

# Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

# Mục lục

Mục lục	6
1 Lời nói đầu	7
I Lập trình	9
2 Giới thiệu	10
2.1 Các vấn đề lập trình	10
2.1.1 Introduction	10
2.1.2 Data Structure	11
2.1.3 OOP	11
2.1.4 Networking	11
2.1.5 Sample Project Ideas	11
2.2 How to ask a question	12
2.3 Các vấn đề lập trình	12
2.4 Các mô hình lập trình	12
2.5 Testing	14
2.6 Logging	16
2.7 Lập trình hàm	17
2.8 Lập trình song song	18
2.9 IDE	19
3 Python	20
3.1 Giới thiệu	20
3.2 Cài đặt	21
3.3 Cơ bản	22
3.4 Cú pháp cơ bản	22
3.5 Yield and Generators	24
3.6 Cấu trúc dữ liệu	28
3.6.1 Number	28
3.6.2 Collection	29
3.6.3 String	31
3.6.4 Datetime	32
3.6.5 Object	33
3.7 Object Oriented Programming	33
3.7.1 Metaclasses	35
3.7.2 Design Patterns	38

3.8	File System IO	39
3.9	Operating System	40
3.10	Networking	40
3.11	Concurrency and Parallelism	40
3.12	Event Based Programming	42
3.13	Web Development	43
3.14	Logging	45
3.15	Configuration	46
3.16	Command Line	46
3.17	Testing	46
3.18	IDE Debugging	47
3.19	Package Manager	49
3.20	Environment	50
3.21	Module	52
3.22	Production	54
3.23	Quản lý gói với Anaconda	54
3.24	Test với python	55
3.25	Xây dựng docs với readthedocs và sphinx	55
3.26	Pycharm Pycharm	58
3.27	Vì sao lại code python?	58
<b>4</b>	<b>C++</b>	<b>59</b>
4.1	Get Started	59
4.2	Basic Syntax	60
4.3	Cấu trúc dữ liệu	60
4.4	Lập trình hướng đối tượng	62
4.5	Cơ sở dữ liệu	62
4.6	Testing	63
4.7	IDE Debugging	64
<b>5</b>	<b>Javascript</b>	<b>65</b>
5.1	Installation	65
5.2	IDE	65
5.3	Basic Syntax	65
5.4	Data Structure	67
5.4.1	Number	67
5.4.2	String	68
5.4.3	Collection	70
5.4.4	Datetime	70
5.4.5	Boolean	70
5.4.6	Object	70
5.5	OOP	70
5.6	Networking	72
5.7	Logging	72
5.8	Documentation	72
5.9	Error Handling	73
5.10	Testing	75
5.11	Package Manager	75
5.12	Build Tool	76
5.13	Make Module	76

<b>6</b>	<b>Java</b>	<b>77</b>
6.1	Get Started	77
6.2	Basic Syntax	77
6.3	Data Structure	82
6.4	OOP	82
6.4.1	Classes	82
6.4.2	Encapsulation	86
6.4.3	Inheritance	86
6.4.4	Polymorphism	88
6.4.5	Abstraction	90
6.5	File System IO	92
6.6	Error Handling	96
6.7	Logging	101
6.8	IDE	102
6.9	Package Manager	103
6.10	Build Tool	103
6.11	Production	103
<b>7</b>	<b>PHP</b>	<b>104</b>
<b>8</b>	<b>R</b>	<b>106</b>
8.1	R Courses	106
8.2	Everything you need to know about R	107
<b>9</b>	<b>Scala</b>	<b>109</b>
9.1	Installation	109
9.2	IDE	109
9.3	Basic Syntax	109
<b>10</b>	<b>NodeJS</b>	<b>111</b>
10.1	Get Started	111
10.2	Basic Syntax	112
10.3	File System IO	112
10.4	Package Manager	117
10.5	Command Line	118
<b>11</b>	<b>Octave</b>	<b>120</b>
11.1	Matrix	120
<b>12</b>	<b>Toolbox</b>	<b>122</b>
12.1	Vim	123
12.2	Virtual Box	125
12.3	VMWare	126
<b>II</b>	<b>Xác suất</b>	<b>129</b>
<b>13</b>	<b>Các hàm phân phối thông dụng</b>	<b>130</b>
13.0.1	Biến rời rạc	130

<b>III Khoa học máy tính</b>	<b>132</b>
<b>14 Data Structure and Algorithm</b>	<b>133</b>
14.1 Introduction . . . . .	133
14.1.1 Greedy Algorithm . . . . .	134
14.1.2 Divide and Conquer . . . . .	135
14.1.3 Dynamic Programming . . . . .	135
14.1.4 7 Steps to Solve Algorithm Problems . . . . .	136
14.2 Data Structures . . . . .	139
14.2.1 Array . . . . .	139
14.2.2 Linked List . . . . .	142
14.2.3 Stack and Queue . . . . .	143
14.2.4 Tree . . . . .	146
14.2.5 Binary Search Tree . . . . .	148
14.3 Heaps . . . . .	149
14.4 Sort . . . . .	150
14.4.1 Introduction . . . . .	150
14.4.2 Bubble Sort . . . . .	151
14.4.3 Insertion Sort . . . . .	152
14.4.4 Selection Sort . . . . .	153
14.4.5 Merge Sort . . . . .	154
14.4.6 Shell Sort . . . . .	155
14.4.7 Quick Sort . . . . .	156
14.5 Search . . . . .	157
14.5.1 Linear Search . . . . .	157
14.5.2 Binary Search . . . . .	157
14.5.3 Interpolation Search . . . . .	158
14.5.4 Hash Table . . . . .	159
14.6 Graph . . . . .	160
14.6.1 Graph Data Structure . . . . .	160
14.6.2 Depth First Traversal . . . . .	161
14.6.3 Breadth First Traversal . . . . .	162
14.7 String . . . . .	162
14.7.1 Tries . . . . .	166
14.7.2 Suffix Array and suffix tree . . . . .	168
14.7.3 Knuth-Morris-Pratt Algorithm . . . . .	169
<b>15 Object Oriented Programming</b>	<b>174</b>
15.1 OOP . . . . .	174
15.2 UML . . . . .	178
15.3 SOLID . . . . .	180
15.4 Design Patterns . . . . .	181
<b>16 Database</b>	<b>184</b>
16.1 Introduction . . . . .	184
16.2 SQL . . . . .	186
16.3 MySQL . . . . .	186
16.4 Redis . . . . .	187
16.5 MongoDB . . . . .	187

