occur error when install requests package

2.23 Quản lý gói với Anaconda

Cài đặt package tại một branch của một project trên github

```
> pip install git+https://github.com/tangentlabs/django-oscar-paypal
    ↪ .git@issue/34/oscar-0.6#egg=django-oscar-paypal
```

Trích xuất danh sách package

```
> pip freeze > requirements.txt
```

Chạy ipython trong environment anaconda
Chạy đống lệnh này

```
    conda install nb_conda
    source activate my_env
    python -m IPython kernelspec install-self --user
    ipython notebook
```

Interactive programming với ipython
Trích xuất ipython ra slide (không hiểu sao default '–to slides' không work nữa, lại phải thêm tham số '–reveal-prefix' [1]

```
jupyter nbconvert "file.ipynb"
  --to slides
  --reveal-prefix "https://cdnjs.cloudflare.com/ajax/libs/reveal.js
      ↪ /3.1.0"
```

Tham khảo thêm
* https://stackoverflow.com/questions/37085665/in-which-conda-environment-is-jupyter-executing * https://github.com/jupyter/notebook/issues/541issuecomment-146387578 * https://stackoverflow.com/a/20101940/772391
python 3.4 hay 3.5
Có lẽ 3.5 là lựa chọn tốt hơn (phải có của tensorflow, pytorch, hỗ trợ mock)
Quản lý môi trường phát triển với conda
Chạy lệnh 'remove' để xóa một môi trường

```
conda remove --name flowers --all
```

2.24 Test với python

Sử dụng những loại test nào?
Hiện tại mình đang viết unittest với default class của python là Unittest. Thực ra toàn sử dụng 'assertEqual' là chính!

Ngoài ra mình cũng đang sử dụng tox để chạy test trên nhiều phiên bản python (python 2.7, 3.5). Điều hay của tox là mình có thể thiết kế toàn bộ cài đặt project và các dependencies package trong file ‘tox.ini‘
Chạy test trên nhiều phiên bản python với tox
Pycharm hỗ trợ debug tox (quá tuyệt!), chỉ với thao tác đơn giản là nhấn chuột phải vào file tox.ini của project.

2.25 Xây dựng docs với readthedocs và sphinx

20/12/2017: Tự nhiên hôm nay tất cả các class có khai báo kế thừa ở project languageflow không thể index được. Vãi thật. Làm thằng đệ không biết đâu mà build model.
Thử build lại chục lần, thay đổi file conf.py và package_reference.rst chán chê không được. Giả thiết đầu tiên là do hai nguyên nhân (1) docstring ghi sai, (2) nội dung trong package_reference.rst bị sai. Sửa chán chê cũng vẫn thế, thử checkout các commit của git. Không hoạt động!
Mất khoảng vài tiếng mới để ý thằng readthedocs có phần log cho từng build một. Lần mò vào build gần nhất và build (mình nhớ là) thành công cách đây 2 ngày
Log build gần nhất

```
Running Sphinx v1.6.5
making output directory...
loading translations [en]... done
loading intersphinx inventory from https://docs.python.org/objects.inv
    ↪ ...
intersphinx inventory has moved: https://docs.python.org/objects.inv ->
    ↪ https://docs.python.org/2/objects.inv
loading intersphinx inventory from http://docs.scipy.org/doc/numpy/
    ↪ objects.inv...
intersphinx inventory has moved: http://docs.scipy.org/doc/numpy/
    ↪ objects.inv -> https://docs.scipy.org/doc/numpy/objects.inv
building [mo]: targets for 0 po files that are out of date
building [readthedocsdirhtml]: targets for 8 source files that are out
    ↪ of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [ 12%] authors
reading sources... [ 25%] contributing
reading sources... [ 37%] history
reading sources... [ 50%] index
reading sources... [ 62%] installation
reading sources... [ 75%] package_reference
reading sources... [ 87%] readme
reading sources... [100%] usage

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 12%] authors
writing output... [ 25%] contributing
writing output... [ 37%] history
writing output... [ 50%] index
writing output... [ 62%] installation
writing output... [ 75%] package_reference
writing output... [ 87%] readme
writing output... [100%] usage
```

Log build hồi trước

```
Running Sphinx v1.5.6
making output directory...
loading translations [en]... done
loading intersphinx inventory from https://docs.python.org/objects.inv
    ↪ ...
intersphinx inventory has moved: https://docs.python.org/objects.inv ->
    ↪ https://docs.python.org/2/objects.inv
loading intersphinx inventory from http://docs.scipy.org/doc/numpy/
    ↪ objects.inv...
intersphinx inventory has moved: http://docs.scipy.org/doc/numpy/
    ↪ objects.inv -> https://docs.scipy.org/doc/numpy/objects.inv
building [mo]: targets for 0 po files that are out of date
building [readthedocs]: targets for 8 source files that are out of date
updating environment: 8 added, 0 changed, 0 removed
reading sources ... [ 12%] authors
reading sources ... [ 25%] contributing
reading sources ... [ 37%] history
reading sources ... [ 50%] index
reading sources ... [ 62%] installation
reading sources ... [ 75%] package_reference
reading sources ... [ 87%] readme
reading sources ... [100%] usage

/home/docs/checkouts/readthedocs.org/user_builds/languageflow/
    ↪ checkouts/develop/languageflow/transformer/count.py:docstring
    ↪ of languageflow.transformer.count.CountVectorizer:106:
    ↪ WARNING: Definition list ends without a blank line; unexpected
    ↪ unindent.
/home/docs/checkouts/readthedocs.org/user_builds/languageflow/
    ↪ checkouts/develop/languageflow/transformer/tfidf.py:docstring of
    ↪ languageflow.transformer.tfidf.TfidfVectorizer:113: WARNING:
    ↪ Definition list ends without a blank line; unexpected unindent.
../README.rst:7: WARNING: nonlocal image URI found: https://img.
    ↪ shields.io/badge/latest-1.1.6-brightgreen.svg
looking for now-outdated files... none found
pickling environment... done
checking consistency ... done
preparing documents... done
writing output... [ 12%] authors
writing output... [ 25%] contributing
writing output... [ 37%] history
writing output... [ 50%] index
writing output... [ 62%] installation
writing output... [ 75%] package_reference
writing output... [ 87%] readme
writing output... [100%] usage
```

Đập vào mắt là sự khác biệt giữa documentation type
Lỗi

```
building [readthedocsdirhtml]: targets for 8 source files that are out
    ↪ of date
```

Chạy

```
building [readthedocs]: targets for 8 source files that are out of date
```

Hí ha hí hửng. Chắc trong cơn bất loạn sửa lại settings đây mà. Sửa lại nó trong phần Settings (Admin gt; Settings gt; Documentation type)
![](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-09-54-23.png)
Khi chạy nó đã cho ra log đúng

```
building [readthedocsdirhtml]: targets for 8 source files that are out
    ↪ of date
```

Nhưng vẫn lỗi. Vãi!!! Sau khoảng 20 phút tiếp tục bấn loạn, chửi bới readthedocs các kiểu. Thì để ý dòng này
Lỗi

```
Running Sphinx v1.6.5
```

Chạy

```
Running Sphinx v1.5.6
```

Ngay dòng đầu tiên mà không để ý, ngu thật. Aha, Hóa ra là thằng readthedocs nó tự động update phiên bản sphinx lên 1.6.5. Mình là mình chúa ghét thay đổi phiên bản (code đã mệt rồi, lại còn phải tương thích với nhiều phiên bản nữa thì ăn c** à). Đầu tiên search với Pycharm thấy dòng này trong ‘conf.py‘

```
# If your documentation needs a minimal Sphinx version, state it here.
# needs_sphinx = '1.0'
```

Đổi thành

```
# If your documentation needs a minimal Sphinx version, state it here.
needs_sphinx = '1.5.6'
```

Vẫn vậy (holy sh*t). Thử sâu một tẹo (thực sự là rất nhiều tẹo). Thấy cái này trong trang Settings
![](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-10-01-39.png)
Ở há. Thằng đần này cho phép trỏ đường dẫn tới một file trong project để cấu hình dependency. Haha. Tạo thêm một file ‘requirements‘ trong thư mục ‘docs‘ với nội dung

```
sphinx==1.5.6
```

Sau đó cấu hình nó trên giao diện web của readthedocs
![](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-10-04-49.png)
Build thử. Build thử thôi. Cảm giác đúng lắm rồi đấy. Và... nó chạy. Ahihi
![](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-10-06-32.png)
Kinh nghiệm
* Khi không biết làm gì, hãy làm 3 việc. Đọc LOG. Phân tích LOG. Và cố gắng để LOG thay đổi theo ý mình.
PS: Trong quá trình này, cũng không thèm build thằng PDF với Epub nữa. Tiết kiệm được bao nhiêu thời gian.

2.26 Pycharm Pycharm

01/2018: Pycharm là trình duyệt ưa thích của mình trong suốt 3 năm vừa rồi. Hôm nay tự nhiên lại gặp lỗi không tự nhận unittest, không resolve được package import bởi relative path. Vụ không tự nhận unittest sửa bằng cách xóa file .idea là xong. Còn vụ không resolve được package import bởi relative path thì vẫn chịu rồi. Nhìn code cứ đỏ lòm khó chịu thật.

2.27 Vì sao lại code python?

01/11/2017 Thích python vì nó quá đơn giản (và quá đẹp).
[^{1}] : $https://github.com/jupyter/nbconvert/issues/91issuecomment-283736634$

Chương 3

C++

C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, servers (e.g. e-commerce, web search or SQL servers), and performance-critical applications (e.g. telephone switches or space probes). C++ is a compiled language, with implementations of it available on many platforms and provided by various organizations, including the Free Software Foundation (FSF's GCC), LLVM, Microsoft, Intel and IBM.
View online http://magizbox.com/training/cpp/site/

3.1 Get Started

What do I need to start with CLion? In general to develop in C/C++ with CLion you need:
CMake, 2.8.11+ (Check JetBrains guide for updates) GCC/G++/Clang (Linux) or MinGW 3. or MinGW—w64 3.-4. or Cygwin 1.7.32 (minimum required) up to 2.0. (Windows) Downloading and Installing CMake Downloading and installing CMake is pretty simple, just go to the website, download and install by following the recommended guide there or the on Desktop Wizard.
Download and install file cmake-3.9.0-win64-x65.msi > cmake Usage
cmake [options] <path-to-source> cmake [options] <path-to-existing-build>
Specify a source directory to (re-)generate a build system for it in the current working directory. Specify an existing build directory to re-generate its build system.
Run 'cmake –help' for more information. Downloading and Getting Cygwin Cygwin is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows
Download file setup-x86$_6$4*.exefromthewebsitehttps : //cygwin.com/install.html*
Install setup-x86$_6$4*.exefile*

This is the root directory where Cygwin will be located, usually the recommended C: works
Choose where to install LOCAL DOWNLOAD PACKAGES: This is not the same as root directory, but rather where packages (ie. extra C libraries and tools) you download using Cygwin will be located
Follow the recommended instructions until you get to packages screen:
Once you get to the packages screen, this is where you customize what libraries or tools you will install. From here on I followed the above guide but here's the gist:
From this window, choose the Cygwin applications to install. For our purposes, you will select certain GNU C/C++ packages.
Click the + sign next to the Devel category to expand it.
You will see a long list of possible packages that can be downloaded. Scroll the list to see more packages.
Pick each of the following packages by clicking its corresponding "Skip" marker. gcc-core: C compiler subpackage gcc-g++: C++ subpackage libgcc1: C runtime library gdb: The GNU Debugger make: The GNU version of the 'make' utility libmpfr4 : A library for multiple-precision floating-point arithmetic with exact rounding Download and install CLion Download file CLion-2017.2.exe from website https://www.jetbrains.com/clion/download/section=windows
Config environment File > Settings... > Build, Execution, Deployment
Choose Cygwin home: C:64 Choose CMake executable: Bundled CMake 3.8.2
Run your first C++ program with CLion

3.2 Basic Syntax

C/C++ Hello World include <iostream> using namespace std;
int main() cout « "hello world"; Convention Naming variable$_{n}ame_{l}ike_{t}hisclass_{d}ata_{m}emeber_{n}ame_{l}ike_{t}his_{k}C$
$//GargantuanTableIterator * iter = table- > NewIterator(); //for(iter- >$
$Seek("foo"); !iter- > done(); iter- > Next())//process(iter- > key(), iter- > value()); ////deleteiter; cl$
$Usea"*"hereforconcatenationoperator.//TODO(Zeke)changethistouserelations.$

3.3 Cấu trúc dữ liệu

Data Structure Number C++ offer the programmer a rich assortment of built-in as well as user defined data types. Following table lists down seven basic C++ data types:
Boolean - bool Character - char Integer - int Floating point - float Double floating point - double Valueless - void Wide character - wchar$_{t}Severalofthebasictypescanbemodifiedusingoneorn$
$signed, unsigned, short, long$
Following is the example, which will produce correct size of various data types on your computer.
include <iostream> using namespace std;
int main() cout « "Size of char : " « sizeof(char) « endl; cout « "Size of int : " « sizeof(int) « endl; cout « "Size of short int : " « sizeof(short int) « endl; cout « "Size of long int : " « sizeof(long int) « endl; cout « "Size of float : " « sizeof(float) « endl; cout « "Size of double : " « sizeof(double) « endl; cout « "Size of wchar$_t$: $" << sizeof(wchar_t) << endl; return0; StringStringBasic$

include <iostream> include <string> using namespace std ;
// assign a string string s1 = "www.java2s.com"; cout « s1;
// input a string string s2; cin » s2;
// concatenate two strings string $s_c = s1 + s2$;
// compare strings s1 == s2; Collection Pointer A pointer is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it.
The general form of a pointer variable declaration is:
type *variable$_name; //exampleint{*}ip; //pointertoanintegerdouble{*}dp; //pointertoadoublefloat*$
$fp; //pointertoafloatchar * ch; //pointertocharacterPointerLab$
include <iostream> using namespace std;
/* * Look at these lines */ int* a; a = new int[3]; a[0] = 10; a[1] = 2; cout « "Address of pointer a: a = " « a « endl; cout « "Value of pointer a: a = " « a « endl « endl; cout « "Address of a[0]: a[0] = " « a[0] « endl; cout « "Value of a[0]: a[0] = " « a[0] « endl; cout « "Value of a[0]: *a = " « *a « endl « endl; cout « "Address of a[1]: a[1] = " « a[1] « endl; cout « "Value of a[1]: a[1] = " « a[1] « endl; cout « "Value of a[1]: *(a+1)= " « *(a+1)« endl « endl; cout « "Address of a[2]: a[2] = " « a[2] « endl; cout « "Value of a[2]: a[2] = " « a[2] « endl; cout « "Value of a[2]: *(a+2)= " « *(a+2)« endl « endl; Result:
Address of pointer a: a = 008FF770 Value of pointer a: a = 00C66ED0
Address of a[0]: a[0] = 00C66ED0 Value of a[0]: a[0] = 10 Value of a[0]: *a = 10
Address of a[1]: a[1] = 00C66ED4 Value of a[1]: a[1] = 2 Value of a[1]: *(a+1)= 2
Address of a[2]: a[2] = 00C66ED8 Value of a[2]: a[2] = -842150451 Value of a[2]: *(a+2)= -842150451 Stack, Queue, Linked List, Array, Deque, List, Map, Set
Datetime The C++ standard library does not provide a proper date type. C++ inherits the structs and functions for date and time manipulation from C. To access date and time related functions and structures, you would need to include header file in your C++ program.
There are four time-related types: clock$_t, time_t, size_t, andtm.Thetypesclock_t, size_tandtime_tarecapableofrepr$
The structure type tm holds the date and time in the form of a C structure having the following elements:
struct tm int tm$_sec; //secondsofminutesfrom0to61inttm_min; //minutesofhourfrom0to59inttm_hour; //hc$
Consider you want to retrieve the current system date and time, either as a local time or as a Coordinated Universal Time (UTC). Following is the example to achieve the same:
include <iostream> include <ctime>
using namespace std;
int main( ) // current date/time based on current system time$_tnow = time(0)$;
// convert now to string form char* dt = ctime(now);
cout « "The local date and time is: " « dt « endl;
// convert now to tm struct for UTC tm *gmtm = gmtime(now); dt = asctime(gmtm); cout « "The UTC date and time is:"« dt « endl; When the above code is compiled and executed, it produces the following result:
The local date and time is: Sat Jan 8 20:07:41 2011
The UTC date and time is:Sun Jan 9 03:07:41 2011

3.4 Lập trình hướng đối tượng

Object Oriented Programming Classes and Objects include <iostream> using namespace std;
class Pacman
private: int x; int y; public: Pacman(int x, int y); void show(); ;
Pacman::Pacman(int x, int y) this->x = x; this->y = y;
void Pacman::show() std::cout « "(" « this->x « ", " « this->y « ")";
int main() // your code goes here Pacman p = Pacman(2, 3); p.show(); return 0; Template Function Template
include <iostream> include <string>
using namespace std;
template <typename T>
T Max(T a, T b) return a < b ? b : a;
int main()
int i = 39; int j = 20; cout « Max(i, j) « endl;
double f1 = 13.5; double f2 = 20.7; cout « Max(f1, f2) « endl;
string s1 = "Hello"; string s2 = "World"; cout « Max(s1, s2) « endl;
double n1 = 20.3; float n2 = 20.4; // it will show an error // Error: no instance of function template "Max" matches the argument list // arguments types are: (double, float) cout « Max(n1, n2) « endl; return 0;

3.5 Cơ sở dữ liệu

Database Sqlite with Visual Studio 2013 Step 1: Create new project 1.1 Create a new C++ Win32 Console application.
Step 2: Download Sqlite DLL
2.1. Download the native SQLite DLL from: http://sqlite.org/sqlite-dll-win32-x86-3070400.zip 2.2. Unzip the DLL and DEF files and place the contents in your project's source folder (an easy way to find this is to right click on the tab and click the "Open Containing Folder" menu item.
Step 3: Build LIB file
3.1. Open a "Developer Command Prompt" and navigate to your source folder. (If you can't find this tool, follow this post in stackoverflow Where is Developer Command Prompt for VS2013? to create it) 3.2. Create an import library using the following command line: LIB /DEF:sqlite3.def
Step 4: Add Dependencies
4.1. Add the library (i.e. sqlite3.lib) to your Project Properties -> Configuration Properties -> Linker -> Input -> Additional Dependencies. 4.2. Download http://sqlite.org/sqlite-amalgamation-3070400.zip 4.3. Unzip the sqlite3.h header file and place into your source directory. 4.4. Include the the sqlite3.h header file in your source code. 4.5. You will need to include the sqlite3.dll in the same directory as your program (or in a System Folder).
Step 5: Run test code
include "stdafx.h" include <ios> include <iostream> include "sqlite3.h"
using namespace std;
int $_tmain(intargc,_TCHAR * argv[])intrc; char * error;$
// Open Database cout « "Opening MyDb.db ..." « endl; sqlite3 *db; rc = sqlite3$_open("MyDb.db", db); if(rc)cerr << "ErroropeningSQLite3database : " << sqlite3_errmsg(db) << e$

// Execute SQL cout « "Creating MyTable ..." « endl; const char *sqlCreateTable = "CREATE TABLE MyTable (id INTEGER PRIMARY KEY, value STRING);"; rc = sqlite3$_exec(db, sqlCreateTable, NULL, NULL, error); if(rc)cerr << "ErrorexecutingSQ$
// Execute SQL cout « "Inserting a value into MyTable ..." « endl; const char *sqlInsert = "INSERT INTO MyTable VALUES(NULL, 'A Value');"; rc = sqlite3$_exec(db, sqlInsert, NULL, NULL, error); if(rc)cerr << "ErrorexecutingSQLite3statement : " <<$
// Display MyTable cout « "Retrieving values in MyTable ..." « endl; const char *sqlSelect = "SELECT * FROM MyTable;"; char **results = NULL; int rows, columns; sqlite3$_get_table(db, sqlSelect, results, rows, columns, error); if(rc)cerr << "ErrorexecutingSQLite$
// Display Cell Value cout.width(12); cout.setf(ios::left); cout « results[cellPosition] « " ";
// End Line cout « endl;
// Display Separator For Header if (0 == rowCtr) for (int colCtr = 0; colCtr < columns; ++colCtr) cout.width(12); cout.setf(ios::left); cout « " "; cout « endl; sqlite3$_free_table(results);$
// Close Database cout « "Closing MyDb.db ..." « endl; sqlite3$_close(db); cout << "ClosedMyDb.db" << endl << endl;$
// Wait For User To Close Program cout « "Please press any key to exit the program ..." « endl; cin.get();
return 0;

3.6 Testing

Create Unit Test in Visual Studio 2013 Step 1. Create TDDLab Solution 1.1 Open Visual Studio 2013
1.2 File -> New Project... ->
Click Visual C++ -> Win32
Choose Win32 Console Application
Fill to Name input text: TDDLab
Click OK -> Next
1.3 In project settings, remove options:
Precompiled Header Securirty Develoment Lifecyde(SQL) check 1.4 Click Finish
Step 2. Create Counter Class 2.1 Right-click TDDLab -> Add -> Class...
2.2 Choose Visual C++ -> C++ Class -> Add
2.3 Fill in Class name box Counter -> Finish
2.4 In Counter.h file, add this below function
int add(int a, int b); 2.5 In Counter.cpp, add this below function
int Counter::add(int a, int b) return a+b; Your Counter class should look like this
Step 3. Create TDDLabTest Project 3.1 Right-click Solution 'TDDLab' -> Add -> New Project...
3.2 Choose Visual C++ -> Test
3.3 Choose Native Unit Test Project
3.4 Fill to Name input text: TDDLabTest
Step 4. Write unit test 4.1 In unittest1.cpp, add header of Counter class
include "../TDDLab/Counter.h" 4.2 In TEST$_METHODfunction$
Counter counter; Assert::AreEqual(2, counter.add(1, 1)); 4.3 Click TEST in menu bar -> Run -> 'All Test (Ctrl + R, A)

Step 5. Fix error LNK 2019: unresolved external symbol 5.1 Change Configuration Type of TDDLab project
Right click TDDLab project -> Properties General -> Configuration Type -> Static library (.lib) -> OK 5.2 Add Reference to TDDLabTest project
Right click TDDLabTest solution -> Properties -> Common Properties -> Add New Reference Choose TDDLab -> OK -> OK Step 6. Run Tests Click TEST in menu bar -> Run -> 'All Test (Ctrl + R, A)
Test should be passed.

3.7 IDE Debugging

Visual Studio 2013 Install Extension
VsVim
googletest guide
Folder Structure with VS 2013
solution README.md |project1 | file011.txt | file012.txt | |project2 | file011.txt | file012.txt | Auto Format
Ctrl + K, Ctrl + D Git in Visual Studio
https://git-scm.com/book/en/v2/Git-in-Other-Environments-Git-in-Visual-Studio
Online IDE codechef ide

Chương 4

Javascript

View online http://magizbox.com/training/java/site/
What is Javascript? JavaScript is a high-level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three core technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern Web browsers without plugins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

4.1 Installation

Google Chrome Pycharm

4.2 IDE

Google Chrome Developer Tools
The Chrome Developer Tools (DevTools for short), are a set of web authoring and debugging tools built into Google Chrome. The DevTools provide web developers deep access into the internals of the browser and their web application. Use the DevTools to efficiently track down layout issues, set JavaScript breakpoints, and get insights for code optimization.

4.3 Basic Syntax

1. Code Formatting Indent with 2 spaces
// Object initializer. var inset = top: 10, right: 20, bottom: 15, left: 12 ;
// Array initializer. this.rows$_=['"Slartibartfast" < fjordmaster@magrathea.com >'$
$,'"ZaphodBeeblebrox" < theprez@universe.gov >','"FordPrefect" < ford@theguide.com >'$

$,' "ArthurDent" < has.no.tea@gmail.com >',' "MarvintheParanoidAndroid" < marv@googlemail.com >',' the.mice@magrathea.com'$];
// Used in a method call. goog.dom.createDom(goog.dom.TagName.DIV, id: 'foo', className: 'some-css-class', style: 'display:none' , 'Hello, world!'); 2. Naming functionNamesLikeThis variableNamesLikeThis ClassNamesLikeThis EnumNamesLikeThis methodNamesLikeThis CONSTANT$_VALUES_LIKE_THISfoo.namespaceNamesLikeThis.$
3.1 Class Comment /** * Class making something fun and easy. * @param string arg1 An argument that makes this more interesting. * @param Array.<number> arg2 List of numbers to be processed. * @constructor * @extends goog.Disposable */ project.MyClass = function(arg1, arg2) // ... ; goog.inherits(project.MyClass, goog.Disposable); 3.2 Method Comment /** * Operates on an instance of MyClass and returns something. * @param project.MyClass obj Instance of MyClass which leads to a long * comment that needs to be wrapped to two lines. * @return boolean Whether something occurred. */ function PR$_someMethod(obj)//...4.ExpressionandStatem$
22 "this is an epression" (5 > 6) ? false : true Statements The Simplest kind of stagement is an expression with a semi colon
!false; 5 + 6; 5. Loop and iteration while var number = 0; while (number <= 12) console.log(number); number = number + 2; do..while do var yourName = prompt("Who are you?"); while (!yourName); console.log(yourName); for for (var i = 0; i < 10; i++) console.log(i); 6. Function 6.1 Defining a Function var square = function(x) return x * x; ; square(5); 6.2 Scope Scope is the area where contains all variable or function are living. Scope has some rules: Child Scope can access all variable and function in parent Scope. (E.g: Local Scope can access Global Scope) function saveName(firstName) var temp = "temp"; function capitalizeName() temp = temp + " here"; return firstName.toUpperCase(); var capitalized = capitalizeName(); return capitalized; alert(saveName("Robert")); But parent Scope can access variable and function inside children scope (E.g: Global Scope cannot acces to local Scope) function talkDirty () var saying = "Oh, you little VB lover, you"; return alert(saying); alert(saying); //->Error 6.3 Call Stack The storage where computer stores context is called CALL STACK. // CALL STACK function greet(who) console.log("Hello " + who); ask("How are you?"); console.log("I'm fine"); ;
function ask(question) console.log("well, " + question); ;
greet("Harry"); console.log("Bye"); Out of Call Stack
function chicken() return egg();
function egg() return chicken(); console.log(chicken() + " came first"); 6.4. Optional Argument We can pass too many or too few arguments to the function without any SyntaxError. If we pass too much arguments, the extra ones are ignored If we pass to few arguments, the missing ones get value undefined function power(base, exponent) if (exponent == undefined) exponent = 2; var result = 1; for (var count = 0; count < exponent; count++) result = result * base; return result; console.log(power(4)); console.log(power(4,3)); upside: flexible downside: hard to control the error
6.5 Closure Look at this example:
function sayHello(name) var text = 'Hello' + name; var say = function() console.log(text); return say; var say2 = sayHello("ahaha"); say2(); if in C program, does say2() work? The answer is nope! Because in C program, when a function returns, the Stack-flame will be destroyed, and all the local variable such as text will undefinded. So, when say2() is called, there is no text anymore, and the error, will be shown! But, in JavaScript, This code works!! Because, it

provides for us an Object called Closure! Closure is borned when we define a function in another function, it keep all the live local variable. So, when say2() is called, the closure will give all the value of local variable outside it, and text will be identity.!
var globalVariable = 10; function func() var name = "xxx"; function getName() return name; function speak() var sound = "alo"; function scream() console.log(globalVariable); console.log(name); return "aaaaaaaaaa!"; function talk() var voice = getName() + " speak " + sound; console.log(voice); return voice; scream(); talk(); speak(); func(); 6.6. Recursion Recursion is function can call itself, as long as it is not overflow
function power(base, exponent) if (exponent == 0) return 1; else return base * power(base, exponent -1); console.log(power(2,3));
function FindSolution(target) function Find(start, history) if (start == target) return history; else if (start > target) return null; else return Find(start + 5, "(" + history + " + 5 ") || Find(start * 3, "(" + history + " * 3)"); return Find(1, "1"); console.log(FindSolution(25)); 6.7. Arguments object The arguments object contains all parameters you pass to a function.
function argumentCounter() console.log("you gave me", arguments.length, "argument."); argumentCounter("Straw man", "Tautology", "Ad hominem"); 6.8. Higher-Order Function Filter array var ancestry = JSON.parse(ANCESTRY$_FILE); console.log(ancestry.leng$
function filter(array, test) var passed = []; for (var i = 0; i < array.length; i++) if (test(array[i])) passed.push(array[i]); return passed; console.log(filter(ancestry, function(person) return person.born > 1900 person.born < 1925;));
TRANSFORMING WITH A MAP function map(array, transform) var mapped = []; for (var i = 0; i < array.length; i++) mapped.push(transform(array[i])); return mapped;
var overNinety = ancestry.filter(function(person) return person.died - person.born > 90;); console.log(map(overNinety, function(person) return person.name;));
REDUCE function reduce(array, combine, start) var current = start; for (var i = 0; i < array.length; i++) current = combine(current, array[i]); return current; console.log(reduce([1, 2, 3, 4], function(a, b) return a + b; , 0)); Problem: using map and reduce, transform [1,2,3,4] to [1,2],[3,4]
var a = [1, 2, 3, 4] a = $_.map(a, function(i) if(i return [[], [i]] else return [[i], []]); a = _.$
$reduce(a, function(x, y) return [x[0].concat(y[0]), x[1].concat(y[1])])$
BINDING FUNCTION var theSet = ["Carel Haverbeke", "Maria van Brussel", "Donald Duck"]; function isInSet(set, person) return set.indexOf(person.name) > -1;
console.log(ancestry.filter(function(person) return isInSet(theSet, person);)); console.log(ancestry.filter(isInSet.bind(null, theSet))); What's the cleanest way to write a multiline string in JavaScript? [duplicate]
Google JavaScript Style Guide

4.4 Data Structure

4.4.1 Number

Some example of number: 10, 1.234, 1.99e9, NaN, Infinity, -Infinity

console.log(2.99e9); console.log(0 /0); console.log(1 /0); console.log(-1 /0); Automatic Conversion
console.log(8 * null); // -> 0 console.log("5" - 1); // -> 4 console.log("5" + 1); //-> 51 console.log(false == 0) //-> true

4.4.2 String

sprintf In index.html
<script src="cdnjs.cloudflare.com/ajax/libs/sprintf/1.0.3/sprintf.js"/>
In script.js
// arguments sprintf(" hello sprintf
// object var user = name: "Dolly" sprintf("Hello Hello Dolly
// array of object var users = [name: "Dolly", name: "Molly"] sprintf("Hello Hello Dolly and Molly Multiline String str = " line 1 line 2 line 3"; Regular Expression in JavaScript This lab is based on Chapter9: EloquentJavaScript
Creating a regular expression There are 2 ways:
var re1 = new RegExp("abc"); var re2 = /abc/ there are some special characters such as question mark, or plus sign. If you want to use them, you have to use backslash. Like this:
var eighteen = /eighteen/; var question = /question/; Testing for match Regular Express has a number of method. Simplest is test
console.log(/abc/.test("abcd")); console.log(/abc/.test("abxde")); Matching a set of character []: Put a set of characters between 2 square bracket
console.log(/[0123456789]/.test("1245")); console.log(/[0-9]/.test("1"); console.log(/[0-9]/.test("acd"); console.log(/[0-9]/.test("aaascacas1")); There are some special character: Ạny digit character (Like [0-9])
var datetime = /-̣-̣:̣/; console.log(datetime.test("16-06-2016 14:09")); console.log(dateTime.test("30-jan-2003 15:20")); An alphanumeric character (“word character”)
var word = //; console.log(word.test("@@")); Any whitespace character (space, tab, newline, and similar)
var space = /̣+abc/; console.log(space.test("1. abd")); console.log(space.test("1. abd")); console.log(space.test("1.abd")); A character that is not a digit
var notDigit = //; console.log(notDigit.test("ww")); console.log(notDigit.test("1a")); console.log(notDigit.test("1124")); A nonalphanumeric character
var nonAlphanumbericChar = //; console.log(nonAlphanumbericChar.test("abc12231")); console.log(nonAlphanumbericChar.test("!@§A nonwhitespace character
var nonWhiteSpace = /§/; console.log(nonWhiteSpace.test("abc123")); console.log(nonWhiteSpace.test("1. abcd")); console.log(nonWhiteSpace.test(" "));
"." Any character ẹxcept for newline
var anyThing = /...̣/; console.log(anyThing.test("abc.")); console.log(anyThing.test("acbacd.")); console.log(anyThing.test("acba")); "$"Usingcaretcharactertomatchanyexcepttheones$
var notBinary = /[^{0}1]/;$console.log(notBinary.test($"01101011100"$));console.log(notBinary.test($"010210100
$"Matchoneormore" * "Matchzeroormore$
console.log(/+̣/.test(1234)); console.log(/+̣/.test());
console.log(/*̣/.test(1234)); console.log(/*̣/.test()) "?" Question mark test a character exist or not is still oke
var ball = /bal?l/; console.log(ball.test("ball")); console.log(ball.test("bal"));
a,b the character before exist from a to b times. Check datetime:

var datetime = /1,2-1,2-4 1,2:1,2/; console.log(datetime.test("20-12-2015 14:09"));
var checkTimes = /waz3,5up/; console.log(checkTimes.test("wazzzzzup")); console.log(checkTimes.test("wazzzup")); console.log(checkTimes.test("wazup"));
Grouping Subexpressions () using prentheses to make whole group like one character
var cartoonCrying = /boo+(hoo+)+/i; //i to match all Captalize or normal text
console.log(cartoonCrying.test("Boohoooohoohooo")); console.log(cartoonCrying.test("boohoooohooOOO"))
Matches and group Test is a simplest method, and it only return true or false. exec (execute) is anther method in regex. It returns null if no match, and object if match.
var match = /+/.exec("one two 100"); console.log(match); console.log(match.input); console.log(match.index); if in the expression has a group subexpression, then it will return the text contain this subexpress, and the text match this subexpress:
var quotedText = /'([']*)'/;*console.log(quotedText.exec("shesaid'hello'"));andifthesubexpressionappearsonemoretimes,thenti*
console.log(/bad(ly)?/.exec("bad")); console.log(/()+/.exec("123")); The date type create new Date(). return the current time
var date = new Date(); console.log(new Date(2009, 11, 9); console.log(new Date(2009, 11, 9, 23, 59, 61)); <!–TimeStamp–> console.log(new Date(2009, 11, 9, 23, 59, 61).getTime()); console.log(new Date(1260378001000)); <!–getFullYear, getMonth,...–> var date = new Date(); console.log(date.getFullYear()); console.log(date.getMonth()); console.log(date.getDate()); console.log(date.getHours()); console.log(date.getMinutes()); console.log(date.getSeconds()); Word and string boundaries console.log(/cat/.test("concatenate")); console.log(/cat/.test("con123cat-129e0enate")); console.log(//.test("concatenate")); console.log(//.test("con123cat-129e0enate")); Choice patterm Only one in the list beween the "|" match
var animalCount = /+ (pig|cow|chicken)s?/; console.log(animalCount.test("15 pigs")); console.log(animalCount.test("15 pigchickens")); Replace Replace will find the first match and replace.if we want to replace all matches, using "g" behind the expresssion
console.log("papa".replace("p", "m")); console.log("Borobudur".replace(/[ou]/, "a")); console.log("Borobudur".replace(/[ou]/g, "a")); Replace can refer back to the matched, and using them
console.log("Le, Khanh, Hung, Bach".replace(/([]+), ([]+)/g, "12")); Greed
function stripComments(code) return code.replace(/.*|[]*/g,""); *console.log(stripComments("1+ / * 2 * /3"));//1 + 3console.log(stripComments("x = 10;//ten!"));//x =* 10; *console.log(stripComments("1/*a*/+/*b*/1"));//11SearchmethodSearchmethodreturnthefirstindex*
1*ifnotfound*
console.log(" word".search(/§/)); // 2 console.log(" ".search(/§/)); // -1 The last index property In the regular expression has a property is lastIndex. And when this Regex do some method, it will start from the lastIndex. And after doing something, the lastIndex will update to the behind the index of the match exec.
var pattern = /y/g; pattern.lastIndex = 3; //lastIndex update to 3 var match = pattern.exec("xyzzy"); //lastIndex update to 5 console.log(pattern.lastIndex); match = pattern.exec("xyzzyxxx"); //Not match any "y" from index 5 console.log(match.index); console.log(pattern.lastIndex); Looping Over the Line Applying the hepoloris of lastIndex, we can using while to do something like this:
var input = "A string with 3 numbers in it... 42 and 88."; var number = /(+)/g;

var match; while (match = number.exec(input)) console.log("Found", match[1], "at", match.index);

4.4.3 Collection

Some useful methods with array push and pop var a = [1,2,3,4]; console.log(a.pop(), a); console.log(a.push(3), a); shift and unshift console.log(a.shift(), a); console.log(a.unshift(1), a); indexOf and lastIndexOf var b = [1,2,3,4,2,3,1]; console.log(b.indexOf(1)); console.log(b.lastIndexOf(1)); slice console.log([0,1,2,3,4].slice(2,4)); console.log([0,1,2,3,4].slice(2)); concat var a = [1,2,3]; var b = [4,5,6]; a.concat(b); console.log(a);

4.4.4 Datetime

Current Time moment().format('MMMM Do YYYY, h:mm:ss a'); Moment.js

4.4.5 Boolean

Boolean has only 2 values: true and false
console.log("Abc" < "Abcd") // -> true console.log("abc" < "Abcd") // -> false console.log("123" == "123") // -> true console.log(NaN == NaN) // -> false what is the different?
console.log("5" == 5); console.log("5" === 5);

4.4.6 Object

Object Define an object var object = number: 10, string: "string", array: [1,2,3], object: a: 1, b: 2 Add new property to object object.newProperty = "value"; object['key'] = 'value'; delete property delete object.newProperty; Window object (global object) The Global scope is stored in an object which called window function test() var local = 10; console.log("local" in window); console.log(window.local); test(); var global = 10; console.log("global" in window); console.log(window.global);

4.5 OOP

1. Classes and Objects Constructor function Ball(position) this.position = position; this.display = function() console.log(this.position[0], ", ", this.position[1]);

ball = new Ball([2, 3]); ball.display(); 2. Inheritance Person = function (name, birthday, job) this.name = name; this.birthday = birthday; this.job = job; ; Person.prototype.display = function () console.log(this.name, ""); console.log(this.birthday, ""); console.log(this.job, ""); ;
Politician = function (name, birthday) Person.call(this, name, birthday, "Politician"); ; Politician.prototype = Object.create(Person.prototype); Politician.prototype.constructor = Politician;
var person1 = new Person("Barack Obama", "04/08/1961", "Politician"); var person2 = new Politician("David Cameron", "09/10/1966"); person1.display(); person2.display();
Object-Oriented Programming var rabbit = ; rabbit.speak = function(line) console.log("The rabit says:'" + line + "'"); ; rabbit.speak("I'm alive");

function speak(line) console.log("The "+ this.type + " rabbit says '" + line + "'");
var whiteRabbit = type: "white", speak: speak; var fatRabbit = type: "fat", speak: speak; whiteRabbit.speak("Oh my ears and whiskers, " + "how late it's getting!"); fatRabbit.speak("I could sure use a carrot right now");
// Prototype // Prototype is another object that is used as a fallback source of properties // When object request a property that it does not have, its prototype will be searched for the property var empty = ; console.log(empty.toString); console.log(empty.toString);
// Get prototype of an object 2 ways: console.log(Object.getPrototypeOf() == Object.prototype); console.log(Object.getPrototypeOf(Object.prototype));
// Using Object.create to create an object with an specific prototype var protoRabbit = speak: function(line) console.log("The " + this.type + " rabbit says '" + line + "'"); ;
var killerRabbit = Object.create(protoRabbit); killerRabbit.type = "Killer"; killerRabbit.speak("Skreeee!");
// Constructor function Rabbit(type) this.type = type; var killerRabbit = new Rabbit("Killer"); var blackRabbit = new Rabbit("black"); console.log(blackRabbit.type);
// using prototype to add a new method Rabbit.prototype.speak = function(line) console.log("The " + this.type + " rabit says '" + line + "'"); ; blackRabbit.speak("Doom...");
// OVERRIDING DERIVED PROPERTIES Rabbit.prototype.teeth = "small"; console.log(killerRabbit.teeth);
killerRabbit.teeth = "Long, sharp, and bloody"; console.log(killerRabbit.teeth); console.log(blackRabbit.teeth); console.log(Rabbit.prototype.teeth);
// PROTOTYPE INTERFERENCE // A prototype can be used at any time to add methods, properties // to all objects based on it Rabbit.prototype.dance = function () console.log("The " + this.type + " rabbit dances a jig"); ; killerRabbit.dance(); // but there is a problem: var map = ; function storePhi(event, phi) map[event] = phi;
storePhi("pizza", 0.069); storePhi("touched tree", -0.081); console.log(map);
Object.prototype.nonsense = "hi"; for (var name in map) console.log(name); console.log("nonsense" in map); console.log("toString" in map); delete Object.prototype.nonsense; // we can use Object.defineProperty to solve it Object.defineProperty(Object.prototype, "hiddenNonsense", enumerable: false, value: "hi");
for (var name in map) console.log(name); console.log(map.hiddenNonsense); // but there still has a problem console.log("toString" in map); console.log(map.hasOwnProperty("toString"));
// PROTOTYPE-LESS OBJECTS // if we only want to create an fresh object, without prototype then we tranform null to create var map = Object.create(null); map["pizza"] = 0.09; console.log("toString" in map); console.log("pizza" in map);
// POLYMORPHISM // laying out a table: example for polymorphism function rowHeights(rows) return rows.map(function(row) return row.reduce(function(max, cell) return Math.max(max, cell.minHeight()); , 0););
function colWidths(rows) return rows[0].map(function($,i) return rows.reduce(function(max, row) return Ma$
function drawTable(rows) var heights = rowHeights(rows); var widths = colWidths(rows);
function drawLine(blocks, lineNo) return blocks.map(function(block) return block[lineNo];).join(" ");

function drawRow(row, rowNum) var blocks = row.map(function(cell, colNum)
return cell.draw(widths[colNum], heights[rowNum]);); return blocks[0].map(function($,lineNo)returndrawLi$
return rows.map(drawRow).join("");
function repeat(string, times) var result = ""; for (var i = 0; i < times; i++)
result += string; return result;
function TextCell(text) this.text = text.split(""); TextCell.prototype.minWidth
= function() return this.text.reduce(function(width, line) return Math.max(width,
line.lenght); , 0); ; TextCell.prototype.minHeight = function() return this.text.length;
TextCell.prototype.minHeight = function() return this.text.lenght; TextCell.prototype.minHeight
= function() return this.text.length; TextCell.prototype.draw = function(width,
height) var result = []; for (var i = 0; i < height; i++) var line = this.text[i] ||
""; result.push(line + repeat(" ", width - line.length)); return result;
var rows = []; for (var i = 0; i < 5; i++) var row = []; for (var j = 0; j < 5; j++)
if ((i + j) row.push(new TextCell("1234")); else row.push(new TextCell("5"));
rows.push(row); console.log(drawTable(rows));
// // GETTERS AND SETTERS // var pile = // elements: ["eggshell", "orange peel", "worm"], // get height() // return this.elements.length; // , // set
height(value) // console.log("Ignoring attemp to set high to ", value); // // ;
// console.log(pile.height); // pile.height = 100; // console.log(pile.height);
[1]: Introduction to Object-Oriented JavaScript [2]: How to call parent constructor?

4.6 Networking

POST $.ajax(type:"POST", url:"http://service.com/items", data:JSON.stringify("name":"newiten$

4.7 Logging

Javascript Logging Having a fancy JavaScript debugger is great, but sometimes the fastest way to find bugs is just to dump as much information to the console as you can.
console.log console.assert console.error

4.8 Documentation

Components jsdoc (with docdash template)
JSDoc is an API documentation generator for JavaScript, similar to JavaDoc or PHPDoc. You add documentation comments directly to your source code, right along side the code itself. The JSDoc Tool will scan your source code, and generate a complete HTML documentation website for you.
gulp, PyCharm
Usage Step 1. Install gulp-jsdoc npm install –save-dev gulp gulp-jsdoc docdash
Step 2. Create documentation task Create documentation task in gulpfile.js file
var template = "path": "./node$_modules/docdash$";
gulp.task('docs', function() return gulp.src("./src/*.js") .pipe(jsdoc('./docs', template));); Step 3. Refresh Gulp tasks In pycharm, click to refresh button in gulp window.

Step 4. Add comment to your code Add comment to your code, You can see an example: should.js
/** * Simple utility function for a bit more easier should assertion * extension * @param Function f So called plugin function. It should accept * 2 arguments: ‘should‘ function and ‘Assertion‘ constructor * @memberOf should * @returns Function Returns ‘should‘ function * @static * @example * * should.use(function(should, Assertion) * Assertion.add(’asset’, function() * this.params = operator: ’to be asset’ ; * * this.obj.should.have.property(’id’).which.is.a.Number(); * this.obj.should.have.property(’path’); *) *) */ should.use = function(f) f(should, should.Assertion); return this; ;
Types: boolean, string, number, Array (see more)
Step 5. Run docs task In pycharm, click to docs task in gulp window.

4.9 Error Handling

In javascript bugs may be displayed is NaN or underfined and program still run but after that, the wrong value can cause some mistake when we use it So, finding bugs and fix them is the quiet hard work in javascript But we can do, and this job is called debugging
STRICT MODE This is the way to find errors that javascript ignores. Example is using an undefined variable. if we dont use strick mode, then everything will be ok, but if using, the error will be shown
function SpotProblem() // "use strict"; for (counter = 0; counter < 10; counter++) console.log("Good!"); SpotProblem(); console.log(counter); strick mode can find error when using this in local, but it is still in global. Example: When we forget to declare the key word "new" when create an new Object
"use strict"; function Person(name) this.name = name; var john = Person("John"); console.log(name); And there are another cases, that trick mode is not allowed: Delete an object is not allowed
"use strict"; var x = 3.14; delete x;
"use strict"; var obj = v1: 3, v2: 4; delete pbj;
"use strict"; var func = function(); delete func; Duplicate parameter is not allowed
"use strict"; var func = function(a1, a1) console.log(a1); Reserve Word is not allowed to name variable
"use strict"; var arguments = 5; var eval = 6; console.log(arguments); console.log(eval); TESTING Testing makes sure that the program working well, and if there are any changes, testing will automatic show us the error, thus, we know where need to fix
function Vector(x, y) this.x = x; this.y = y; Vector.prototype.plus = function(other) return new Vector(this.x + other.x, this.y + other.y);
function TestVector() var p1 = new Vector(10, 20); var p2 = new Vector(-10, 5); var p3 = p1.plus(p2);
if (p1.x !== 10) return "fail: x property"; if (p1.y !== 20) return "fail: y property"; if (p2.x !== -10) return "fail: nagative x property"; if (p2.y !== 5) return "fail: y property"; if (p3.x !== 0) return "fail: x property from plus"; if (p3.y !== 25) return "fail: y property from plus"; return "Vector is Oke"; TestVector(); DEBUGGING when the testing is fail, we have to debug to find the bugs. The first we should guess the bug. And then we put break point in the line, we

assume it make bug If that is the exactly bug we want to find, then we fix it, and write more test for this case In this example code below, the function convert the number in the decima to another. we run and see the result is wrong, so we guess that the error may be caused by the result variable, then we put break point in the line contains result variable.
function ConvertNumber(n, base) var result = "", sign = ""; if (n < 0) sign = "-"; n = -n; do result = String(n n /= base; //-> n = Math.floor(n / base); while (n > 0); return sign + result; console.log(ConvertNumber(13, 10)); console.log(ConvertNumber(14, 2)); ERROR PROPAGATION Sometime our code is working well with normal input. But with special one, they can cause error. So, we have to consider all situation can make Flaws, and handling them. This example code below has an if..else to handle the wrong input if user types not a number in the prompt input
function promptNumber(question) var result = Number(prompt(question, "")); if (isNaN(result)) return null; else return result; console.log(promptNumber("How many trees do you see?")); EXCEPTION In the Error Propagation, we can control the errors if we know them. But what will happen if we don't know the error? For solving this problem, javascript provides for us an try...catch.. to control error we dont know or not sure
try throw new Error("Invalid defination"); catch (error) console.log(error);
function promtDirection(question) var result = prompt(question, ""); if (result.toLowerCase() == "left") return "L"; if (result.toLowerCase() == "right") return "R"; throw new Error("Invalid direction: " + result);
function look() if (promtDirection("Which way?") == "L") return "a house"; else return "two angry bears";
try console.log("you see", look()); catch (error) console.log("Something went wrong: " + error); CLEAN UP AFTER EXCEPTIONS We have a block of code below:
var context = null; function withContext(newContext, body) var oldContext = context; context = newContext; var result = body(); context = oldContext; return result; withContext("new", function() var a = b/0; return a;); What would happend with context? It cannot be excute the last line code, because in withContext function, it will throw off the stack by an exception. So javascript provides a try...finally...
var context = null; function withContext(newContext, body) var oldContext = context; context = newContext; try return body(); finally context = oldContext; withContext("new", function() var a = b/0; return a;); SELECTIVE CATCHING There are some errors cannot handle by environment. So, if we let the error go through, it can cause broken program. For examnple, the Error() in environment cannot catch the infinitive loop in the try block, if we dont catch this problem, the programm will crash soon
for (;;) try var dir = promtDirection("Where?"); console.log("You chose ", dir); break; catch (e) console.log("Not a valid direction. Try again."); The loop will break out if the promptDirection() can excute. But it doesn't. Because it is not defined before, so the environment catch it and go through the catch to show error The circle again and again will make the program crash. So we will create a special Exception.
function InputError(message) this.message = message; this.stack = (new Error()).stack; InputError.prototype = Object.create(Error.prototype); InputError.prototype.name = "InputError"; Error: has an property is stack. it contains

all exception, which environment can catch. Then, we have the promptDirection function to return the result if Enter valid format, or an exception if invalid function promptDirection(question) var result = prompt(question, ""); if (result.toLowerCase() == "left") return "L"; if (result.toLowerCase() == "right") return "R"; throw new InputError("Invalid direction: " + result); Finally, we can catch all exception we want
for (;;) try var dir = promptDirection("Where?"); console.log("You choose ", dir); break; catch(e) if (e instanceof InputError) console.log("Not a valid direction. Try again. "); else throw e; ASSERTIONS function AssertionFailed(message) this.message = message; AssertionFailed.prototype = Object.create(Error.prototype);
function assert(test, message) if (!test) throw new AssertionFailed(message);
function lastElement(array) assert(array.length > 0, "empty array in lastElement"); return array[array.length - 1];

4.10 Testing

Mocha Mocha is a feature-rich JavaScript test framework running on Node.js and the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.
Installation bower install -D mocha chai Usage Step 1. Make index.html
<!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>Tests</title> <link rel="stylesheet" media="all" href="mocha.css"> </head> <body> <div id="mocha"></div> <script src="mocha.js"></script> <script src="chai.js"></script> <script src="functions.js"></script> <script>mocha.setup('bdd'); chai.should();</script> <script src="tests.js"></script> <script>mocha.run();</script> </body> </html> Step 2. Edit functions.js
function sum(a, b) return a + b;
function asynchronusSum(a, b) return new Promise(function(fulfill, reject) fulfill(a + b);); Step 3. Edit tests.js
describe('Calculator', function() this.timeout(5000); describe('sum()', function() it('should return sum of two number', function() sum(2, 3).should.equal(5)););
describe('asynchronusSum()', function() it('should return sum of two number', function(done) asynchronusSum(2, 3).then(function(output) output.should.equal(5); done();));););

4.11 Package Manager

Bower A package manager for the web
Web sites are made of lots of things — frameworks, libraries, assets, utilities, and rainbows. Bower manages all these things for you.
Bower works by fetching and installing packages from all over, taking care of hunting, finding, downloading, and saving the stuff you're looking for. Bower keeps track of these packages in a manifest file, bower.json. How you use packages is up to you. Bower provides hooks to facilitate using packages in your tools and workflows.

Bower is optimized for the front-end. Bower uses a flat dependency tree, requiring only one version for each package, reducing page load to a minimum.
http://bower.io/
[code] bower install jquery underscore moment sprintf -S [/code]
HTML <bower based>
<script src="./bower$_c omponents/jquery/dist/jquery.js" >< /script >< scriptsrc =$
$"./bower_c omponents/moment/moment.js" >< /script >< scriptsrc = "./bower_c omponents/underscore/u$
$/script >< scriptsrc = "./bower_c omponents/sprintf/src/sprintf.js" ><$
$/script > HTML < cdnbased >$
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.0.0-beta1/jquery.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/underscore.js/1.8.3/underscore.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/sprintf/1.0.3/sprintf.js"></script>

4.12 Build Tool

Gulp
Automate and enhance your workflow
Here's some of the sweet stuff you try out with this repo.
Compile CoffeeScript (with source maps!) Compile Handlebars Templates Compile SASS with Compass LiveReload require non-CommonJS code, with dependencies Set up module aliases Run a static Node server (with logging) Pop open your app in a Browser Report Errors through Notification Center Image processing Installation npm install -S gulp gulp-concat Usage Watch
var gulp = require('gulp'); var concat = require('gulp-concat'); var uglify = require('gulp-uglify'); var jsdoc = require("gulp-jsdoc");
var third$_p arties = ["bower_c omponents/jquery/dist/jquery.js", "bower_c omponents/bootstrap/dist/js/boots$
var modules = ["modules/your$_s cript.js"];$
gulp.watch(third$_p arties, ['js_t hirdparty']); gulp.watch(modules, ['js_m odules']);$
gulp.task('js$_t hirdparty', function()returngulp.src(third_p arties).pipe(concat('third_p arty.uglify.js')).pipe(u$
gulp.task('js$_m odules', function()returngulp.src(modules).pipe(concat('modules.uglify.js'))//.pipe(uglify($
gulp.task('documentation', function () return gulp .src("./modules/*/*.js") .pipe(jsdoc('./documentation')););
gulp.task('default', ['js$_t hirdparty', 'js_m odules']); http : //gulpjs.com/$
Deprecated grunt

4.13 Make Module

Make Module sample modules: underscore, momentjs
Folder Structure |- docs |- test |- src | |– your$_m odule.js| - .gitignore| - bower.json$

Chương 5

Java

01/11/2017: Java đơn giản là gay nhé. Không chơi. Viết java chỉ viết thế này thôi. Không viết hơn. Thề!
View online http://magizbox.com/training/java/site/
Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

5.1 Get Started

Installation Ubuntu Step 1. Download sdk
http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
Step 2. Create folder jvm
sudo mkdir /usr/lib/jvm/ Step 3. cd to folder downloads jdk and run command
sudo mv jdk1.7.0$_x//usr/lib/jvm/jdk1.7.0_x Runinstall javasudoupdate-alternatives-$
$-install/usr/bin/javajava/usr/lib/jvm/jdk1.7.0_x/jre/bin/java0Addpathjdk:$
$/usr/lib/jvm/jdk1.7.0_x$
su - nano /etc/environment

5.2 Basic Syntax

Variable Types Although Java is object oriented, not all types are objects. It is built on top of basic variable types called primitives.
Here is a list of all primitives in Java:

byte (number, 1 byte) short (number, 2 bytes) int (number, 4 bytes) long (number, 8 bytes) float (float number, 4 bytes) double (float number, 8 bytes) char (a character, 2 bytes) boolean (true or false, 1 byte) Java is a strong typed language, which means variables need to be defined before we use them. Numbers To declare and assign a number use the following syntax:
int myNumber; myNumber = 5; Or you can combine them:
int myNumber = 5; To define a double floating point number, use the following syntax:
double d = 4.5; d = 3.0; If you want to use float, you will have to cast:
float f = (float) 4.5; Or, You can use this:
float f = 4.5f (f is a shorter way of casting float) Characters and Strings In Java, a character is it's own type and it's not simply a number, so it's not common to put an ascii value in it, there is a special syntax for chars:
char c = 'g'; String is not a primitive. It's a real type, but Java has special treatment for String.
Here are some ways to use a string:
// Create a string with a constructor String s1 = new String("Who let the dogs out?"); // Just using "" creates a string, so no need to write it the previous way. String s2 = "Who who who who!"; // Java defined the operator + on strings to concatenate: String s3 = s1 + s2; There is no operator overloading in Java! The operator + is only defined for strings, you will never see it with other objects, only primitives.
You can also concat string to primitives:
int num = 5; String s = "I have " + num + " cookies"; //Be sure not to use "" with primitives. boolean Every comparison operator in java will return the type boolean that not like other languages can only accept two special values: true or false.
boolean b = false; b = true;
boolean toBe = false; b = toBe || !toBe; if (b) System.out.println(toBe);
int children = 0; b = children; // Will not work if (children) // Will not work // Will not work Operators Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:
Arithmetic Operators Relational Operators Bitwise Operators Logical Operators Assignment Operators Misc Operators The Arithmetic Operators Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra.
The following table lists the arithmetic operators:
Operator Description Example + (Addition) Adds values on either side of the operator 10 + 20 -> 30 - (Subtraction) Subtracts right hand operand from left hand operand 10 - 20 -> -10 * (Multiplication) Multiplies values on either side of the operator 10 * 20 -> 200 / (Division) Divides left hand operand by right hand operand 20 / 10 -> 2 ++ (Increment) Increases the value of operand by 1 a = 20
a++ -> 21
– (Decrement) Decreases the value of operand by 1 a = 20
a– -> 19
The Relational Operators There are following relational operators supported by Java language
== (equal to) Checks if the values of two operands are equal or not, if yes then condition becomes true.

Example: (A == B) is not true. 2 != (not equal to) Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
Example: (A != B) is true.
3 > (greater than) Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
Example: (A > B) is not true. 4 < (less than) Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
Example: (A < B) is true. 5 >= (greater than or equal to) Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
Example (A >= B) is not true. 6 <= (less than or equal to) Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.
example(A <= B) is true.
The Bitwise Operators Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.
Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13; now in binary format they will be as follows:
a = 0011 1100
b = 0000 1101
ab = 0000 1100
a|b = 0011 1101
a^b = 00110001
 a = 1100 0011
The following table lists the bitwise operators:
Assume integer variable A holds 60 and variable B holds 13 then:
(bitwise and) Binary AND Operator copies a bit to the result if it exists in both operands.
Example: (A B) will give 12 which is 0000 1100 2 | (bitwise or) Binary OR Operator copies a bit if it exists in either operand.
Example: (A | B) will give 61 which is 0011 1101 3 $^(bitwiseXOR)BinaryXOROperatorcopiesthebitifitissetin$
Example: (A $^B)willgive49whichis00110001$4 $(bitwisecompliment)BinaryOnesComplementOperatorisunar$
Example: (A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number. 5 « (left shift) Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand
Example: A « 2 will give 240 which is 1111 0000 6 » (right shift) Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.
Example: A » 2 will give 15 which is 1111 7 »> (zero fill right shift) Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.
Example: A »>2 will give 15 which is 0000 1111
The Logical Operators The following table lists the logical operators:
Assume Boolean variables A holds true and variable B holds false, then:
(logical and) Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.
Example (A B) is false. 2 || (logical or) Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.

Example (A || B) is true. 3 ! (logical not) Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.
Example !(A B) is true.
The Assignment Operators There are following assignment operators supported by Java language:
Show Examples
SR.NO Operator and Description 1 = Simple assignment operator, Assigns values from right side operands to left side operand.
Example: C = A + B will assign value of A + B into C 2 += Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.
Example: C += A is equivalent to C = C + A 3 -= Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.
Example:C -= A is equivalent to C = C - A 4 *= Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.
Example: C *= A is equivalent to C = C * A 5 /= Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand
ExampleC /= A is equivalent to C = C / A 6
Example: C
ExampleC «= 2 is same as C = C « 2 8 »= Right shift AND assignment operator
Example C »= 2 is same as C = C » 2 9 = Bitwise AND assignment operator.
Example: C = 2 is same as C = C 2 10 $^{=}bitwiseexclusiveORandassignmentoperator.$
Example: C $^{=}2issameasC = C^{2}11| = bitwiseinclusiveORandassignmentoperator.$
Example: C |= 2 is same as C = C | 2
Miscellaneous Operators There are few other operators supported by Java Language.
Conditional Operator (? :) Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide which value should be assigned to the variable. The operator is written as:
variable x = (expression) ? value if true : value if false Following is the example:
public class Test
public static void main(String args[]) int a, b; a = 10; b = (a == 1) ? 20: 30; System.out.println("Value of b is : " + b);
b = (a == 10) ? 20: 30; System.out.println("Value of b is : " + b);   This would produce the following result ?
Value of b is : 30 Value of b is : 20 Precedence of Operators Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:
For example, x = 7 + 3 * 2; here x is assigned 13, not 20 because operator * has higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.
Category Operator Associativity Postfix () [] . (dot operator) Left toright Unary ++ - - ! Right to left Multiplicative * /
Conditional Java uses boolean variables to evaluate conditions. The boolean values true and false are returned when an expression is compared or evaluated. For example:
int a = 4; boolean b = a == 4;
if (b) System.out.println("It's true!"); Of course we don't normally assign a conditional expression to a boolean, we just use the short version:
int a = 4;
if (a == 4) System.out.println("Ohhh! So a is 4!"); Boolean operators There aren't that many operators to use in conditional statements and most of them are pretty strait forward:
int a = 4; int b = 5; boolean result; result = a < b; // true result = a > b; // false result = a <= 4 // a smaller or equal to 4 - true result = b >= 6 // b bigger or equal to 6 - false result = a == b // a equal to b - false result = a != b // a is not equal to b - true result = a > b || a < b // Logical or - true result = 3 < a a < 6 // Logical and - true result = !result // Logical not - false if - else and between The if, else statement in java is pretty simple.
if (a == b) // a and b are equal, let's do something cool And we can also add an else statement after an if, to do something if the condition is not true
if (a == b) // We already know this part else // a and b are not equal... :/
The if - else statements doesn't have to be in several lines with , if can be used in one line, or without the , for a single line statment.
if (a == b) System.out.println("Another line Wow!"); else System.out.println("Double rainbow!"); Although this method might be useful for making your code shorter by using fewer lines, we strongly recommend for beginners not to use this short version of statements and always use the full version with . This goes to every statement that can be shorted to a single line (for, while, etc).
The ugly side of if There is a another way to write a one line if - else statement by using the operator ? :
int a = 4; int result = a == 4 ? 1 : 8;
// result will be 1 // This is equivalent to int result;
if (a == 4) result = 1; else result = 8; Again, we strongly recommend for beginners not to use this version of if.
== and equals The operator == works a bit different on objects than on primitives. When we are using objects and want to check if they are equal, the operator == will say if they are the same, if you want to check if they are logically equal, you should use the equals method on the object. For example:
String a = new String("Wow"); String b = new String("Wow"); String sameA = a;
boolean r1 = a == b; // This is false, since a and b are not the same object boolean r2 = a.equals(b); // This is true, since a and b are logically equals boolean r3 = a == sameA; // This is true, since a and sameA are really the same object

5.3 Data Structure

Data Structure Number, String Convert number to string
String.valueOf(1000) Make a random
// create a random number from 0 to 99 (new Random()).nextInt(100) Collection Arrays Arrays in Java are also objects. They need to be declared and then created. In order to declare a variable that will hold an array of integers, we use the following syntax:
int[] arr; Notice there is no size, since we didn't create the array yet.
arr = new int[10]; This will create a new array with the size of 10. We can check the size by printing the array's length:
System.out.println(arr.length); We can access the array and set values:
arr[0] = 4; arr[1] = arr[0] + 5; Java arrays are 0 based, which means the first element in an array is accessed at index 0 (e.g: arr[0], which accesses the first element). Also, as an example, an array of size 5 will only go up to index 4 due to it being 0 based.
int[] arr = new int[5] //accesses and sets the first element arr[0] = 4; We can also create an array with values in the same line:
int[] arr = 1, 2, 3, 4, 5; Don't try to print the array without a loop, it will print something nasty like [I@f7e6a96.
Set
import java.util.HashSet; import java.util.Set;
public class HelloWorld
public static void main(String []args) Set<Dog> dogs = new HashSet<Dog>(); Dog dog1 = new Dog("a", 1); Dog dog2 = new Dog("a", 2); Dog dog3 = new Dog("a", 1); Dog dog4 = new Dog("b", 1); dogs.add(dog1); dogs.add(dog2); dogs.add(dog3); dogs.add(dog4); System.out.println(dogs.size());
// 3 public class Dog public String name; public int age; public int value; public Dog(String name, int age) this.name = name; this.age = age; value = (this.name + String.valueOf(this.age)).hashCode();
@Override public int hashCode() return value;
@Override public boolean equals(Object obj) return (obj instanceof Dog ((Dog) obj).value == this.value); List List<String> places = Arrays.asList("Buenos Aires", "Córdoba", "La Plata"); Datetime Calendar c = Calendar.getInstance(); Suggest Readings Initialization of an ArrayList in one line How to convert from int to String?

5.4 OOP

5.4.1 Classes

Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts
Classes and Objects Encapsulation Inheritance Polymorphism Abstraction Instance Method Message Parsing In this chapter, we will look into the concepts - Classes and Objects.
Object Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class. Class A class can be defined as a template/blueprint

that describes the behavior/state that the object of its type support. Objects Let us now look deep into what are objects. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.
If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running.
If you compare the software object with a real-world object, they have very similar characteristics.
Software objects also have a state and a behavior. A software object's state is stored in fields and behavior is shown via methods.
So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.
Classes A class is a blueprint from which individual objects are created.
Following is a sample of a class.
Example
public class Dog String breed; int ageC String color;
void barking()
void hungry()
void sleeping() A class can contain any of the following variable types.
Local variables Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed. Instance variables Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class. Class variables Class variables are variables declared within a class, outside any method, with the static keyword. A class can have any number of methods to access the value of various kinds of methods. In the above example, barking(), hungry() and sleeping() are methods.
Following are some of the important topics that need to be discussed when looking into classes of the Java Language.
Constructors When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.
Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.
Following is an example of a constructor
Example
public class Puppy public Puppy()
public Puppy(String name) // This constructor has one parameter, name. Java also supports Singleton Classes where you would be able to create only one instance of a class.
Note We have two different types of constructors. We are going to discuss constructors in detail in the subsequent chapters.
Creating an Object As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.
There are three steps when creating an object from a class

Declaration A variable declaration with a variable name with an object type. Instantiation The 'new' keyword is used to create the object. Initialization The 'new' keyword is followed by a call to a constructor. This call initializes the new object. Following is an example of creating an object
Example
public class Puppy public Puppy(String name) // This constructor has one parameter, name. System.out.println("Passed Name is :" + name);
public static void main(String []args) // Following statement would create an object myPuppy Puppy myPuppy = new Puppy("tommy"); If we compile and run the above program, then it will produce the following result
Passed Name is :tommy Accessing Instance Variables and Methods Instance variables and methods are accessed via created objects. To access an instance variable, following is the fully qualified path
/* First create an object */ ObjectReference = new Constructor();
/* Now call a variable as follows */ ObjectReference.variableName;
/* Now you can call a class method as follows */ ObjectReference.MethodName();
Example
This example explains how to access instance variables and methods of a class.
public class Puppy int puppyAge;
public Puppy(String name) // This constructor has one parameter, name. System.out.println("Name chosen is :" + name);
public void setAge(int age) puppyAge = age;
public int getAge( ) System.out.println("Puppy's age is :" + puppyAge); return puppyAge;
public static void main(String []args) /* Object creation */ Puppy myPuppy = new Puppy("tommy");
/* Call class method to set puppy's age */ myPuppy.setAge(2);
/* Call another class method to get puppy's age */ myPuppy.getAge( );
/* You can access instance variable as follows as well */ System.out.println("Variable Value :" + myPuppy.puppyAge); If we compile and run the above program, then it will produce the following result
Output
Name chosen is :tommy Puppy's age is :2 Variable Value :2 Source File Declaration Rules As the last part of this section, let's now look into the source file declaration rules. These rules are essential when declaring classes, import statements and package statements in a source file.
There can be only one public class per source file. A source file can have multiple non-public classes. The public class name should be the name of the source file as well which should be appended by .java at the end. For example: the class name is public class Employee then the source file should be as Employee.java. If the class is defined inside a package, then the package statement should be the first statement in the source file. If import statements are present, then they must be written between the package statement and the class declaration. If there are no package statements, then the import statement should be the first line in the source file. Import and package statements will imply to all the classes present in the source file. It is not possible to declare different import and/or package statements to different classes in the source file. Classes have several access levels and there are different types of classes; abstract classes, final classes, etc. We will be explaining about all these in the access modifiers chapter.

Apart from the above mentioned types of classes, Java also has some special classes called Inner classes and Anonymous classes.
Java Package In simple words, it is a way of categorizing the classes and interfaces. When developing applications in Java, hundreds of classes and interfaces will be written, therefore categorizing these classes is a must as well as makes life much easier.
Import Statements In Java if a fully qualified name, which includes the package and the class name is given, then the compiler can easily locate the source code or classes. Import statement is a way of giving the proper location for the compiler to find that particular class.
For example, the following line would ask the compiler to load all the classes available in directory java$_{i}nstallation/java/io$
import java.io.*; A Simple Case Study For our case study, we will be creating two classes. They are Employee and EmployeeTest.
First open notepad and add the following code. Remember this is the Employee class and the class is a public class. Now, save this source file with the name Employee.java.
The Employee class has four instance variables - name, age, designation and salary. The class has one explicitly defined constructor, which takes a parameter.
Example
import java.io.*; public class Employee
String name; int age; String designation; double salary;
// This is the constructor of the class Employee public Employee(String name)
this.name = name;
// Assign the age of the Employee to the variable age. public void empAge(int empAge) age = empAge;
/* Assign the designation to the variable designation.*/ public void empDesignation(String empDesig) designation = empDesig;
/* Assign the salary to the variable salary.*/ public void empSalary(double empSalary) salary = empSalary;
/* Print the Employee details */ public void printEmployee() System.out.println("Name:"+ name); System.out.println("Age:" + age); System.out.println("Designation:" + designation); System.out.println("Salary:" + salary); As mentioned previously in this tutorial, processing starts from the main method. Therefore, in order for us to run this Employee class there should be a main method and objects should be created. We will be creating a separate class for these tasks.
Following is the EmployeeTest class, which creates two instances of the class Employee and invokes the methods for each object to assign values for each variable.
Save the following code in EmployeeTest.java file.
import java.io.*; public class EmployeeTest
public static void main(String args[]) /* Create two objects using constructor */ Employee empOne = new Employee("James Smith"); Employee empTwo = new Employee("Mary Anne");
// Invoking methods for each object created empOne.empAge(26); empOne.empDesignation("Senior Software Engineer"); empOne.empSalary(1000); empOne.printEmployee();
empTwo.empAge(21); empTwo.empDesignation("Software Engineer"); empTwo.empSalary(500);
empTwo.printEmployee(); Now, compile both the classes and then run EmployeeTest to see the result as follows
Output

C:$javacEmployee.javaC : javacEmployeeTest.javaC : javaEmployeeTestName : JamesSmithAge : 26Designation : SeniorSoftwareEngineerSalary : 1000.0Name : MaryAnneAge : 21Designation : SoftwareEngineerSalary : 500.0$

5.4.2 Encapsulation

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.
Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.
Implementation To achieve encapsulation in Java
Declare the variables of a class as private. Provide public setter and getter methods to modify and view the variables values. Example Following is an example that demonstrates how to achieve Encapsulation in Java
/* File name : EncapTest.java */ public class EncapTest private String name; private String idNum; private int age;
public int getAge() return age;
public String getName() return name;
public String getIdNum() return idNum;
public void setAge(int newAge) age = newAge;
public void setName(String newName) name = newName;
public void setIdNum(String newId) idNum = newId; The public setXXX() and getXXX() methods are the access points of the instance variables of the EncapTest class. Normally, these methods are referred as getters and setters. Therefore, any class that wants to access the variables should access them through these getters and setters.
The variables of the EncapTest class can be accessed using the following program
/* File name : RunEncap.java */ public class RunEncap
public static void main(String args[]) EncapTest encap = new EncapTest(); encap.setName("James"); encap.setAge(20); encap.setIdNum("12343ms");
System.out.print("Name : " + encap.getName() + " Age : " + encap.getAge());
This will produce the following result
Name : James Age : 20 Benefits The fields of a class can be made read-only or write-only. A class can have total control over what is stored in its fields. The users of a class do not know how the class stores its data. A class can change the data type of a field and users of the class do not need to change any of their code. Related Readings "Java Inheritance". www.tutorialspoint.com. N.p., 2016. Web. 10 Dec. 2016.

5.4.3 Inheritance

In the preceding lessons, you have seen inheritance mentioned several times. In the Java language, classes can be derived from other classes, thereby inheriting fields and methods from those classes.
The idea of inheritance is simple but powerful: When you want to create a new class and there is already a class that includes some of the code that you want,

you can derive your new class from the existing class. In doing this, you can reuse the fields and methods of the existing class without having to write (and debug!) them yourself.
A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.
Class Hierarchy The Object class, defined in the java.lang package, defines and implements behavior common to all classes—including the ones that you write. In the Java platform, many classes derive directly from Object, other classes derive from some of those classes, and so on, forming a hierarchy of classes.
At the top of the hierarchy, Object is the most general of all classes. Classes near the bottom of the hierarchy provide more specialized behavior.
An Example Here is the sample code for a possible implementation of a Bicycle class that was presented in the Classes and Objects lesson:
public class Bicycle
// the Bicycle class has three fields public int cadence; public int gear; public int speed;
// the Bicycle class has one constructor public Bicycle(int startCadence, int startSpeed, int startGear) gear = startGear; cadence = startCadence; speed = startSpeed;
// the Bicycle class has four methods public void setCadence(int newValue) cadence = newValue;
public void setGear(int newValue) gear = newValue;
public void applyBrake(int decrement) speed -= decrement;
public void speedUp(int increment) speed += increment;
A class declaration for a MountainBike class that is a subclass of Bicycle might look like this:
public class MountainBike extends Bicycle
// the MountainBike subclass adds one field public int seatHeight;
// the MountainBike subclass has one constructor public MountainBike(int startHeight, int startCadence, int startSpeed, int startGear) super(startCadence, startSpeed, startGear); seatHeight = startHeight;
// the MountainBike subclass adds one method public void setHeight(int newValue) seatHeight = newValue; MountainBike inherits all the fields and methods of Bicycle and adds the field seatHeight and a method to set it. Except for the constructor, it is as if you had written a new MountainBike class entirely from scratch, with four fields and five methods. However, you didn't have to do all the work. This would be especially valuable if the methods in the Bicycle class were complex and had taken substantial time to debug.
What You Can Do in a Subclass A subclass inherits all of the public and protected members of its parent, no matter what package the subclass is in. If the subclass is in the same package as its parent, it also inherits the package-private members of the parent. You can use the inherited members as is, replace them, hide them, or supplement them with new members:
The inherited fields can be used directly, just like any other fields. You can declare a field in the subclass with the same name as the one in the superclass, thus hiding it (not * recommended). You can declare new fields in the subclass that are not in the superclass. The inherited methods can be used directly as they are. You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it. You can write

a new static method in the subclass that has the same signature as the one in the superclass, thus hiding it. You can declare new methods in the subclass that are not in the superclass. You can write a subclass constructor that invokes the constructor of the superclass, either implicitly or by using the keyword super. The following sections in this lesson will expand on these topics.
Private Members in a Superclass A subclass does not inherit the private members of its parent class. However, if the superclass has public or protected methods for accessing its private fields, these can also be used by the subclass.
A nested class has access to all the private members of its enclosing class—both fields and methods. Therefore, a public or protected nested class inherited by a subclass has indirect access to all of the private members of the superclass.
Casting Objects We have seen that an object is of the data type of the class from which it was instantiated. For example, if we write
public MountainBike myBike = new MountainBike(); then myBike is of type MountainBike.
MountainBike is descended from Bicycle and Object. Therefore, a MountainBike is a Bicycle and is also an Object, and it can be used wherever Bicycle or Object objects are called for.
The reverse is not necessarily true: a Bicycle may be a MountainBike, but it isn't necessarily. Similarly, an Object may be a Bicycle or a MountainBike, but it isn't necessarily.
Casting shows the use of an object of one type in place of another type, among the objects permitted by inheritance and implementations. For example, if we write
Object obj = new MountainBike(); then obj is both an Object and a MountainBike (until such time as obj is assigned another object that is not a MountainBike). This is called implicit casting.
If, on the other hand, we write
MountainBike myBike = obj; we would get a compile-time error because obj is not known to the compiler to be a MountainBike. However, we can tell the compiler that we promise to assign a MountainBike to obj by explicit casting:
MountainBike myBike = (MountainBike)obj; This cast inserts a runtime check that obj is assigned a MountainBike so that the compiler can safely assume that obj is a MountainBike. If obj is not a MountainBike at runtime, an exception will be thrown.
Related Readings "Inheritance". docs.oracle.com. N.p., 2016. Web. 8 Dec. 2016. "Java Inheritance". www.tutorialspoint.com. N.p., 2016. Web. 8 Dec. 2016. Friesen, Jeff. "Java 101: Inheritance In Java, Part 1". JavaWorld. N.p., 2016. Web. 8 Dec. 2016.

5.4.4 Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.
Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object.
It is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared,

the type of a reference variable cannot be changed.
The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.
A reference variable can refer to any object of its declared type or any subtype of its declared type. A reference variable can be declared as a class or interface type.
Example Let us look at an example.
public interface Vegetarian public class Animal public class Deer extends Animal implements Vegetarian Now, the Deer class is considered to be polymorphic since this has multiple inheritance. Following are true for the above examples
A Deer IS-A Animal A Deer IS-A Vegetarian A Deer IS-A Deer A Deer IS-A Object When we apply the reference variable facts to a Deer object reference, the following declarations are legal
Deer d = new Deer(); Animal a = d; Vegetarian v = d; Object o = d; All the reference variables d, a, v, o refer to the same Deer object in the heap.
Virtual Methods In this section, I will show you how the behavior of overridden methods in Java allows you to take advantage of polymorphism when designing your classes.
We already have discussed method overriding, where a child class can override a method in its parent. An overridden method is essentially hidden in the parent class, and is not invoked unless the child class uses the super keyword within the overriding method.
/* File name : Employee.java */ public class Employee private String name; private String address; private int number;
public Employee(String name, String address, int number) System.out.println("Constructing an Employee"); this.name = name; this.address = address; this.number = number;
public void mailCheck() System.out.println("Mailing a check to " + this.name + " " + this.address);
public String toString() return name + " " + address + " " + number;
public String getName() return name;
public String getAddress() return address;
public void setAddress(String newAddress) address = newAddress;
public int getNumber() return number; Now suppose we extend Employee class as follows
/* File name : Salary.java */ public class Salary extends Employee private double salary; // Annual salary
public Salary(String name, String address, int number, double salary) super(name, address, number); setSalary(salary);
public void mailCheck() System.out.println("Within mailCheck of Salary class "); System.out.println("Mailing check to " + getName() + " with salary " + salary);
public double getSalary() return salary;
public void setSalary(double newSalary) if(newSalary >= 0.0) salary = newSalary;

public double computePay() System.out.println("Computing salary pay for " + getName()); return salary/52; Now, you study the following program carefully and try to determine its output
/* File name : VirtualDemo.java */ public class VirtualDemo

public static void main(String [] args) Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00); Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00); System.out.println("Call mailCheck using Salary reference –"); s.mailCheck(); System.out.println("Call mailCheck using Employee reference–"); e.mailCheck(); This will produce the following result Constructing an Employee Constructing an Employee
Call mailCheck using Salary reference – Within mailCheck of Salary class Mailing check to Mohd Mohtashim with salary 3600.0
Call mailCheck using Employee reference– Within mailCheck of Salary class Mailing check to John Adams with salary 2400.0 Here, we instantiate two Salary objects. One using a Salary reference s, and the other using an Employee reference e.
While invoking s.mailCheck(), the compiler sees mailCheck() in the Salary class at compile time, and the JVM invokes mailCheck() in the Salary class at run time.
mailCheck() on e is quite different because e is an Employee reference. When the compiler sees e.mailCheck(), the compiler sees the mailCheck() method in the Employee class.
Here, at compile time, the compiler used mailCheck() in Employee to validate this statement. At run time, however, the JVM invokes mailCheck() in the Salary class.
This behavior is referred to as virtual method invocation, and these methods are referred to as virtual methods. An overridden method is invoked at run time, no matter what data type the reference is that was used in the source code at compile time.
Related Readings "Java Polymorphism". www.tutorialspoint.com. N.p., 2016. Web. 10 Dec. 2016.

5.4.5 Abstraction

As per dictionary, abstraction is the quality of dealing with ideas rather than events. For example, when you consider the case of e-mail, complex details such as what happens as soon as you send an e-mail, the protocol your e-mail server uses are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver, and click send.
Likewise in Object-oriented programming, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.
In Java, abstraction is achieved using Abstract classes and interfaces.
Abstract Class A class which contains the abstract keyword in its declaration is known as abstract class.
Abstract classes may or may not contain abstract methods, i.e., methods without body (public void get();) But, if a class has at least one abstract method, then the class must be declared abstract. If a class is declared abstract, it cannot be instantiated. To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it. If you inherit an abstract class, you have to provide implementations to all the abstract methods in it. Example

This section provides you an example of the abstract class. To create an abstract class, just use the abstract keyword before the class keyword, in the class declaration.
/* File name : Employee.java */ public abstract class Employee private String name; private String address; private int number;
public Employee(String name, String address, int number) System.out.println("Constructing an Employee"); this.name = name; this.address = address; this.number = number;
public double computePay() System.out.println("Inside Employee computePay"); return 0.0;
public void mailCheck() System.out.println("Mailing a check to " + this.name + " " + this.address);
public String toString() return name + " " + address + " " + number;
public String getName() return name;
public String getAddress() return address;
public void setAddress(String newAddress) address = newAddress;
public int getNumber() return number; You can observe that except abstract methods the Employee class is same as normal class in Java. The class is now abstract, but it still has three fields, seven methods, and one constructor.
Now you can try to instantiate the Employee class in the following way
/* File name : AbstractDemo.java */ public class AbstractDemo
public static void main(String [] args) /* Following is not allowed and would raise error */ Employee e = new Employee("George W.", "Houston, TX", 43);
System.out.println("Call mailCheck using Employee reference–"); e.mailCheck();
When you compile the above class, it gives you the following error
Employee.java:46: Employee is abstract; cannot be instantiated Employee e = new Employee("George W.", "Houston, TX", 43); $^{1}errorInheritingtheAbstractClassWecaninherittheprope$
/* File name : Salary.java */ public class Salary extends Employee private double salary; // Annual salary
public Salary(String name, String address, int number, double salary) super(name, address, number); setSalary(salary);
public void mailCheck() System.out.println("Within mailCheck of Salary class "); System.out.println("Mailing check to " + getName() + " with salary " + salary);
public double getSalary() return salary;
public void setSalary(double newSalary) if(newSalary >= 0.0) salary = newSalary;

public double computePay() System.out.println("Computing salary pay for " + getName()); return salary/52; Here, you cannot instantiate the Employee class, but you can instantiate the Salary Class, and using this instance you can access all the three fields and seven methods of Employee class as shown below.
/* File name : AbstractDemo.java */ public class AbstractDemo
public static void main(String [] args) Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00); Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00); System.out.println("Call mailCheck using Salary reference –"); s.mailCheck(); System.out.println("mailCheck using Employee reference–"); e.mailCheck(); This produces the following result
Constructing an Employee Constructing an Employee Call mailCheck using Salary reference – Within mailCheck of Salary class Mailing check to Mohd Mohtashim with salary 3600.0

Call mailCheck using Employee reference– Within mailCheck of Salary class Mailing check to John Adams with salary 2400.0 Abstract Methods If you want a class to contain a particular method but you want the actual implementation of that method to be determined by child classes, you can declare the method in the parent class as an abstract.
abstract keyword is used to declare the method as abstract. You have to place the abstract keyword before the method name in the method declaration. An abstract method contains a method signature, but no method body. Instead of curly braces, an abstract method will have a semoi colon (;) at the end. Following is an example of the abstract method.
public abstract class Employee private String name; private String address; private int number;
public abstract double computePay(); // Remainder of class definition Declaring a method as abstract has two consequences
The class containing it must be declared as abstract. Any class inheriting the current class must either override the abstract method or declare itself as abstract. Note Eventually, a descendant class has to implement the abstract method; otherwise, you would have a hierarchy of abstract classes that cannot be instantiated.
Suppose Salary class inherits the Employee class, then it should implement the computePay() method as shown below
/* File name : Salary.java */ public class Salary extends Employee private double salary; // Annual salary
public double computePay() System.out.println("Computing salary pay for " + getName()); return salary/52; // Remainder of class definition Related Readings "Java Abstraction". www.tutorialspoint.com. N.p., 2016. Web. 10 Dec. 2016.

5.5 File System IO

The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc.
Stream A stream can be defined as a sequence of data. There are two kinds of Streams
InPutStream The InputStream is used to read data from a source. OutPutStream The OutputStream is used for writing data to a destination.
Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionality related to streams and I/O. We will see the most commonly used examples one by one
Byte Streams Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, FileInputStream and FileOutputStream. Following is an example which makes use of these two classes to copy an input file into an output file
Example
import java.io.*; public class CopyFile

public static void main(String args[]) throws IOException FileInputStream in = null; FileOutputStream out = null;
try in = new FileInputStream("input.txt"); out = new FileOutputStream("output.txt"); int c; while ((c = in.read()) != -1) out.write(c); finally if (in != null) in.close(); if (out != null) out.close(); Now let's have a file input.txt with the following content
This is test for copy file. As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following $javacCopyFile.java$java CopyFile Character Streams Java Byte streams are used to perform input and output of 8-bit bytes, whereas Java Character streams are used to perform input and output for 16-bit unicode. Though there are many classes related to character streams but the most frequently used classes are, FileReader and FileWriter. Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here the major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time. We can re-write the above example, which makes the use of these two classes to copy an input file (having unicode characters) into an output file
Example
import java.io.*; public class CopyFile
public static void main(String args[]) throws IOException FileReader in = null; FileWriter out = null;
try in = new FileReader("input.txt"); out = new FileWriter("output.txt"); int c; while ((c = in.read()) != -1) out.write(c); finally if (in != null) in.close(); if (out != null) out.close(); Now let's have a file input.txt with the following content
This is test for copy file. As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following $javacCopyFile.java$java CopyFile Standard Streams All the programming languages provide support for standard I/O where the user's program can take input from a keyboard and then produce an output on the computer screen. If you are aware of C or C++ programming languages, then you must be aware of three standard devices STDIN, STDOUT and STDERR. Similarly, Java provides the following three standard streams
Standard Input This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as System.in. Standard Output This is used to output the data produced by the user's program and usually a computer screen is used for standard output stream and represented as System.out. Standard Error This is used to output the error data produced by the user's program and usually a computer screen is used for standard error stream and represented as System.err. Following is a simple program, which creates InputStreamReader to read standard input stream until the user types a "q"
Example
import java.io.*; public class ReadConsole
public static void main(String args[]) throws IOException InputStreamReader cin = null;
try cin = new InputStreamReader(System.in); System.out.println("Enter characters, 'q' to quit."); char c; do c = (char) cin.read(); System.out.print(c);

while(c != 'q'); finally if (cin != null) cin.close(); Let's keep the above code in ReadConsole.java file and try to compile and execute it as shown in the following program. This program continues to read and output the same character until we press 'q'
$javacReadConsole.java$java ReadConsole Enter characters, 'q' to quit. 1 1 e e q q Reading and Writing Files As described earlier, a stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.
Here is a hierarchy of classes to deal with Input and Output streams.
The two important streams are FileInputStream and FileOutputStream, which would be discussed in this tutorial.
FileInputStream This stream is used for reading data from the files. Objects can be created using the keyword new and there are several types of constructors available.
Following constructor takes a file name as a string to create an input stream object to read the file
InputStream f = new FileInputStream("C:/java/hello"); Following constructor takes a file object to create an input stream object to read the file. First we create a file object using File() method as follows
File f = new File("C:/java/hello"); InputStream f = new FileInputStream(f);
Once you have InputStream object in hand, then there is a list of helper methods which can be used to read to stream or to do other operations on the stream.
Method Description 1 public void close() throws IOException
This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.
2 protected void finalize()throws IOException
This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
3 public int read(int r)throws IOException
This method reads the specified byte of data from the InputStream. Returns an int. Returns the next byte of data and -1 will be returned if it's the end of the file.
4 public int read(byte[] r) throws IOException
This method reads r.length bytes from the input stream into an array. Returns the total number of bytes read. If it is the end of the file, -1 will be returned.
5 public int available() throws IOException
Gives the number of bytes that can be read from this file input stream. Returns an int.
There are other important input streams available, for more detail you can refer to the following links
ByteArrayInputStream DataInputStream FileOutputStream FileOutputStream is used to create a file and write data into it. The stream would create a file, if it doesn't already exist, before opening it for output.
Here are two constructors which can be used to create a FileOutputStream object.
Following constructor takes a file name as a string to create an input stream object to write the file

OutputStream f = new FileOutputStream("C:/java/hello") Following constructor takes a file object to create an output stream object to write the file. First, we create a file object using File() method as follows
File f = new File("C:/java/hello"); OutputStream f = new FileOutputStream(f);
Once you have OutputStream object in hand, then there is a list of helper methods, which can be used to write to stream or to do other operations on the stream.
Method Description 1 public void close() throws IOException
This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.
2 protected void finalize()throws IOException
This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
3 public void write(int w)throws IOException
This methods writes the specified byte to the output stream.
4 public void write(byte[] w)
Writes w.length bytes from the mentioned byte array to the OutputStream.
There are other important output streams available, for more detail you can refer to the following links
ByteArrayOutputStream DataOutputStream Example
Following is the example to demonstrate InputStream and OutputStream
import java.io.*; public class fileStreamTest
public static void main(String args[])
try byte bWrite [] = 11,21,3,40,5; OutputStream os = new FileOutputStream("test.txt"); for(int x = 0; x < bWrite.length ; x++) os.write(bWrite[x]); // writes the bytes os.close();
InputStream is = new FileInputStream("test.txt"); int size = is.available();
for(int i = 0; i < size; i++) System.out.print((char)is.read() + " "); is.close();
catch(IOException e) System.out.print("Exception"); The above code would create file test.txt and would write given numbers in binary format. Same would be the output on the stdout screen.
File Navigation and I/O There are several other classes that we would be going through to get to know the basics of File Navigation and I/O.
File Class FileReader Class FileWriter Class Directories in Java A directory is a File which can contain a list of other files and directories. You use File object to create directories, to list down files available in a directory. For complete detail, check a list of all the methods which you can call on File object and what are related to directories.
Creating Directories There are two useful File utility methods, which can be used to create directories
The mkdir( ) method creates a directory, returning true on success and false on failure. Failure indicates that the path specified in the File object already exists, or that the directory cannot be created because the entire path does not exist yet.
The mkdirs() method creates both a directory and all the parents of the directory.
Following example creates "/tmp/user/java/bin" directory
Example
import java.io.File; public class CreateDir

public static void main(String args[]) String dirname = "/tmp/user/java/bin"; File d = new File(dirname);
// Create directory now. d.mkdirs(); Compile and execute the above code to create "/tmp/user/java/bin".
Note Java automatically takes care of path separators on UNIX and Windows as per conventions. If you use a forward slash (/) on a Windows version of Java, the path will still resolve correctly.
Listing Directories You can use list( ) method provided by File object to list down all the files and directories available in a directory as follows
Example
import java.io.File; public class ReadDir
public static void main(String[] args) File file = null; String[] paths;
try // create new file object file = new File("/tmp");
// array of files and directory paths = file.list();
// for each name in the path array for(String path:paths) // prints filename and directory name System.out.println(path); catch(Exception e) // if any error occurs e.printStackTrace(); This will produce the following result based on the directories and files available in your /tmp directory
test1.txt test2.txt ReadDir.java ReadDir.class Related Readings "Java Files And I/O". www.tutorialspoint.com. N.p., 2016. Web. 15 Dec. 2016.

5.6 Error Handling

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.
An exception can occur for many different reasons. Following are some scenarios where an exception occurs.
A user has entered an invalid data. A file that needs to be opened cannot be found. A network connection has been lost in the middle of communications or the JVM has run out of memory. Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.
Based on these, we have three categories of Exceptions. You need to understand them to know how exception handling works in Java.
Type of exceptions Checked Exception
A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions.
For example, if you use FileReader class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a FileNotFoundException occurs, and the compiler prompts the programmer to handle the exception.

```
import java.io.File;
import java.io.FileReader;

public class FilenotFound_Demo {
    public static void main(String args[]) {
        File  file  = new File("E://file.txt");
        FileReader fr  = new FileReader(file);
    }
}
```

If you try to compile the above program, you will get the following exceptions.
C:$javacFilenotFound_Demo.javaFilenotFound_Demo.java : 8 : error : unreportedexceptionFileNotFound$
$newFileReader(file);^1 errorNoteSincethemethodsread()andclose()ofFileReaderclassthrowsIOException$
Unchecked exceptions
An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.
For example, if you have declared an array of size 5 in your program, and trying to call the 6th element of the array then an ArrayIndexOutOfBoundsExceptionexception occurs.
public class Unchecked$_Demo$
public static void main(String args[]) int num[] = 1, 2, 3, 4; System.out.println(num[5]);
If you compile and execute the above program, you will get the following exception.
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5 at Exceptions.Unchecked$_Demo.main(Unchecked_Demo.java : 8)Errors$
These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.
Exception Hierarchy All exception classes are subtypes of the java.lang.Exception class. The exception class is a subclass of the Throwable class. Other than the exception class there is another subclass called Error which is derived from the Throwable class.
Errors are abnormal conditions that happen in case of severe failures, these are not handled by the Java programs. Errors are generated to indicate errors generated by the runtime environment. Example: JVM is out of memory. Normally, programs cannot recover from errors.
The Exception class has two main subclasses: IOException class and RuntimeException Class.
Following is a list of most common checked and unchecked Java's Built-in Exceptions
Exceptions Methods Following is the list of important methods available in the Throwable class.
1 public String getMessage() Returns a detailed message about the exception that has occurred. This message is initialized in the Throwable constructor. 2 public Throwable getCause() Returns the cause of the exception as represented by a Throwable object. 3 public String toString() Returns the name of the class concatenated with the result of getMessage(). 4 public void printStackTrace() Prints the result of toString() along with the stack trace to System.err, the error output stream. 5 public StackTraceElement [] getStackTrace() Returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack, and the last element in the array represents the method

at the bottom of the call stack. 6 public Throwable fillInStackTrace() Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace. Catching Exceptions A method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following
Syntax
try // Protected code catch(ExceptionName e1) // Catch block The code which is prone to exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it. Every try block should be immediately followed either by a catch block or finally block.
A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the catch block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter.
Example
The following is an array declared with 2 elements. Then the code tries to access the 3rd element of the array which throws an exception.
// File Name : ExcepTest.java import java.io.*;
public class ExcepTest
public static void main(String args[]) try int a[] = new int[2]; System.out.println("Access element three :" + a[3]); catch(ArrayIndexOutOfBoundsException e) System.out.println("Exception thrown :" + e); System.out.println("Out of the block"); This will produce the following result
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3 Out of the block Multiple Catch Blocks A try block can be followed by multiple catch blocks. The syntax for multiple catch blocks looks like the following
try // Protected code catch(ExceptionType1 e1) // Catch block catch(ExceptionType2 e2) // Catch block catch(ExceptionType3 e3) // Catch block The previous statements demonstrate three catch blocks, but you can have any number of them after a single try. If an exception occurs in the protected code, the exception is thrown to the first catch block in the list. If the data type of the exception thrown matches ExceptionType1, it gets caught there. If not, the exception passes down to the second catch statement. This continues until the exception either is caught or falls through all catches, in which case the current method stops execution and the exception is thrown down to the previous method on the call stack.
Example
Here is code segment showing how to use multiple try/catch statements.
try file = new FileInputStream(fileName); x = (byte) file.read(); catch(IOException i) i.printStackTrace(); return -1; catch(FileNotFoundException f) // Not valid! f.printStackTrace(); return -1; Catching Multiple Type of Exceptions Since Java 7, you can handle more than one exception using a single catch block, this feature simplifies the code. Here is how you would do it
catch (IOException|FileNotFoundException ex) logger.log(ex); throw ex; The Throws/Throw Keywords If a method does not handle a checked exception, the

method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.
You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the throw keyword.
Try to understand the difference between throws and throw keywords, throws is used to postpone the handling of a checked exception and throw is used to invoke an exception explicitly.
The following method declares that it throws a RemoteException
import java.io.*; public class className
public void deposit(double amount) throws RemoteException // Method implementation throw new RemoteException(); // Remainder of class definition
A method can declare that it throws more than one exception, in which case the exceptions are declared in a list separated by commas. For example, the following method declares that it throws a RemoteException and an InsufficientFundsException
import java.io.*; public class className
public void withdraw(double amount) throws RemoteException, InsufficientFundsException // Method implementation // Remainder of class definition
The Finally Block The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.
Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.
A finally block appears at the end of the catch blocks and has the following syntax
Syntax
try // Protected code catch(ExceptionType1 e1) // Catch block catch(ExceptionType2 e2) // Catch block catch(ExceptionType3 e3) // Catch block finally // The finally block always executes.
Example
public class ExcepTest
public static void main(String args[]) int a[] = new int[2]; try System.out.println("Access element three :" + a[3]); catch(ArrayIndexOutOfBoundsException e) System.out.println("Exception thrown :" + e); finally a[0] = 6; System.out.println("First element value: " + a[0]); System.out.println("The finally statement is executed"); This will produce the following result
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3 First element value: 6 The finally statement is executed Note the following
A catch clause cannot exist without a try statement. It is not compulsory to have finally clauses whenever a try/catch block is present. The try block cannot be present without either catch clause or finally clause. Any code cannot be present in between the try, catch, finally blocks. The try-with-resources Generally, when we use any resources like streams, connections, etc. we have to close them explicitly using finally block. In the following program, we are reading data from a file using FileReader and we are closing it using finally block.
import java.io.File; import java.io.FileReader; import java.io.IOException;
public class ReadData$_{D}emo$
public static void main(String args[]) FileReader fr = null; try File file = new File("file.txt"); fr = new FileReader(file); char [] a = new char[50]; fr.read(a); // reads the content to the array for(char c : a) System.out.print(c); // prints the characters one by one catch(IOException e) e.printStackTrace(); finally try

fr.close(); catch(IOException ex) ex.printStackTrace(); try-with-resources, also referred as automatic resource management, is a new exception handling mechanism that was introduced in Java 7, which automatically closes the resources used within the try catch block.
To use this statement, you simply need to declare the required resources within the parenthesis, and the created resource will be closed automatically at the end of the block. Following is the syntax of try-with-resources statement.
Syntax
try(FileReader fr = new FileReader("file path")) // use the resource catch() // body of catch Following is the program that reads the data in a file using try-with-resources statement.
Example
import java.io.FileReader; import java.io.IOException;
public class Try$_withDemo$
public static void main(String args[]) try(FileReader fr = new FileReader("E://file.txt")) char [] a = new char[50]; fr.read(a); // reads the contentto the array for(char c : a) System.out.print(c); // prints the characters one by one catch(IOException e) e.printStackTrace(); Following points are to be kept in mind while working with try-with-resources statement.
To use a class with try-with-resources statement it should implement AutoCloseable interface and the close() method of it gets invoked automatically at runtime. You can declare more than one class in try-with-resources statement. While you declare multiple classes in the try block of try-with-resources statement these classes are closed in reverse order. Except the declaration of resources within the parenthesis everything is the same as normal try/catch block of a try block. The resource declared in try gets instantiated just before the start of the try-block. The resource declared at the try block is implicitly declared as final. User-defined Exceptions You can create your own exceptions in Java. Keep the following points in mind when writing your own exception classes
All exceptions must be a child of Throwable. If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the Exception class. If you want to write a runtime exception, you need to extend the RuntimeException class. We can define our own Exception class as below
class MyException extends Exception You just need to extend the predefined Exception class to create your own Exception. These are considered to be checked exceptions. The following InsufficientFundsException class is a user-defined exception that extends the Exception class, making it a checked exception. An exception class is like any other class, containing useful fields and methods.
Example
// File Name InsufficientFundsException.java import java.io.*;
public class InsufficientFundsException extends Exception private double amount; public InsufficientFundsException(double amount) this.amount = amount; public double getAmount() return amount; To demonstrate using our user-defined exception, the following CheckingAccount class contains a withdraw() method that throws an InsufficientFundsException.
// File Name CheckingAccount.java import java.io.*;
public class CheckingAccount private double balance; private int number;
public CheckingAccount(int number) this.number = number;

public void deposit(double amount) balance += amount;
public void withdraw(double amount) throws InsufficientFundsException if(amount <= balance) balance -= amount; else double needs = amount - balance; throw new InsufficientFundsException(needs);
public double getBalance() return balance;
public int getNumber() return number; The following BankDemo program demonstrates invoking the deposit() and withdraw() methods of CheckingAccount.
// File Name BankDemo.java public class BankDemo
public static void main(String [] args) CheckingAccount c = new CheckingAccount(101); System.out.println("Depositing 500..."); *c.deposit*(500.00);
try System.out.println("100..."); *c.withdraw*(100.00); *System.out.println*("600..."); c.withdraw(600.00); catch(InsufficientFundsException e) System.out.println("Sorry, but you are short "+*e.getAmount*()); *e.printStackTrace*(); *CompilealltheabovethreefilesandrunBankDem*
Output
Depositing 500...
Withdrawing 100...
Withdrawing 600...*Sorry*, *butyouareshort*200.0 InsufficientFundsException at CheckingAccount.withdraw(CheckingAccount.java:25) at BankDemo.main(BankDemo.java:13)
Common Exceptions In Java, it is possible to define two catergories of Exceptions and Errors.
JVM Exceptions These are exceptions/errors that are exclusively or logically thrown by the JVM. Examples: NullPointerException, ArrayIndexOutOfBoundsException, ClassCastException. Programmatic Exceptions These exceptions are thrown explicitly by the application or the API programmers. Examples: IllegalArgumentException, IllegalStateException. Suggested Readings "Java Exceptions". 2016. www.Tutorialspoint.Com. https://www.tutorialspoint.com/java/java$_e$*xceptions.htm*.

5.7 Logging

Log4j log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License. log4j is a popular logging package written in Java. log4j has been ported to the C, C++, C, Perl, Python, Ruby, and Eiffel languages.
log4j is highly configurable through external configuration files at runtime. It views the logging process in terms of levels of priorities and offers mechanisms to direct logging information to a great variety of destinations, such as a database, file, console, UNIX Syslog, etc.
log4j has three main components:
loggers: Responsible for capturing logging information. appenders: Responsible for publishing logging information to various preferred destinations. layouts: Responsible for formatting logging information in different styles. log4j features It is thread-safe. It is optimized for speed. It is based on a named logger hierarchy. It supports multiple output appenders per logger. It supports internationalization. It is not restricted to a predefined set of facilities. Logging behavior can be set at runtime using a configuration file. It is designed to handle Java Exceptions from the start. It uses multiple levels, namely ALL, TRACE, DEBUG, INFO, WARN, ERROR and FATAL. The format of the log output can be easily changed by extending the Layout class. The target of the log output

as well as the writing strategy can be altered by implementations of the Appender interface. It is fail-stop. However, although it certainly strives to ensure delivery, log4j does not guarantee that each log statement will be delivered to its destination. Example Step 1: Add log4j dependency to your build.gradle file compile group: 'log4j', name: 'log4j', version: '1.2.17' Step 2: Add log configuration in main/resources/log4j.property
Set root logger level to DEBUG and its only appender to A1. log4j.rootLogger=DEBUG, A1
A1 is set to be a ConsoleAppender. log4j.appender.A1=org.apache.log4j.ConsoleAppender
A1 uses PatternLayout. log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=
Print only messages of level WARN or above in the package com.foo. log4j.logger.com.foo=WARN
Here is another configuration file that uses multiple appenders:
log4j.rootLogger=debug, stdout, R
log4j.appender.stdout=org.apache.log4j.ConsoleAppender log4j.appender.stdout.layout=org.apache.log4j.Pat
Pattern to output the caller's file name and line number. log4j.appender.stdout.layout.ConversionPattern=
log4j.appender.R=org.apache.log4j.RollingFileAppender log4j.appender.R.File=example.log
log4j.appender.R.MaxFileSize=100KB Keep one backup file log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.layout=org.apache.log4j.PatternLayout log4j.appender.R.layout.ConversionPattern=Step 3: Sample log4j program
package logging;
import org.apache.log4j.Logger;
public class LoggingDemo public static void main(String[] args) final Logger logger = Logger.getLogger(LoggingDemo.class); logger.debug("debug statement"); logger.info("info statement"); logger.error("error statement"); Output DEBUG [main] (LoggingDemo.java:10) - debug statement INFO [main] (LoggingDemo.java:11) - info statement ERROR [main] (LoggingDemo.java:12) - error statement Suggested Readings "Log4j Tutorial". 2016. www.tutorialspoint.com. http://www.tutorialspoint.com/log4j/. "Java Logging". 2016. tutorials.jenkov.com. http://tutorials.jenkov.com/java-logging/index.html.

5.8 IDE

Java: IDE IntellIJ 1. Project Manager 2. Search Replace 3. Navigation 4. Formatting 5. Debugging 6. Build Release 7. Git Integration 1. Project Manager 1.1 Create New Project
1.2 Import Maven Project
https://www.jetbrains.com/help/idea/2016.1/importing-project-from-maven-model.html
2. Search Replace Global Search Shift Shift 3. Navigation Next/Previous Error F2 / Shift + F2 4. Formatting Auto Format Ctrl + Alt + L

5.9 Package Manager

Java: Package Manager Gradle
Create your first project with gradle Step 1: Create new project folder
mkdir gradle$_sampleStep2 : Makefolderstructure$
gradle init –type java-library Step 3: Import to IntelliJ

Open IntelliJ, click File > New... > Project From Existing Sources... Plugins
Application plugin Usages
1. Using the application plugin
Add this line in build.gradle
apply plugin: 'application' 2. Configure the application main class
mainClassName = "org.gradle.sample.Main"

5.10 Build Tool

Java: Build Tool Apache Ant
Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks. 1
Install Ant Download and extract Apache Ant 1.9.6
wget http://mirrors.viethosting.vn/apache//ant/binaries/apache-ant-1.9.6-bin.tar.gz
tar -xzf apache-ant-1.9.6-bin.tar.gz Set path to ant folder
Build Ant through proxy Requirement: 1.9.5+
Add the following lines into build.xml
<target name="ivy-init" depends="ivy-proxy, ivy-probe-antlib, ivy-init-antlib" description="-> initialise Ivy settings"> <ivy:settings file="*ivy.dir/ivysettings.xml*"/ >< */target* >< *targetname* = "*ivy−proxy*"*description* = "−− > *ProxyIvysettings*" >< *propertyname* = "*proxy.host*"*value* = "*proxy.com*"/ >< *propertyname* = "*proxy.port*"*value* = "8080"/ >< *propertyname* = "*proxy.user*"*value* = "*user*"/ >< *propertyname* = "*proxy.password*"*value* = "*password*"/ >< *setproxyproxyhost* = "proxy.host" proxyport="*proxy.port*"*proxyuser* = "proxy.user" proxypassword="*proxy.password*"/ >< */target* > *ApacheAnt*$^{\mathrm{TM}}$

5.11 Production

Java: Production (Docker) Production with java
Base Image: [java]/java
Docker Folder
your-app/ app bin your$_a$*pp.shlibDockerfilerun.shDockerfile*
FROM java:7
COPY run.sh run.sh run.sh
cd /app/bin chmod u+x your$_a$*pp.sh./your*$_a$*pp.shCompose*
service: build: ./your$_a$*ppcommand* :′ *bashrun.sh*′

Chương 6

PHP

PHP là ngôn ngữ lập trình web dominate tất cả các anh tài khác mà (chắc là) chỉ dịu đi khi mô hình REST xuất hiện. Nhớ lần đầu gặp bạn Laravel mà cảm giác cuộc đời sang trang.
Cuối tuần này lại phải xem làm sao cài được xdebug vào PHPStorm cho thằng em tập tành lập trình. Haizzz
Tương tác với cơ sở dữ liệu
Liệt kê danh sách các bản ghi trong bảng groups
"' $sql = "SELECT * FROM`groups`"$;groups = mysqli$_query($conn, sql); "'
Xóa một bản ghi trong bảng groups
"' $sql = "DELETEFROM`groups`WHEREid = `5`"; mysqli_query($conn, sql); "'
Cài đặt debug trong PHPStorm
https://www.youtube.com/watch?v=mEJ21RB0F14
(1) XAMPP
- Download XAMPP (cho PHP 7.1.x - do XDebug chưa chính thức hỗ trợ 7.2.0) https://www.apachefriends.org/xampp-files/7.1.12/xampp-win32-7.1.12-0-VC14-installer.exe - Install XAMPP xampp-win32-7.1.12-0-VC14-installer.exe
- Truy cập vào địa chỉ http://localhost/dashboard/phpinfo.php để kiểm tra cài đặt đã thành công chưa
(2) Tải và cài đặt PHPStorm
- Download PHPStorm https://download-cf.jetbrains.com/webide/PhpStorm-2017.3.2.exe - Install PHPStorm
(3) Tạo một web project trong PHPStorm - Chọn interpreter trỏ đến PHP trong xampp
(4) Viết một chương trình add.php
"'php $a = 2;$b = 3; $c =$a + b;
echo c; "'
Click vào 'add.php', chọn Debug, PHPStorm sẽ báo chưa cài XDebug
(5) Cài đặt XDebug theo hướng dẫn tại https://gist.github.com/odan/1abe76d373a9cbb15bed
Click vào add.php, chọn Debug
(6) Cài đặt XDebug với PHPStorm Marklets Vào trang https://www.jetbrains.com/phpstorm/marklets/
Trong phần Zend Debugger - chọn cổng 9000 - IP: 127.0.0.1 Nhấn nút Generate
Bookmark các link quot;Start debuggerquot;, quot;Stop debuggerquot; lên trình duyệt
(7) Debug PHP từ trình duyệt

* Vào trang http://localhost/untitled/add.php * Click vào bookmark Start debugger * Trong PHPStorm, nhấn vào biểu tượng quot;Start Listening for PHP Debug Connectionsquot; * Đặt breakpoint tại dòng thứ 5 * Refresh lại trang http://localhost/untitled/add.php, lúc này, breakpoint sẽ dừng ở dòng 5

Chương 7

R

View online http://magizbox.com/training/r/site/
R R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.
R is a GNU package. The source code for the R software environment is written primarily in C, Fortran, and R. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several graphical front-ends available.[
R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S. The project was conceived in 1992, with an initial version released in 1995 and a stable beta version in 2000.

7.1 R Courses

I'm going to give a course about R, but it's take a lot of time to finish. I will give at least one lesson a week. You can track it here
(next) Data visualization with R Everything you need to know about R Read and Write Data Importing data from JSON into R Manipulate Data Manipulate String and Datetime Actually, beside my works, there are a lot of excellent and free courses in the internet for you
Beginner
tryr from codeschool
tryr is a course for beginners created by codeschool. This course contains R Syntax, Vectors, Matrices, Summary Statistics, Factors, Data Frames and Working With Real-World Data sections.
Introduction to R from datacamp

This course created by datacamp - a "online learning platform that focuses on building the best learning experience for Data Science in specific". Here is the introduction about this course quoted from authors "In this introduction to R, you will master the basics of this beautiful open source language such as factors, lists and data frames. With the knowledge gained in this course, you will be ready to undertake your first very own data analysis." It contains 6 chapters: Intro to basics, Vectors, Matrices, Factors, Data frames and Lists.
Intermediate and Advanced
R Programming of Johns Hopkins University in coursera Learn how to program in R and how to use R for effective data analysis. This is the second course in the Johns Hopkins Data Science Specialization. It's a 4-weeks course, contains: Overview of R, R data types and objects, reading and writing data (week 1), Control structures, functions, scoping rules, dates and times (week 2), Loop functions, debugging tools (week 3) and Simulation, code profiling (week 4)
An Introduction to Statistical Learning with Applications in R of two experts Trevor Hastie and Rob Tibshirani from Standfor Unitiversity
This course was introduced by Kevin Markham in r-blogger in september 2014. "I found it to be an excellent course in statistical learning (also known as "machine learning"), largely due to the high quality of both the textbook and the video lectures. And as an R user, it was extremely helpful that they included R code to demonstrate most of the techniques described in the book." In this course you will learn about Statistical Learning, Linear Regression, Classification, Resampling Methods, Linear Model Selection and Regularization, Moving Beyond Linearity, Tree-Based Methods, Support Vector Machines and Unsupervised Learning
Cheatsheet – Python R codes for common Machine Learning Algorithms

7.2 Everything you need to know about R

In this post I maintain all useful references for someone want to write nice R code.
Google's R Style Guide at google R is a high-level programming language used primarily for statistical computing and graphics. The goal of the R Programming Style Guide is to make our R code easier to read, share, and verify. The rules below were designed in collaboration with the entire R user community at Google.
Installing R packages at r-bloggers https://www.r-bloggers.com/installing-r-packages/
This is a short post giving steps on how to actually install R packages.
Managing your projects in a reproducible fashion at nicercode https://nicercode.github.io/blog/2013-04-05-projects/
Managing your projects in a reproducible fashion doesn't just make your science reproducible, it makes your life easier.
Creating R Packages http://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf
This tutorial gives a practical introduction to creating R packages. We discuss how object oriented programming and S formulas can be used to give R code the usual look and feel, how to start a package from a collection of R functions, and how to test the code once the package has been created. As running example we

use functions for standard linear regression analysis which are developed from scratch
How to write trycatch in R http://stackoverflow.com/questions/12193779/how-to-write-trycatch-in-r
Welcome to the R world
Debugging with RStudio https://support.rstudio.com/hc/en-us/articles/200713843-Debugging-with-RStudio
RStudio includes a visual debugger that can help you understand code and find bugs.
Optimising code http://adv-r.had.co.nz/Profiling.htmlperformance-profiling
Optimising code to make it run faster is an iterative process:
Find the biggest bottleneck (the slowest part of your code). Try to eliminate it (you may not succeed but that's ok). Repeat until your code is "fast enough."
This sounds easy, but it's not.

Chương 8

Scala

View online http://magizbox.com/training/scala/site/
Scala is a programming language for general software applications. Scala has full support for functional programming and a very strong static type system. This allows programs written in Scala to be very concise and thus smaller in size than other general-purpose programming languages. Many of Scala's design decisions were inspired by criticism of the shortcomings of Java.

8.1 Installation

Windows Step 1. Download scala from http://www.scala-lang.org/downloads
Step 2. Run installer
Step 3. Verify
Open terminal and check which version of scala
$scala - version$
Scala code runner version 2.11.5 – Copyright 2002-2013, LAMP/EPFL

8.2 IDE

I use IntelliJ IDEA 2016.2 as scala IDE
IntelliJ IDEA Installation Guide Online IDE You can use tryscala as an online IDE
http://www.tryscala.com/

8.3 Basic Syntax

Print print > println("Hello, Scala!");
Hello, Scala! Conditional if Statement
if statement consists of a Boolean expression followed by one or more statements.
var x = 10; if(x < 20) println("This is if statement"); if-else Statement
var x = 30 if(x < 20) println("This is if statement"); else println("This is else statement"); if-else if-else Statement
var x = 30; if(x == 10) println("Value of X is 10"); else if(x == 20) println("Value of X is 20"); else if(x == 30) println("Value of X is 30");

else println("This is else statement"); Coding Convention 1 Keep It Simple Don't pack two much in one expression /* * It's amazing what you can get done in a single statement * But that does not mean you have to do it. */ jp.getRawClasspath.filter($_{.}getEntryKind == IClasspathEntry.CPE_{S}OURCE).iterator.flatMap(entry =$
$flatten(ResourcesPlugin.getWorkspace.getRoot.findMember(entry.getPath)))RefactorThere'salotofval$
$jp.getRawClasspath.filter(_{.}getEntryKind == IClasspathEntry.CPE_{S}OURCE)defworkspaceRoot =$
$ResourcesPlugin.getWorkspace.getRootdeffilesOfEntry(entry : Set[File]) =$
$flatten(worspaceRoot.findMember(entry.getPath)sources.iteratorflatMapfilesOfEntryPreferFunctio$
use vals, not vars use recursions or combinators, not loops use immutable collections concentrate on transformations, not CRUD When to deviate from the default - sometimes, mutable gives better performance. - sometimes (but not that often!) it adds convenience
But don't diablolize local state Why does mutable state lead to complexity?
It interacts with different program parts in ways that are hard to track.
=> Local state is less harmful than global state.
"Var" Shortcuts var interfaces = parseClassHeader()... if (isAnnotation) interfaces += ClassFileAnnotation Refactor
val parsedIfaces = parseClassHeader() val interfaces = if (isAnnotation) parsedIfaces + ClassFileAnnotation else parsedIfaces Martin Odersky - Scala with Style

Chương 9

NodeJS

View online http://magizbox.com/training/nodejs/site/
Node.js is an open-source, cross-platform JavaScript runtime environment for developing a diverse variety of tools and applications. Although Node.js is not a JavaScript framework, many of its basic modules are written in JavaScript, and developers can write new modules in JavaScript. The runtime environment interprets JavaScript using Google's V8 JavaScript engine. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in Web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games). Node.js was originally written in 2009 by Ryan Dahl. The initial release supported only Linux. Its development and maintenance was led by Dahl and later sponsored by Joyent.

9.1 Get Started

Installation Windows In this section I will show you how to Install Node.js® and NPM on Windows
Prerequisites Node isn't a program that you simply launch like Word or Photoshop: you won't find it pinned to the taskbar or in your list of Apps. To use Node you must type command-line instructions, so you need to be comfortable with (or at least know how to start) a command-line tool like the Windows Command Prompt, PowerShell, Cygwin, or the Git shell (which is installed along with Github for Windows).
Installation Overview Installing Node and NPM is pretty straightforward using the installer package available from the Node.js® web site.
Installation Steps 1. Download the Windows installer from the Nodes.js® web site.
2. Run the installer (the .msi file you downloaded in the previous step.)
3. Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings).
4. Restart your computer. You won't be able to run Node.js® until you restart your computer.
Ubuntu In this section I will show you how to Install Node.js® and NPM on Ubuntu

update os sudo apt-get update install node with apt-get sudo apt-get install nodejs install npm with apt-get sudo apt-get install npm Test Make sure you have Node and NPM installed by running simple commands to see what version of each is installed and to run a simple test program:
> node -v v6.9.5
> npm -v 3.10.10 Suggested Readings How To Install Node.js on an Ubuntu 14.04 server How to Install Node.js® and NPM on Windows

9.2 Basic Syntax

Print console.log("Hello World"); Conditional if(you$_{s}mart)console.log(''learnnodejs'');elseconsole.log(''goau$
$0; count < 10; count{+}{+})console.log(count);Functionfunctionprint_{i}nfo(arg1, arg2)console.log(arg1); conso$

9.3 File System IO

File System IO Node implements File I/O using simple wrappers around standard POSIX functions. The Node File System (fs) module can be imported using the following syntax
var fs = require("fs") Synchronous vs Asynchronous Every method in the fs module has synchronous as well as asynchronous forms. Asynchronous methods take the last parameter as the completion function callback and the first parameter of the callback function as error. It is better to use an asynchronous method instead of a synchronous method, as the former never blocks a program during its execution, whereas the second one does.
Example
Create a text file named input.txt with the following content
Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!! Let us create a js file named main.js with the following code
var fs = require("fs");
// Asynchronous read fs.readFile('input.txt', function (err, data) if (err) return console.error(err); console.log("Asynchronous read: " + data.toString()););
// Synchronous read var data = fs.readFileSync('input.txt'); console.log("Synchronous read: " + data.toString());
console.log("Program Ended"); Now run the main.js to see the result
$nodemain.jsVerifytheOutput.$
Synchronous read: Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!!
Program Ended Asynchronous read: Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!! The following sections in this chapter provide a set of good examples on major File I/O methods. Open a File Syntax
Following is the syntax of the method to open a file in asynchronous mode
fs.open(path, flags[, mode], callback) Parameters
Here is the description of the parameters used
path This is the string having file name including path. flags Flags indicate the behavior of the file to be opened. All possible values have been mentioned below. mode It sets the file mode (permission and sticky bits), but only if the

file was created. It defaults to 0666, readable and writeable. callback This is the callback function which gets two arguments (err, fd). Flags
Flags for read/write operations are
r - Open file for reading. An exception occurs if the file does not exist. r+ - Open file for reading and writing. An exception occurs if the file does not exist. rs - Open file for reading in synchronous mode. rs+ - Open file for reading and writing, asking the OS to open it synchronously. See notes for 'rs' about using this with caution. w - Open file for writing. The file is created (if it does not exist) or truncated (if it exists). wx - Like 'w' but fails if the path exists. w+ - Open file for reading and writing. The file is created (if it does not exist) or truncated (if it exists). wx+ - Like 'w+' but fails if path exists. a - Open file for appending. The file is created if it does not exist. ax - Like 'a' but fails if the path exists. a+ - Open file for reading and appending. The file is created if it does not exist. ax+ - Like 'a+' but fails if the the path exists. Example
Let us create a js file named main.js having the following code to open a file input.txt for reading and writing.
var fs = require("fs");
// Asynchronous - Opening File console.log("Going to open file!"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!");); Now run the main.js to see the result
$node main.js Verify the Output.$
Going to open file! File opened successfully! Get File Information Syntax
Following is the syntax of the method to get the information about a file
fs.stat(path, callback) Parameters
Here is the description of the parameters used
path This is the string having file name including path. callback This is the callback function which gets two arguments (err, stats) where stats is an object of fs.Stats type which is printed below in the example. Apart from the important attributes which are printed below in the example, there are several useful methods available in fs.Stats class which can be used to check file type. These methods are given in the following table.
Method Description
stats.isFile() - Returns true if file type of a simple file. stats.isDirectory() - Returns true if file type of a directory. stats.isBlockDevice() - Returns true if file type of a block device. stats.isCharacterDevice() - Returns true if file type of a character device. stats.isSymbolicLink() - Returns true if file type of a symbolic link. stats.isFIFO() - Returns true if file type of a FIFO. stats.isSocket() - Returns true if file type of asocket. Example
Let us create a js file named main.js with the following code
var fs = require("fs");
console.log("Going to get file info!"); fs.stat('input.txt', function (err, stats) if (err) return console.error(err); console.log(stats); console.log("Got file info successfully!");
// Check file type console.log("isFile ? " + stats.isFile()); console.log("isDirectory ? " + stats.isDirectory());); Now run the main.js to see the result
$node main.js Verify the Output.$
Going to get file info! dev: 1792, mode: 33188, nlink: 1, uid: 48, gid: 48, rdev: 0, blksize: 4096, ino: 4318127, size: 97, blocks: 8, atime: Sun Mar 22 2015 13:40:00 GMT-0500 (CDT), mtime: Sun Mar 22 2015 13:40:57 GMT-0500 (CDT), ctime:

Sun Mar 22 2015 13:40:57 GMT-0500 (CDT) Got file info successfully! isFile ? true isDirectory ? false Writing a File Syntax
Following is the syntax of one of the methods to write into a file
fs.writeFile(filename, data[, options], callback) This method will over-write the file if the file already exists. If you want to write into an existing file then you should use another method available.
Parameters
Here is the description of the parameters used
path This is the string having the file name including path. data This is the String or Buffer to be written into the file. options The third parameter is an object which will hold encoding, mode, flag. By default. encoding is utf8, mode is octal value 0666. and flag is 'w' callback This is the callback function which gets a single parameter err that returns an error in case of any writing error. Example
Let us create a js file named main.js having the following code
var fs = require("fs");
console.log("Going to write into existing file"); fs.writeFile('input.txt', 'Simply Easy Learning!', function(err) if (err) return console.error(err);
console.log("Data written successfully!"); console.log("Let's read newly written data"); fs.readFile('input.txt', function (err, data) if (err) return console.error(err); console.log("Asynchronous read: " + data.toString()););); Now run the main.js to see the result
$node main.js Verify the Output.$
Going to write into existing file Data written successfully! Let's read newly written data Asynchronous read: Simply Easy Learning! Reading a File Syntax
Following is the syntax of one of the methods to read from a file
fs.read(fd, buffer, offset, length, position, callback) This method will use file descriptor to read the file. If you want to read the file directly using the file name, then you should use another method available.
Parameters
Here is the description of the parameters used
fd This is the file descriptor returned by fs.open(). buffer This is the buffer that the data will be written to. offset This is the offset in the buffer to start writing at. length This is an integer specifying the number of bytes to read. position This is an integer specifying where to begin reading from in the file. * If position is null, data will be read from the current file position. callback This is the callback function which gets the three arguments, (err, bytesRead, buffer). Example
Let us create a js file named main.js with the following code
var fs = require("fs"); var buf = new Buffer(1024);
console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!"); console.log("Going to read the file"); fs.read(fd, buf, 0, buf.length, 0, function(err, bytes) if (err) console.log(err); console.log(bytes + " bytes read"); // Print only read bytes to avoid junk. if(bytes > 0) console.log(buf.slice(0, bytes).toString()););); Now run the main.js to see the result
$node main.js Verify the Output.$
Going to open an existing file File opened successfully! Going to read the file 97 bytes read Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!! Closing a File Syntax

Following is the syntax to close an opened file
fs.close(fd, callback) Parameters
Here is the description of the parameters used
fd This is the file descriptor returned by file fs.open() method. callback This is the callback function No arguments other than a possible exception are given to the completion callback. Example Let us create a js file named main.js having the following code
var fs = require("fs"); var buf = new Buffer(1024);
console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!"); console.log("Going to read the file");
fs.read(fd, buf, 0, buf.length, 0, function(err, bytes) if (err) console.log(err);
// Print only read bytes to avoid junk. if(bytes $>$ 0) console.log(buf.slice(0, bytes).toString());
// Close the opened file. fs.close(fd, function(err) if (err) console.log(err); console.log("File closed successfully.");););); Now run the main.js to see the result
$node main.js Verify the Output.$
Going to open an existing file File opened successfully! Going to read the file Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!!
File closed successfully. Truncate a File Syntax
Following is the syntax of the method to truncate an opened file
fs.ftruncate(fd, len, callback) Parameters
Here is the description of the parameters used
fd This is the file descriptor returned by fs.open(). len This is the length of the file after which the file will be truncated. callback This is the callback function No arguments other than a possible ekxception are given to the completion callback. Example
Let us create a js file named main.js having the following code
var fs = require("fs"); var buf = new Buffer(1024);
console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!"); console.log("Going to truncate the file after 10 bytes");
// Truncate the opened file. fs.ftruncate(fd, 10, function(err) if (err) console.log(err); console.log("File truncated successfully."); console.log("Going to read the same file");
fs.read(fd, buf, 0, buf.length, 0, function(err, bytes) if (err) console.log(err);
// Print only read bytes to avoid junk. if(bytes $>$ 0) console.log(buf.slice(0, bytes).toString());
// Close the opened file. fs.close(fd, function(err) if (err) console.log(err); console.log("File closed successfully."););););); Now run the main.js to see the result
$node main.js Verify the Output.$
Going to open an existing file File opened successfully! Going to truncate the file after 10 bytes File truncated successfully. Going to read the same file Tutorials File closed successfully. Delete a File Syntax Following is the syntax of the method to delete a file
fs.unlink(path, callback) Parameters
Here is the description of the parameters used

path This is the file name including path. callback This is the callback function No arguments other than a possible exception are given to the completion callback. Example
Let us create a js file named main.js having the following code
var fs = require("fs");
console.log("Going to delete an existing file"); fs.unlink('input.txt', function(err) if (err) return console.error(err); console.log("File deleted successfully!"););
Now run the main.js to see the result
$nodemain.jsVerifytheOutput.$
Going to delete an existing file File deleted successfully! Create a Directory Syntax
Following is the syntax of the method to create a directory
fs.mkdir(path[, mode], callback) Parameters
Here is the description of the parameters used
path This is the directory name including path. mode This is the directory permission to be set. Defaults to 0777. callback This is the callback function No arguments other than a possible exception are given to the completion callback. Example
Let us create a js file named main.js having the following code
var fs = require("fs");
console.log("Going to create directory /tmp/test"); fs.mkdir('/tmp/test',function(err) if (err) return console.error(err); console.log("Directory created successfully!");); Now run the main.js to see the result
$nodemain.jsVerifytheOutput.$
Going to create directory /tmp/test Directory created successfully! Read a Directory Syntax
Following is the syntax of the method to read a directory
fs.readdir(path, callback) Parameters
Here is the description of the parameters used
path This is the directory name including path. callback This is the callback function which gets two arguments (err, files) where files is an array of the names of the files in the directory excluding '.' and '..'. Example
Let us create a js file named main.js having the following code
var fs = require("fs");
console.log("Going to read directory /tmp"); fs.readdir("/tmp/",function(err, files) if (err) return console.error(err); files.forEach(function (file) console.log(file););); Now run the main.js to see the result
$nodemain.jsVerifytheOutput.$
Going to read directory /tmp ccmzx99o.out ccyCSbkF.out employee.ser hsperfdata$_{a}pachetesttest.txtRemove$
Following is the syntax of the method to remove a directory
fs.rmdir(path, callback) Parameters
Here is the description of the parameters used
path This is the directory name including path. callback This is the callback function No argume nts other than a possible exception are given to the completion callback. Example
Let us create a js file named main.js having the following code
var fs = require("fs");
console.log("Going to delete directory /tmp/test"); fs.rmdir("/tmp/test",function(err) if (err) return console.error(err); console.log("Going to read directory /tmp");

fs.readdir("/tmp/",function(err, files) if (err) return console.error(err); files.forEach(function (file) console.log(file);););); Now run the main.js to see the result $nodemain.jsVerifytheOutput.$
Going to read directory /tmp ccmzx99o.out ccyCSbkF.out employee.ser hsperfdata$_apachetest.txt$

9.4 Package Manager

Package Manager: NPM Node Package Manager (NPM) provides two main functionalities
Online repositories for node.js packages/modules which are searchable on search.nodejs.org
Command line utility to install Node.js packages, do version management and dependency management of Node.js packages. NPM comes bundled with Node.js installables after v0.6.3 version. To verify the same, open console and type the following command and see the result
$npm--version2.7.1IfyouarerunninganoldversionofNPMthenitisquiteeasytoupdateittothelatestversion.$
$sudonpminstallnpm-g/usr/bin/npm->/usr/lib/node_modules/npm/bin/npm-$
$cli.jsnpm@2.7.1/usr/lib/node_modules/npmInstallingModulesThereisasimplesyntaxtoinstallanyNode.jsn$
$npminstall<ModuleName>Forexample,followingisthecommandtoinstallafamousNode.jswebframeu$
$npminstallexpressNowyoucanusethismoduleinyourjsfileasfollowing$
var express = require('express'); Global vs Local Installation By default, NPM installs any dependency in the local mode. Here local mode refers to the package installation in node$_modulesdirectorylyinginthefolderwhereNodeapplicationispresent.Locallydeployedpacka$
$ls-ltotal0drwxr-xr-x3rootroot20Mar1702:23node_modulesAlternatively,youcanusenpmlscommandtolis$
Globally installed packages/dependencies are stored in system directory. Such dependencies can be used in CLI (Command Line Interface) function of any node.js but cannot be imported using require() in Node application directly. Now let's try installing the express module using global installation.
$npminstallexpress-gThiswillproduceasimilarresultbutthemodulewillbeinstalledglobally.Here,thefirstlin$
express@4.12.2 /usr/lib/node$_modules/expressmerge-descriptors@1.0.0utils-$
$merge@1.0.0cookie-signature@1.0.6methods@1.1.1fresh@0.2.4cookie@0.1.2escape-$
$html@1.0.1range-parser@1.0.2content-type@1.0.1finalhandler@0.3.3vary@1.0.0parseurl@1.3.0content-$
$disposition@0.5.0path-to-regexp@0.1.3depd@1.0.0qs@2.3.3on-finished@2.2.0(ee-$
$first@1.1.0)etag@1.5.1(crc@3.2.1)debug@2.1.3(ms@0.7.0)proxy-addr@1.0.7(forwarded@0.1.0,ipaddr.js@0$
$static@1.9.2(send@0.12.2)accepts@1.2.5(negotiator@0.5.1,mime-types@2.0.10)type-$
$is@1.6.1(media-typer@0.3.0,mime-types@2.0.10)Youcanusethefollowingcommandtocheckallthemodules$
$npmls-gUsingpackage.jsonpackage.jsonispresentintherootdirectoryofanyNodeapplication/moduleandis$
"name": "express", "description": "Fast, unopinionated, minimalist web framework", "version": "4.11.2", "author":
"name": "TJ Holowaychuk", "email": "tj@vision-media.ca" ,
"contributors": ["name": "Aaron Heckmann", "email": "aaron.heckmann+github@gmail.com"
,
"name": "Ciaran Jessup", "email": "ciaranj@gmail.com" ,
"name": "Douglas Christopher Wilson", "email": "doug@somethingdoug.com"
,
"name": "Guillermo Rauch", "email": "rauchg@gmail.com" ,
"name": "Jonathan Ong", "email": "me@jongleberry.com" ,
"name": "Roman Shtylman", "email": "shtylman+expressjs@gmail.com" ,
"name": "Young Jae Sim", "email": "hanul@hanul.me"], "license": "MIT",
"repository": "type": "git", "url": "https://github.com/strongloop/express" ,

"homepage": "https://expressjs.com/", "keywords": ["express", "framework", "sinatra", "web", "rest", "restful", "router", "app", "api"], "dependencies": "accepts": " 1.2.3", "content-disposition": "0.5.0", "cookie-signature": "1.0.5", "debug": " 2.1.1", "depd": " 1.0.0", "escape-html": "1.0.1", "etag": " 1.5.1", "finalhandler": "0.3.3", "fresh": "0.2.4", "media-typer": "0.3.0", "methods": " 1.1.1", "on-finished": " 2.2.0", "parseurl": " 1.3.0", "path-to-regexp": "0.1.3", "proxy-addr": " 1.0.6", "qs": "2.3.3", "range-parser": " 1.0.2", "send": "0.11.1", "serve-static": " 1.8.1", "type-is": " 1.5.6", "vary": " 1.0.0", "cookie": "0.1.2", "merge-descriptors": "0.0.2", "utils-merge": "1.0.0" , "devDependencies": "after": "0.8.1", "ejs": "2.1.4", "istanbul": "0.3.5", "marked": "0.3.3", "mocha": " 2.1.0", "should": " 4.6.2", "supertest": " 0.15.0", "hjs": " 0.0.6", "body-parser": " 1.11.0", "connect-redis": " 2.2.0", "cookie-parser": " 1.3.3", "express-session": " 1.10.2", "jade": " 1.9.1", "method-override": " 2.3.1", "morgan": " 1.5.1", "multiparty": " 4.1.1", "vhost": " 3.0.0" , "engines": "node": ">= 0.10.0" , "files": ["LICENSE", "History.md", "Readme.md", "index.js", "lib/"], "scripts": "test": "mocha –require test/support/env –reporter spec –bail –check-leaks test/ test/acceptance/", "test-cov": "istanbul cover node$_m$odules/mocha/bin/$_m$ocha−
−−−requiretest/support/env−−reporterdot−−check−leakstest/test/acceptance/”,”test−
tap” : ”mocha−−requiretest/support/env−−reportertap−−check−leakstest/test/acceptance/”,”test−
travis” : ”istanbulcovernode$_m$odules/mocha/bin/$_m$ocha − −reportlcovonly −
−−−requiretest/support/env−−reporterspec−−check−leakstest/test/acceptance/”,”gitHead” :
”63ab25579bda70b4927a179b580a9c580b6c7ada”,”bugs” : ”url” : ”https : //github.com/strongloop/express/
”express@4.11.2”,”$_s$hasum” : ”8df3d5a9ac848585f00a0777601823faecd3b148”,”$_f$rom” :
”express@∗”,”$_n$pmVersion” : ”1.4.28”,”$_n$pmUser” : ”name” : ”dougwilson”,”email” : ”doug@somethingd
[”name” : ”tjholowaychuk”,”email” : ”tj@vision − media.ca”,”name” : ”jongleberry”,”email” : ”jonatha
”shasum” : ”8df3d5a9ac848585f00a0777601823faecd3b148”,”tarball” : ”https : //registry.npmjs.org/expr
,”$_r$esolved” : ”https : //registry.npmjs.org/express/−/express−4.11.2.tgz”,”readme” :
”ERROR : NoREADMEdatafound!”AttributesofPackage.jsonnamenameofthepackageversionversiono
npmuninstallexpressOnceNPMuninstallsthepackage, youcanverifyitbylookingatthecontentof/node$_m$odu
npmlsUpdatingaModuleUpdatepackage.jsonandchangetheversionofthedependencytobeupdatedandrunthef
npmupdateexpressSearchaModuleSearchapackagenameusingNPM.
npmsearchexpressCreateaModuleCreatingamodulerequirespackage.jsontobegenerated.Let′sgeneratepack
npminit
This utility will walk you through creating a package.json file. It only covers the most common items, and tries to guess sane defaults.
See 'npm help json' for definitive documentation on these fields and exactly what they do.
Use 'npm install <pkg> –save' afterwards to install a package and save it as a dependency in the package.json file.
Press ^{C}atanytimetoquit.name : (webmaster)Youwillneedtoprovidealltherequiredinformationaboutyourmo
mentionedpackage.jsonfiletounderstandthemeaningsofvariousinformationdemanded.Oncepackage.jsonn
npmadduserUsername : mcmohdPassword : Email : (thisISpublic)mcmohd@gmail.comItistimenowtopub
npmpublishIfeverythingisfinewithyourmodule, thenitwillbepublishedintherepositoryandwillbeaccessiblet

9.5 Command Line

Pass command line arguments The arguments are stored in process.argv
Here are the node docs on handling command line args:

process.argv is an array containing the command line arguments. The first element will be 'node', the second element will be the name of the JavaScript file. The next elements will be any additional command line arguments.
// print process.argv process.argv.forEach(function (val, index, array) console.log(index + ': ' + val);); This will generate:
$nodeprocess-2.jsonetwo = threefour0 : node1 : /Users/mjr/work/node/process-2.js2 : one3 : two = three4 : four$

Chương 10

Octave

View online http://magizbox.com/training/octave/site/
GNU Octave is software featuring a high-level programming language, primarily intended for numerical computations. Octave helps in solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with MATLAB. It may also be used as a batch-oriented language. Since it is part of the GNU Project, it is free software under the terms of the GNU General Public License.
Known Issues Plot window not responding

10.1 Matrix

Creating Matrix A = [1, 1, 2; 3, 5, 8; 13, 21, 32] A = 1 1 2 3 5 8 13 21 32
Creating an 1D column vector a = [1; 2; 3] a = 1 2 3
Creating an 1D row vector b = [1 2 3] b = 1 2 3
Creating a random m x n matrix rand(3, 2) ans = 0.13567 0.51230 0.67646 0.19012 0.76147 0.89694
Creating a zero m x n matrix zeros(3, 2) ans = 0 0 0 0 0 0
Creating an m x n matrix of ones ones(3,2) ans = 1 1 1 1 1 1
Creating an identity matrix eye(3) Diagonal Matrix 1 0 0 0 1 0 0 0 1
Creating a diagonal matrix a = [1 2 3] diag(a) Diagonal Matrix 1 0 0 0 2 0 0 0 3 Accessing Matrix Elements Getting the dimension of a matrix A = [1 2 3; 4 5 6] size(A) ans = 2 3
Selecting rows A = [1 2 3; 4 5 6; 7 8 9] A(1, :) ans = 1 2 3 A(1:2, :) ans = 1 2 3 4 5 6
Selecting columns A = [1 2 3; 4 5 6; 7 8 9] A(:, 1) ans = 1 4 7 A(:, 1:2) ans = 1 2 4 5 7 8
Extracting rows and columns by criteria A = [1 2 3; 4 5 9; 7 8 9] A(A(:,3) == 9, :) ans = 4 5 9 7 8 9
Accessing elements A = [1 2 3; 4 5 6; 7 8 9] A(1, 1) ans = 1 A(2, 3) ans = 6 Manipulating Shape and Dimensions Converting a matrix into a row vector (by column) A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A(:) ans = 1 4 7 2 5 8 3 6 9
Converting row to column vectors b = [1 2 3] b = 1 2 3 b' ans = 1 2 3

Reshaping Matrices A = [1 2 3 4; 5 6 7 8; 9 10 11 12] A = 1 2 3 4 5 6 7 8 9 10 11 12 reshape(A,4,3) ans = 1 6 11 5 10 4 9 3 8 2 7 12
Concatenating matrices A = [1 2 3; 4 5 6] A = 1 2 3 4 5 6 B = [7 8 9; 10 11 12] B = 7 8 9 10 11 12 C = [A; B] C = 1 2 3 4 5 6 7 8 9 10 11 12
Stacking vectors and matrices a = [1 2 3] a = 1 2 3 b = [4 5 6] b = 4 5 6 c = [a' b'] c = 1 4 2 5 3 6 Basic Operations Matrix-scalar operations A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A * 2 ans = 2 4 6 8 10 12 14 16 18 A + 2 ans = 3 4 5 6 7 8 9 10 11 A - 2 ans = -1 0 1 2 3 4 5 6 7 A / 2 ans = 0.50000 1.00000 1.50000 2.00000 2.50000 3.00000 3.50000 4.00000 4.50000
Matrix-matrix multiplication A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A * A ans = 30 36 42 66 81 96 102 126 150
Matrix-vector multiplication A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 b = [1; 2; 3] b = 1 2 3 A * b ans = 14 32 50
Element-wise matrix-matrix operations A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A .* A ans = 1 4 9 16 25 36 49 64 81 A .+ A ans = 2 4 6 8 10 12 14 16 18 A .- A ans = 0 0 0 0 0 0 0 0 0 A ./ A ans = 1 1 1 1 1 1 1 1 1
Matrix elements to power n A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A.$^2 ans = 149162536496481$
Matrix to power n A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A $^2 ans = 303642668196102126150$
Matrix transpose A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A' ans = 1 4 7 2 5 8 3 6 9
Determinant of a matrix A = [6 1 1; 4 -2 5; 2 8 7] A = 6 1 1 4 -2 5 2 8 7 det(A) ans = -306
Inverse of a matrix A = [4 7; 2 6] A = 4 7 2 6 inv(A) ans = 0.60000 -0.70000 -0.20000 0.40000 Advanced Operations Calculating the covariance matrix of 3 random variables x1 = [4.0000 4.2000 3.9000 4.3000 4.1000]' x1 = 4.0000 4.2000 3.9000 4.3000 4.1000 x2 = [2.0000 2.1000 2.0000 2.1000 2.2000]' x2 = 2.0000 2.1000 2.0000 2.1000 2.2000 x3 = [0.60000 0.59000 0.58000 0.62000 0.63000]' x3 = 0.60000 0.59000 0.58000 0.62000 0.63000 cov([x1,x2,x3]) ans = 2.5000e-002 7.5000e-003 1.7500e-003 7.5000e-003 7.0000e-003 1.3500e-003 1.7500e-003 1.3500e-003 4.3000e-004
Calculating eigenvectors and eigenvalues A = [3 1; 1 3] A = 3 1 1 3 [eig$_vec, eig_val] = eig(A)eig_vec = -0.707110.707110.707110.70711eig_val = DiagonalMatrix2004$
Generating a Gaussian dataset pkg load statistics mean = [0 0] mean = 0 0 cov = [2 0; 0 2] cov = 2 0 0 2 mvnrnd(mean,cov,5) ans = -0.7442485 -0.0099190 -1.7695915 0.0418147 -0.8780206 0.6145333 0.5145315 -0.9834832 -1.4736628 0.4570979

Chương 11

Toolbox

View online http://magizbox.com/training/toolbox/site/
Toolbox by MG The Toolbox contains all the little tools you never know where to find.
Text Editor Vim : Vim is a clone of Bill Joy's vi text editor program for Unix. It was written by Bram Moolenaar based on source for a port of the Stevie editor to the Amiga and first released publicly in 1991. Vim is designed for use both from a command-line interface and as a standalone application in a graphical user interface. Vim is free and open source software and is released under a license that includes some charityware clauses, encouraging users who enjoy the software to consider donating to children in Uganda. The license is compatible with the GNU General Public License. Although it was originally released for the Amiga, Vim has since been developed to be cross-platform, supporting many other platforms. In 2006, it was voted the most popular editor amongst Linux Journal readers; in 2015 the Stack Overflow developer survey found it to be the third most popular text editor; and in 2016 the Stack Overflow developer survey found it to be the fourth most popular development environment.
Virtual Machine VirtualBox : Oracle VM VirtualBox (formerly Sun VirtualBox, Sun xVM VirtualBox and Innotek VirtualBox) is a free and open-source hypervisor for x86 computers currently being developed by Oracle Corporation. Developed initially by Innotek GmbH, it was acquired by Sun Microsystems in 2008 which was in turn acquired by Oracle in 2010. VirtualBox may be installed on a number of host operating systems, including: Linux, macOS, Windows, Solaris, and OpenSolaris. There are also ports to FreeBSD and Genode. It supports the creation and management of guest virtual machines running versions and derivations of Windows, Linux, BSD, OS/2, Solaris, Haiku, OSx86 and others, and limited virtualization of macOS guests on Apple hardware. For some guest operating systems, a "Guest Additions" package of device drivers and system applications is available which typically improves performance, especially of graphics.
VMWare : VMware, Inc. is a subsidiary of Dell Technologies that provides cloud computing and platform virtualization software and services. It was the first commercially successful company to virtualize the x86 architecture. VMware's desktop software runs on Microsoft Windows, Linux, and macOS, while its enterprise software hypervisor for servers, VMware ESXi, is a bare-metal hypervisor that runs directly on server hardware without requiring an additional underlying

operating system.

11.1 Vim

Vim Running Vim for the First Time To start Vim, enter this command:
gvim file.txt In UNIX you can type this at any command prompt. If you are running Microsoft Windows, open an MS-DOS prompt window and enter the command. In either case, Vim starts editing a file called file.txt. Because this is a new file, you get a blank window. This is what your screen will look like:
+————————————+ | | | | | | | | | | | | |"file.txt" [New file] | +————————————+ ('" is the cursor position.) The tilde ( ) lines indicate lines not in the file. In other words, when Vim runs out of file to display, it displays tilde lines. At the bottom of the screen, a message line indicates the file is named file.txt and shows that you are creating a new file. The message information is temporary and other information overwrites it.
THE VIM COMMAND
The gvim command causes the editor to create a new window for editing. If you use this command:
vim file.txt the editing occurs inside your command window. In other words, if you are running inside an xterm, the editor uses your xterm window. If you are using an MS-DOS command prompt window under Microsoft Windows, the editing occurs inside this window. The text in the window will look the same for both versions, but with gvim you have extra features, like a menu bar. More about that later.
Inserting text The Vim editor is a modal editor. That means that the editor behaves differently, depending on which mode you are in. The two basic modes are called Normal mode and Insert mode. In Normal mode the characters you type are commands. In Insert mode the characters are inserted as text. Since you have just started Vim it will be in Normal mode. To start Insert mode you type the "i" command (i for Insert). Then you can enter the text. It will be inserted into the file. Do not worry if you make mistakes; you can correct them later. To enter the following programmer's limerick, this is what you type:
iA very intelligent turtle Found programming UNIX a hurdle After typing "turtle" you press the key to start a new line. Finally you press the key to stop Insert mode and go back to Normal mode. You now have two lines of text in your Vim window:
+————————————+ |A very intelligent turtle | |Found programming UNIX a hurdle | | | | | | | | +————————————+ WHAT IS THE MODE?
To be able to see what mode you are in, type this command:
:set showmode You will notice that when typing the colon Vim moves the cursor to the last line of the window. That's where you type colon commands (commands that start with a colon). Finish this command by pressing the <Enter> key (all commands that start with a colon are finished this way). Now, if you type the "i" command Vim will display –INSERT– at the bottom of the window. This indicates you are in Insert mode.
+————————————+ |A very intelligent turtle | |Found programming UNIX a hurdle | | | | | |– INSERT – | +————————————+ If you press <Esc> to go back to Normal mode the last line will be made blank.

GETTING OUT OF TROUBLE
One of the problems for Vim novices is mode confusion, which is caused by forgetting which mode you are in or by accidentally typing a command that switches modes. To get back to Normal mode, no matter what mode you are in, press the key. Sometimes you have to press it twice. If Vim beeps back at you, you already are in Normal mode.
==
Moving around After you return to Normal mode, you can move around by using these keys:
h left *hjkl* j down k up l right At first, it may appear that these commands were chosen at random. After all, who ever heard of using l for right? But actually, there is a very good reason for these choices: Moving the cursor is the most common thing you do in an editor, and these keys are on the home row of your right hand. In other words, these commands are placed where you can type them the fastest (especially when you type with ten fingers).
Note: You can also move the cursor by using the arrow keys. If you do, however, you greatly slow down your editing because to press the arrow keys, you must move your hand from the text keys to the arrow keys. Considering that you might be doing it hundreds of times an hour, this can take a significant amount of time. Also, there are keyboards which do not have arrow keys, or which locate them in unusual places; therefore, knowing the use of the hjkl keys helps in those situations. One way to remember these commands is that h is on the left, l is on the right and j points down. In a picture:
k h l j The best way to learn these commands is by using them. Use the "i" command to insert some more lines of text. Then use the hjkl keys to move around and insert a word somewhere. Don't forget to press to go back to Normal mode. The |vimtutor| is also a nice way to learn by doing.
For Japanese users, Hiroshi Iwatani suggested using this:
Komsomolsk $^{|}HuanHo <------> LosAngeles(Yellowriver)|vJava(theisland, nottheprogramminglan$
Deleting characters To delete a character, move the cursor over it and type "x". (This is a throwback to the old days of the typewriter, when you deleted things by typing xxxx over them.) Move the cursor to the beginning of the first line, for example, and type xxxxxxx (seven x's) to delete "A very ". The result should look like this:
+———————————————+ |intelligent turtle | |Found programming UNIX a hurdle | | | | | | | | +———————————————+ Now you can insert new text, for example by typing:
iA young <Esc> This begins an insert (the i), inserts the words "A young", and then exits insert mode (the final). The result:
+———————————————+ |A young intelligent turtle | |Found programming UNIX a hurdle | | | | | | | | +———————————————+ DELETING A LINE
To delete a whole line use the "dd" command. The following line will then move up to fill the gap:
+———————————————+ |Found programming UNIX a hurdle | | | | | | | | | +———————————————+ DELETING A LINE BREAK
In Vim you can join two lines together, which means that the line break between them is deleted. The "J" command does this. Take these two lines:
A young intelligent turtle Move the cursor to the first line and press "J":
A young intelligent turtle ===

Undo and Redo Suppose you delete too much. Well, you can type it in again, but an easier way exists. The "u" command undoes the last edit. Take a look at this in action: After using "dd" to delete the first line, "u" brings it back. Another one: Move the cursor to the A in the first line:
A young intelligent turtle Now type xxxxxxx to delete "A young". The result is as follows:
intelligent turtle Type "u" to undo the last delete. That delete removed the g, so the undo restores the character.
g intelligent turtle The next u command restores the next-to-last character deleted:
ng intelligent turtle The next u command gives you the u, and so on:
ung intelligent turtle oung intelligent turtle young intelligent turtle young intelligent turtle A young intelligent turtle
Note: If you type "u" twice, and the result is that you get the same text back, you have Vim configured to work Vi compatible. Look here to fix this: |not-compatible|. This text assumes you work "The Vim Way". You might prefer to use the good old Vi way, but you will have to watch out for small differences in the text then. REDO
If you undo too many times, you can press CTRL-R (redo) to reverse the preceding command. In other words, it undoes the undo. To see this in action, press CTRL-R twice. The character A and the space after it disappear:
young intelligent turtle There's a special version of the undo command, the "U" (undo line) command. The undo line command undoes all the changes made on the last line that was edited. Typing this command twice cancels the preceding "U".
A very intelligent turtle xxxx Delete very
A intelligent turtle xxxxxx Delete turtle
A intelligent Restore line with "U" A very intelligent turtle Undo "U" with "u"
A intelligent The "U" command is a change by itself, which the "u" command undoes and CTRL-R redoes. This might be a bit confusing. Don't worry, with "u" and CTRL-R you can go to any of the situations you had. More about that in section |32.2|.
Reference: http://vimdoc.sourceforge.net/htmldoc/$usr_02.html$

11.2 Virtual Box

Virtual Box Export and Import VirtualBox VM images? Export Open VirtualBox and enter into the File option to choice Export Appliance...
You will then get an assistance window to help you generating the image.
Select the VM to export Enter the output file path and name
You can choice a format, which I always leave the default OVF 1.
Finally you can write metadata like Version and Description Now you have an OVA file that you can carry to whatever machine to use it.
Import Open VirtualBox and enter into the File option to choice Import
You will then get an assistance window to help you loading the image.
Enter the path to the file that you have previously exported
Then you can modify the settings of the VM like RAM size, CPU, etc.
My recommendation on this is to enable the Reinitialize the MAC address of all the network cards option

Press Import and done! Now you have cloned the VM from the host machine into another one
Reference: https://askubuntu.com/questions/588426/how-to-export-and-import-virtualbox-vm-images
Install Guest Additions Guest Additions installs on the guest system and includes device drivers and system applications that optimize performance of the machine. Launch the guest OS in VirtualBox and click on Devices and Install Guest Additions.
The AutoPlay window opens on the guest OS and click on the Run VBox Windows Additions executable.
Click yes when the UAC screen comes up.
Now simply follow through the installation wizard.
During the installation wizard you can choose the Direct3D acceleration if you would like it. Remember this is going to take up more of your Host OS's resources and is still experimental possibly making the guest unstable.
When the installation starts you will need to allow the Sun display adapters to be installed.
After everything has completed a reboot is required.

11.3 VMWare

VMWare VMware Workstation is a program that allows you to run a virtual computer within your physical computer. The virtual computer runs as if it was its own machine. A virtual machine is great for trying out new operating systems such as Linux, visiting websites you don't trust, creating a computing environment specifically for children, testing the effects of computer viruses, and much more. You can even print and plug in USB drives. Read this guide to get the most out of VMware Workstation.
Installing VMware Workstation
1. Make sure your computer meets the system requirements. Because you will be running an operating system from within your own operating system, VMware Workstation has fairly high system requirements. If you don't meet these, you may not be able to run VMware effectively. You must have a 64-bit processor. VMware supports Windows and Linux operating systems. You must have enough memory to run your operating system, the virtual operating system, and any programs inside that operating system. 1 GB is the minimum, but 3 or more is recommended. You must have a 16-bit or 32-bit display adapter. 3D effects will most likely not work well inside the virtual operating system, so gaming is not always efficient. You need at least 1.5 GB of free space to install VMware Workstation, along with at least 1 GB per operating system that you install.
2. Download the VMware software. You can download the VMware installer from the Download Center on the VMware website. Select the newest version and click the link for the installer. You will need to login with your VMware username. You will be asked to read and review the license agreement before you can download the file. You can only have one version of VMware Workstation installed at a time.
3. Install VMware Workstation. Once you have downloaded the file, right-click on the file and select "Run as administrator". You will be asked to review the license again. Most users can use the Typical installation option. At the end of

the installation, you will be prompted for your license key. Once the installation is finished, restart the computer. Part
Installing an Operating System
1. Open VMware. Installing a virtual operating system is much like installing it on a regular PC. You will need to have the installation disc or ISO image as well as any necessary licenses for the operating system that you want to install. You can install most distributions of Linux as well as any version of Windows.
2. Click File. Select New Virtual Machine and then choose Typical. VMware will prompt you for the installation media. If it recognizes the operating system, it will enable Easy Installation:
Physical disc – Insert the installation disc for the operating system you want to install and then select the drive in VMware. ISO image – Browse to the location of the ISO file on your computer. Install operating system later. This will create a blank virtual disk. You will need to manually install the operating system later.
3. Enter in the details for the operating system. For Windows and other licensed operating systems, you will need to enter your product key. You will also need to enter your preferred username and a password if you want one. * If you are not using Easy Install, you will need to browse the list for the operating system you are installing.
4. Name your virtual machine. The name will help you identify it on your physical computer. It will also help distinguish between multiple virtual computers running different operating systems.
5. Set the disk size. You can allocate any amount of free space on your computer to the virtual machine to act as the installed operating system's hard drive. Make sure to set enough to install any programs that you want to run in the virtual machine.
6. Customize your virtual machine's virtual hardware. You can set the virtual machine to emulate specific hardware by clicking the "Customize Hardware" button. This can be useful if you are trying to run an older program that only supports certain hardware. Setting this is optional.
7. Set the virtual machine to start. Check the box labeled "Power on this virtual machine after creation" if you want the virtual machine to start up as soon as you finish making it. If you don't check this box, you can select your virtual machine from the list in VMware and click the Power On button.
8. Wait for your installation to complete. Once you've powered on the virtual machine for the first time, the operating system will begin to install automatically. If you provided all of the correct information during the setup of the virtual machine, then you should not have to do anything. If you didn't enter your product key or create a username during the virtual machine setup, you will most likely be prompted during the installation of the operating system.
9. Check that VMware Tools is installed. Once the operating system is installed, the program VMware Tools should be automatically installed. Check that it appears on the desktop or in the program files for the newly installed operating system.
VMware tools are configuration options for your virtual machine, and keeps your virtual machine up to date with any software changes.
Navigating VMware
1. Start a virtual machine. To start a virtual machine, click the VM menu and select the virtual machine that you want to turn on. You can choose to start

the virtual machine normally, or boot directly to the virtual BIOS.
2. Stop a virtual machine. To stop a virtual machine, select it and then click the VM menu. Select the Power option.
Power Off – The virtual machine turns off as if the power was cut out. Shut Down Guest – This sends a shutdown signal to the virtual machine which causes the virtual machine to shut down as if you had selected the shutdown option. You can also turn off the virtual machine by using the shutdown option in the virtual operating system.
3. Move files between the virtual machine and your physical computer. Moving files between your computer and the virtual machine is as simple as dragging and dropping. Files can be moved in both directions between the computer and the virtual machine, and can also be dragged from one virtual machine to another.
When you drag and drop, the original will stay in the original location and a copy will be created in the new location. You can also move files by copying and pasting. Virtual machines can connect to shared folders as well.
4. Add a printer to your virtual machine. You can add any printer to your virtual machine without having to install any extra drivers, as long as it is already installed on your host computer.
Select the virtual machine that you want to add the printer to. Click the VM menu and select Settings. Click the Hardware tab, and then click Add. This will start the Add Hardware wizard. Select Printer and then click Finish. Your virtual printer will be enabled the next time you turn the virtual machine on.
5. Connect a USB drive to the virtual machine. Virtual machines can interact with a USB drive the same way that your normal operating system does. The USB drive cannot be accessed on both the host computer and the virtual machine at the same time.
If the virtual machine is the active window, the USB drive will be automatically connected to the virtual machine when it is plugged in. If the virtual machine is not the active window or is not running, select the virtual machine and click the VM menu. Select Removable Devices and then click Connect. The USB drive will automatically connect to your virtual machine.
6. Take a snapshot of a virtual machine. A snapshot is a saved state and will allow you to load the virtual machine to that precise moment as many times as you need.
Select your virtual machine, click the VM menu, hover over Snapshot and select Take Snapshot. Give your Snapshot a name. You can also give it a description, though this is optional. Click OK to save the Snapshot. Load a saved Snapshot by clicking the VM menu and then selecting Snapshot. Choose the Snapshot you wish to load from the list and click Go To.
7. Become familiar with keyboard shortcuts. A combination of the "Ctrl" and other keys are used to navigate virtual machines. For example, "Ctrl," "Alt" and "Enter" puts the current virtual machine in full screen mode or moves through multiple machines. "Ctrl," "Alt" and "Tab" will move between virtual machines when the mouse is being used by 1 machine.

Phần II

Khoa học dữ liệu

Chương 12

Học sâu

Tài liệu cũ: http://magizbox.com/training/deep_learning/site/
Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy.

12.1 Deep Feedforward Networks

deep feedforward networks

Deep feedforward networks, (hay còn gọi là feedforward neural networks) hoặc multilayer perceptrons (MLPs) là mô hình deep learning cơ bản nhất. Mục tiêu của một feedforward network là xấp xỉ một hàm $f*$. Ví dụ, với bài toán phân loại, $y = f*(x)$ chuyển đầu vào x thành một nhãn y. Một feedforward network định nghĩa một ánh xạ $y = f(x, \theta)$ và học giá trị của θ để xấp xỉ hàm $f*$ tốt nhất.

Mô hình được gọi là feedforward vì giá trị của x đang lan truyền qua mạng đến output, mà không có chiều ngược lại. Các mô hình neural có sự lan truyền ngược lại được gọi là recurrent neural network.

Feedforward là nền tảng cho rất nhiều mô hình mạng neural hiện đại. Ví dụ, các ứng dụng trong việc nhận diện ảnh thường áp dụng mô hình convolutional, là một dạng đặc biệt của feedforward network. Feedforward network cũng là nền tảng cho mạng recurrent network, có nhiều ứng trọng xử lý ngôn ngữ.

Feedforward neural networks được gọi là network vì nó thường được biểu diễn bởi một đồ thị gồm nhiều tầng, định nghĩa các hàm được kết hợp với nhau như thế nào. Ví dụ, chúng ta có 3 hàm f(1)(x), f(2)(x), f(3)(x) được đặt với nhau thành một chuỗi, hình thành hàm f(x) = f(3)(f(2)(f(1)(x))). Cấu trúc chuỗi này là dạng phổ biến nhất trong mạng neural. Trong trường hợp này f(1)(x) được gọi là layer 1, f(2)(x) được gọi là layer 2, và cứ tiếp tục như vậy. Độ dài của chuỗi thể hiện độ sâu của mô hình. Thuật ngữ "deep learning" xuất phát từ điều này. Layer cuối cũng được goi là output layer. Trong mạng neural, mục tiêu là học hàm f*(x). Ứng với mỗi giá trị đầu vào x, sẽ có tương ứng một giá trị y. Mục tiêu của mạng là tìm các tham số để y xấp xỉ với f*(x). Nhưng việc

học của hàm không hoàn toàn phụ thuộc vào x, y, mà do learning algorithm quyết định. Vì dữ liệu huấn luyện không chỉ rõ giá trị ở những layer ở giữa, nên chúng được gọi là các hidden layers Cuối cùng, mạng neural được gọi là neural vì nó lấy cảm hứng từ ngành khoa học thần kinh. Mỗi hidden layer trong mạng thường là các giá trị vector. Mỗi phần tử trong vector sẽ quyết định việc hành xử thế nào với các input đầu vào và đưa ra output.
Ta có thể coi mỗi layer chứa rất nhiều units hoạt động song song, đóng vai trò như một hàm ánh xạ vector sang giá trị số. Mặc dù các kiến trúc của mạng neural lấy rất nhiều ý tưởng từ ngành khoa học thần kinh, tuy nhiên, mục tiêu của mạng neural không phải để mô phỏng bộ não. Một cách nhìn tốt hơn là xem mạng neural như một máy học, được thiết kế với các điểm nhấn từ những gi chúng ta biết về não người. Một cách để hiểu mạng neural là bắt đầu với các mô hình linear, và xem xét các hạn chế của các mô hình này. Linear models, như logistic regression hay linear regression, hấp dẫn ở chỗ chúng có thể fit rất hữu quả và đang tin cậy, với các close form và tối ưu hóa lồi. Một hàn chế hiển nhiên của linear là các hàm linear, nên model không thể hiểu tương tác giữa các input. Để mở rộng lienar model để biểu diễn hàm nonlinear của x, chúng ta có thể áp dụng linear model không phải cho x mà cho (x), trong đó là một nonlinear transformer. Có thể xem như một tập các feature mô tả x, hoặc một cách để biểu diễn x. Vấn đề là chọn mapping . Có 3 cách thông dụng, chọn một generic như RBF, thiết kế bằng tay, hoặc học .
Nội dung chương này:

- 1. Ví dụ cơ bản củ feedforward network
- 2. Design decisions cho việc thiết kế mạng feedforward network
- 3. Choose activation functions trong hidden layer
- 4. Thiết kế mạng neural, bao nhiêu layers, các layer kết nối với nhau như thế nào, có bao nhiêu unit trong một layer
  - chosing the optimizer
  - cost function
  - form of the output units
- 5. Thuật toán Back-propagation
- 6. Góc nhìn lịch sử

Tham khảo Chapter 6, Deep Learning Book Neural Network Zoo http://www.asimovinstitute.org/neural-network-zoo/
word2vec Example Step 1: Download word2vec example from github
dir
02/06/2017 11:45 DIR . 02/06/2017 11:45 DIR .. 02/06/2017 10:12 9,144 word2vec$_basic.pyStep2 :$
$Runword2vec_basicexample$
$activatetensorflow - gpu$ python word2vec$_basic.py$
Found and verified text8.zip Data size 17005207 Most common words (+UNK) [['UNK', 418391], ('the', 1061396), ('of', 593677), ('and', 416629), ('one', 411764)] Sample data [5241, 3082, 12, 6, 195, 2, 3136, 46, 59, 156] ['anarchism', 'originated', 'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against'] 3082 originated -> 5241 anarchism 3082 originated -> 12 as 12 as -> 6 a 12 as -> 3082 originated

6 a -> 195 term 6 a -> 12 as 195 term -> 2 of 195 term -> 6 a Initialized Average loss at step 0 : 288.173675537 Nearest to its: nasl, tinkering, derivational, yachts, emigrated, fatalism, kingston, kochi, Nearest to into: streetcars, neglecting, deutschlands, lecture, realignment, bligh, donau, medalists, Nearest to state: canterbury, exceptions, disaffection, crete, westernmost, earthly, organize, richland,

12.2 Tài liệu Deep Learning

Lang thang thế nào lại thấy trang này My Reading List for Deep Learning! của một anh ở Microsoft. Trong đó, (đương nhiên) có Deep Learning của thánh Yoshua Bengio, có một vụ hay nữa là bài review "Deep Learning" của mấy thánh Yann Lecun, Yoshua Bengio, Geoffrey Hinton trên tạp chí Nature. Ngoài ra còn có nhiều tài liệu hữu ích khác.

12.3 Các layer trong deep learning

12.3.1 Sparse Layers

nn.Embedding (hướng dẫn)
grep code: Shawn1993/cnn-text-classification-pytorch
Đóng vai trò như một lookup table, map một word với dense vector tương ứng

12.3.2 Convolution Layers

nn.Conv1d, nn.Conv2d, nn.Conv3d)
grep code: Shawn1993/cnn-text-classification-pytorch, galsang/CNN-sentence-classification-pytorch
Các tham số trong Convolution Layer
* *kernel_size* (hay là filter size)
Đối với NLP, *kernel_size* thường bằng *region_size* ∗ *word_dim* (đối với conv1d) hay (*region_size*, *word_dim*) đối với conv2d
<small>Quá trình tạo feature map đối với region size bằng 2</small> ![](https://media.giphy.com/media/l2Q
* ‘in$_c$*hannels*‘, ‘*out$_c$hannels*‘(*lslng*‘*featuremaps*‘)
Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.
* ‘stride‘
Định nghĩa bước nhảy của filter.
![](http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/Screen-Shot-2015-11-05-at-10.18.08-AM-1024x251.png)
Hình minh họa sự khác biệt giữa các feature map đối với stride=1 và stride=2. Feature map đối với stride = 1 có kích thước là 5, feature map đối với stride = 3 có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.
Trong bài báo của Kim 2014, ‘stride = 1‘ đối với ‘nn.conv2d‘ và ‘stride = word$_d$*im*‘*Øivi*‘*nn.conv*1*d*‘
Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

![](http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/Screen-Shot-2015-11-06-at-8.03.47-AM.png)
| Description | Values | |——————|—————| | input word vectors | Google word2vec | | filter region size | (3, 4, 5) | | feature maps | 100 | | activation function | ReLU | | pooling | 1-max pooling | | dropout rate | 0.5 | | *latexlamp*; $s =$ 22 norm constraint | 3 |
Đọc thêm:
* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017](http://web.stanford.edu/class/cs224n 2017-lecture13-CNNs.pdf) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks](https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015](https://arxiv.org/pdf/1510.03820.pdf)

12.4 Recurrent Neural Networks

12.4.1 What are RNNs?

The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:
A recurrent neural network and the unfolding in time of the computation involved in its forward computation
A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature The above diagram shows a RNN being unrolled (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in a RNN are as follows:
xtxt is the input at time step tt. For example, x1x1 could be a one-hot vector corresponding to the second word of a sentence. stst is the hidden state at time step tt. It's the "memory" of the network. stst is calculated based on the previous hidden state and the input at the current step: st=f(Uxt+Wst1)st=f(Uxt+Wst1). The function ff usually is a nonlinearity such as tanh or ReLU. s1s1, which is required to calculate the first hidden state, is typically initialized to all zeroes. otot is the output at step tt. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. ot=softmax(Vst)ot=softmax(Vst). There are a few things to note here:
You can think of the hidden state stst as the memory of the network. stst captures information about what happened in all the previous time steps. The

output at step otot is calculated solely based on the memory at time tt. As briefly mentioned above, it's a bit more complicated in practice because stst typically can't capture information from too many time steps ago. Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (UU, VV, WW above) across all steps. This reflects the fact that we are performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters we need to learn. The above diagram has outputs at each time step, but depending on the task this may not be necessary. For example, when predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word. Similarly, we may not need inputs at each time step. The main feature of an RNN is its hidden state, which captures some information about a sequence.

12.4.2 What can RNNs do?

RNNs have shown great success in many NLP tasks. At this point I should mention that the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. But don't worry, LSTMs are essentially the same thing as the RNN we will develop in this tutorial, they just have a different way of computing the hidden state. We'll cover LSTMs in more detail in a later post. Here are some example applications of RNNs in NLP (by non means an exhaustive list).
Language Modeling and Generating Text Given a sequence of words we want to predict the probability of each word given the previous words. Language Models allow us to measure how likely a sentence is, which is an important input for Machine Translation (since high-probability sentences are typically correct). A side-effect of being able to predict the next word is that we get a generative model, which allows us to generate new text by sampling from the output probabilities. And depending on what our training data is we can generate all kinds of stuff. In Language Modeling our input is typically a sequence of words (encoded as one-hot vectors for example), and our output is the sequence of predicted words. When training the network we set ot=xt+1ot=xt+1 since we want the output at step tt to be the actual next word.
Research papers about Language Modeling and Generating Text:
Recurrent neural network based language model Extensions of Recurrent neural network based language model Generating Text with Recurrent Neural Networks Machine Translation Machine Translation is similar to language modeling in that our input is a sequence of words in our source language (e.g. German). We want to output a sequence of words in our target language (e.g. English). A key difference is that our output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.
RNN for Machine Translation
RNN for Machine Translation. Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf
Research papers about Machine Translation:
A Recursive Recurrent Neural Network for Statistical Machine Translation Sequence to Sequence Learning with Neural Networks Joint Language and Translation Modeling with Recurrent Neural Networks Speech Recognition Given an

input sequence of acoustic signals from a sound wave, we can predict a sequence of phonetic segments together with their probabilities.
Research papers about Speech Recognition:
Towards End-to-End Speech Recognition with Recurrent Neural Networks Generating Image Descriptions Together with convolutional Neural Networks, RNNs have been used as part of a model to generate descriptions for unlabeled images. It's quite amazing how well this seems to work. The combined model even aligns the generated words with features found in the images.
Deep Visual-Semantic Alignments for Generating Image Descriptions. Source: http://cs.stanford.edu/people/karpathy/deepimagesent/

12.4.3 Training RNNs

Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm, but with a little twist. Because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps. For example, in order to calculate the gradient at t=4t=4 we would need to backpropagate 3 steps and sum up the gradients. This is called Backpropagation Through Time (BPTT). If this doesn't make a whole lot of sense yet, don't worry, we'll have a whole post on the gory details. For now, just be aware of the fact that vanilla RNNs trained with BPTT have difficulties learning long-term dependencies (e.g. dependencies between steps that are far apart) due to what is called the vanishing/exploding gradient problem. There exists some machinery to deal with these problems, and certain types of RNNs (like LSTMs) were specifically designed to get around them.
RNN Extensions Over the years researchers have developed more sophisticated types of RNNs to deal with some of the shortcomings of the vanilla RNN model. We will cover them in more detail in a later post, but I want this section to serve as a brief overview so that you are familiar with the taxonomy of models.
Bidirectional RNNs are based on the idea that the output at time t may not only depend on the previous elements in the sequence, but also future elements. For example, to predict a missing word in a sequence you want to look at both the left and the right context. Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs.
Deep (Bidirectional) RNNs are similar to Bidirectional RNNs, only that we now have multiple layers per time step. In practice this gives us a higher learning capacity (but we also need a lot of training data).
Deep Bidirectional RNN LSTM networks are quite popular these days and we briefly talked about them above. LSTMs don't have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called cells and you can think of them as black boxes that take as input the previous state ht1ht1 and current input xtxt. Internally these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies. LSTMs can be quite confusing in the beginning but if you're interested in learning more this post has an excellent explanation.

Conclusion So far so good. I hope you've gotten a basic understanding of what RNNs are and what they can do. In the next post we'll implement a first version of our language model RNN using Python and Theano. Please leave questions in the comments!
$[^1] : [UnderstandingConvolutionalNeuralNetworksforNLP](http : //www.wildml.com/2015/11/understa$
$convolutional-neural-networks-for-nlp)[^2] : [http : //pytorch.org/docs/master/nn.html](http :$
$//pytorch.org/docs/master/nn.html)$

Chương 13

Pytorch

Sau khi nghe bài này thì hâm mộ luôn anh Soumith Chintala, tìm loạt bài anh trình bày luôn
* [PyTorch: Fast Differentiable Dynamic Graphs in Python with a Tensor JIT](https://www.youtube.com/watch?v=DBVLcgq2Eg0amp;t=2s), Strange Loop Sep 2017 * [Keynote: PyTorch: Framework for fast, dynamic deep learning and scientific computing](https://www.youtube.com/watch?v=LAMwEJZqesUamp;t=66s), EuroSciPy Aug 2017

08/12/2017: Cảm giác talk của anh Soumith Chintala (Facebook, research editor) ở Deep Learning Summer School 2017 khá thú vị

13.1 So sánh giữa Tensorflow và Pytorch?

Có 2 điều cần phải nói khi mọi người luôn luôn so sánh giữa Tensorflow và Pytorch. (1) Tensorflow khiến mọi người "không thoải mái" (2) Pytorch thực sự là một đối thủ trên bàn cân. Một trong những câu trả lời hay nhất mình tìm được là của anh Hieu Pham (Google Brain) [trả lời trên quora (25/11/2017)](https://www.quora.com/What-are-your-reviews-between-PyTorch-and-TensorFlow/answer/Hieu-Pham-20?srid=5O2u). Điều quan trọng nhất trong câu trả lời này là *"Dùng Pytorch rất sướng cho nghiên cứu, nhưng scale lên mức business thì Tensorflow là lựa chọn tốt hơn"*
Behind The Scene
(15/11/2017) Hôm nay bắt đầu thử nghiệm pytorch với project thần thánh classification sử dụng cnn https://github.com/Shawn1993/cnn-text-classification-pytorch
Cảm giác đầu tiên là make it run khá đơn giản
“‘ conda create -n test-torch python=3.5 pip install http://download.pytorch.org/whl/cu80/torch-0.2.0.post3-cp35-cp35m-manylinux1$_x86_64.whlpipinstalltorchvisionpipinstalltorchtext$“‘
Thế là ‘main.py‘ chạy! Hay thật. Còn phải vọc để bạn này chạy với CUDA nữa.
Cài đặt CUDA trong ubuntu 16.04
Kiểm tra VGA
“‘ $lspci|grepVGA01:00.0VGAcompatiblecontroller:NVIDIACorporationGM204[GeForceGTX980](rev$
Kiểm tra CUDA đã cài đặt trong Ubuntu [1]
“‘ $nvcc--versionnvcc:NVIDIA(R)CudacompilerdriverCopyright(c)2005-2016NVIDIACorporationBuiltonSun_Sep_42 2{:}14{:}01_CDT_2016Cudacompilationtools,release8.0,V8.0.44$“‘
Kiểm tra pytorch chạy với cuda ‘test$_cuda.py$‘

"'python import torch print("Cuda:", torch.cuda.is$_a vailable())$"'
"' $pythontest_c uda.pyCUDA : True$"'
Chỉ cần cài đặt thành công CUDA là pytorch tự work luôn. Ngon thật!
Ngày X
Chẳng hiểu sao update system kiểu nào mà hôm nay lại không sử dụng được CUDA 'torch.cuda.is$_a vailable() = False$'$.Saukhidnglnh$'$torch.Tensor().cuda()$'$thgpli$
"' AssertionError: The NVIDIA driver on your system is too old (found version 8000). Please update your GPU driver by downloading and installing a new version from the URL: http://www.nvidia.com/Download/index.aspx Alternatively, go to: https://pytorch.org/binaries to install a PyTorch version that has been compiled with your version of the CUDA driver. "'
Kiểm tra lại thì mình đang dùng nvidia-361, làm thử theo [link này](http://www.linuxandubuntu.com/home/h
to-install-latest-nvidia-drivers-in-linux) để update NVIDIA, chưa biết kết quả ra sao?
May quá, sau khi update lên nvida-387 là ok. Haha
Ngày 2
Hôm qua đã bắt đầu implement một nn với pytorch rồi. Hướng dẫn ở [Deep Learning with PyTorch: A 60 Minute Blitz](http://pytorch.org/tutorials/beginner/deep$_l earning_6 0min_b litz.h$
Hướng dẫn implement các mạng neural với pytorch rất hay tại [PyTorch-Tutorial](https://github.com/MorvanZhou/PyTorch-Tutorial)
(lượm lặt) Trang này [Awesome-pytorch-list](https://github.com/bharathgs/Awesome-pytorch-list) chứa rất nhiều link hay về pytorch như tập hợp các thư viện liên quan, các hướng dẫn và ví dụ sau đó là các cài đặt của các paper sử dụng pytorch.
(lượm lặt) Loạt video hướng dẫn pytorch [PyTorchZeroToAll](https://www.youtube.com/watch?v=SKq-pmkekTkamp;list=PLlMkM4tgfjnJ3I-dbhO9JTw7gNty6o$_2 m)catcgiSungKimtrnyoutube.$
Bước tiếp theo là visualize loss và graph trong tensorboard, sử dụng [tensorboard$_l ogger](https : //github.com/TeamHG - Memex/tensorboard_l ogger)khhay.$
"' pip install tensorboard$_l oggerpipinstalltensorboard$"'
Chạy tensorboard server
"' tensorboard –log-dir=runs "'
Ngày 3: Vấn đề kỹ thuật
Hôm qua cố gắng implement một phần thuật toán CNN cho bài toán phân lớp văn bản. Vấn đề đầu tiên là biểu diễn sentence thế nào. Cảm giác load word vector vào khá chậm. Mà thằng tách từ của underthesea cũng chậm kinh khủng.
Một vài link tham khảo về bài toán CNN: [Implementing a CNN for Text Classification in TensorFlow](http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/), [Text classification using CNN : Example](https://agarnitin86.github.io/blog/2016/12/23/text-classification-cnn)
[^{1}] : $https : //askubuntu.com/questions/799184/how - can - i - install - cuda - on - ubuntu - 16 - 04$

Chương 14

Tensorflow

14.1 Cài đặt Tensorflow trong Windows

Step 1: Download the Anaconda installer
Step 2: Double click the Anaconda installer and follow the prompts to install to the default location.
After a successful installation you will see output like this:
CUDA Toolkit 8.0
The NVIDIA CUDA Toolkit provides a comprehensive development environment for C and C++ developers building GPU-accelerated applications. The CUDA Toolkit includes a compiler for NVIDIA GPUs, math libraries, and tools for debugging and optimizing the performance of your applications. You'll also find programming guides, user manuals, API reference, and other documentation to help you get started quickly accelerating your application with GPUs.
Step 1: Verify the system has a CUDA-capable GPU.
Step 2: Download the NVIDIA CUDA Toolkit.
Step 3: Install the NVIDIA CUDA Toolkit.
Step 4: Test that the installed software runs correctly and communicates with the hardware.

```
> nvcc -V

nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2016 NVIDIA Corporation
Built on Tue_Jan_10_13:22:03_CST_2017
Cuda compilation tools, release 8.0, V8.0.61
```

cuDNN
The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. cuDNN is part of the NVIDIA Deep Learning SDK.
Step 1: Register an NVIDIA developer account
Step 2: Download cuDNN v5.1, you will get file like that cudnn-8.0-windows7-x64-v5.1.zip
Step 3: Copy CUDNN files to CUDA install

Extract your cudnn-8.0-windows7-x64-v5.1.zip file, and copy files to corresponding CUDA folder
In my environment, CUDA installed in C:
Program Files
NVIDIA GPU Computing Toolkit
CUDA
v8.0, you must copy append three folders bin, include, lib
Install Tensorflow Package
CPU TensorFlow environment

```
conda create --name tensorflow python=3.5
```

activate tensorflow

```
conda install -y jupyter scipy
pip install tensorflow
```

GPU TensorFlow environment

```
conda create --name tensorflow-gpu python=3.5
activate tensorflow-gpu
conda install -y jupyter scipy
pip install tensorflow-gpu
```

14.2 Cài đặt tensorflow trong Ubuntu 16.04

Làm theo hướng dẫn rất hay trong bài viết Install CUDA Toolkit v8.0 and cuDNN v6.0 on Ubuntu 16.04
Cài đặt CUDA 8.0
Kiểm tra CUDA sau khi cài đặt

```
> nvcc
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005 2016 NVIDIA Corporation
Built on Tue_Jan_10_13:22:03_CST_2017
Cuda compilation tools, release 8.0, V8.0.61
```

Cài đặt cuDNN v6.0

Tài liệu tham khảo

Chỉ mục

convolution, 129

deep feedforward networks, 127

Ghi chú

▨ 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning . . . 127

▨ 22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. . . . 127

▨ 08/12/2017: Cảm giác talk của anh Soumith Chintala (Facebook, research editor) ở Deep Learning Summer School 2017 khá thú vị . . . 134