<i>MỤC LỤC</i>	5
<b>17 Hệ điều hành</b>	<b>189</b>
<b>18 Ubuntu</b>	<b>190</b>
<b>19 Networking</b>	<b>191</b>
<b>20 UX - UI</b>	<b>193</b>
<b>21 Service-Oriented Architecture</b>	<b>194</b>
<b>22 License</b>	<b>196</b>
<b>23 Semantic Web</b>	<b>197</b>
23.1 Web 3.0 . . . . .	197
23.2 RDF . . . . .	197
23.3 SPARQL . . . . .	197
 <b>IV Khoa học dữ liệu</b>	 <b>199</b>
<b>24 Học máy</b>	<b>200</b>
<b>25 Học sâu</b>	<b>203</b>
25.1 Tài liệu Deep Learning . . . . .	203
25.2 Các layer trong deep learning . . . . .	203
25.2.1 Sparse Layers . . . . .	203
25.2.2 Convolution Layers . . . . .	203
<b>26 Xử lý ngôn ngữ tự nhiên</b>	<b>205</b>
<b>27 Nhận dạng tiếng nói</b>	<b>206</b>
<b>28 Phân loại văn bản</b>	<b>210</b>
<b>29 Pytorch</b>	<b>211</b>
 <b>V Linh tinh</b>	 <b>213</b>
<b>30 Nghiên cứu</b>	<b>214</b>
<b>31 Nghề lập trình</b>	<b>216</b>
<b>32 Latex</b>	<b>217</b>
<b>33 Chào hàng</b>	<b>219</b>
<b>34 Phát triển phần mềm</b>	<b>220</b>
<b>35 Phương pháp làm việc</b>	<b>221</b>

<i>MỤC LỤC</i>	6
<b>Tài liệu</b>	<b>223</b>
<b>Chỉ mục</b>	<b>224</b>

# Chương 1

## Lời nói đầu

Đọc quyển Deep Learning quá xá hay luôn. Rồi lại đọc SLP 2. Thấy sao các thánh viết hay và chuẩn thể (đấy là lý do các thánh được gọi là ... các thánh chẳng =))

Tính đến thời điểm này đã được 2 năm 10 tháng rồi. Quay lại với latex. Thỏa mãn được điều kiện của mình là một tool offline. Mình thích xuất ra pdf (có gì đọc lại hoặc tra cứu cũng dễ).

Hi vọng gắn bó với thằng này được lâu.

**Chào từ hồi [magizbox.wordpress.com](https://magizbox.wordpress.com), cái này tồn tại được 77 ngày (hơn 2 tháng) (từ 01/11/2017 đến 17/01/2018)**

Chào Khách,

Mình là Vũ Anh. Tính đến thời điểm viết bài này thì đã lập trình được 7 năm (lập trình từ hồi năm 2010). Mình thích viết lách, bằng chứng là đã thay đổi host 2 lần [datayo.wordpress.com](https://datayo.wordpress.com), [magizbox.com](https://magizbox.com). Thành tựu ổn nhất hiện tại chỉ có một project [underthesea](https://github.com/magizbox/underthesea), xếp loại tạm được.

Blog này chứa những ghi chép loạn cào cào của mình về mọi thứ. Đúng với phong cách "vô tổ chức" của mình. Chắc chắn nó sẽ không hữu ích lắm với bạn. Tuy nhiên, cảm ơn bạn đã ghé qua.

Nếu Khách quan tâm, thì mình chỉ post bài xầm vào thứ 7 thôi nhé. Những ngày còn lại chỉ post bài nghiêm túc thôi. (bài này quá xầm nhưng được post vào thứ 5 nhé)

**Làm sao để thực hiện blog này** <ul> <li>Viết mark-down và latex hỗ trợ bởi wordpress</li> <li>Server cho phép lưu ảnh động <a href="https://giphy.com/">giphy</a></li> <li>Vấn đề lưu trữ ảnh: sử dụng tính năng live upload của github.com</li> </ul>

Bỏ cái này vì quá chậm. Không hỗ trợ tốt latex (công thức toán và reference). Mình vẫn thích một công cụ offline hơn.

**Chào từ hồi [magizbox.com](https://magizbox.com), cái này tồn tại được 488 ngày (1 năm 4 tháng. wow) (từ 01/07/2016 đến 01/11/2017)**



Hello World,  
 My name is Vu Anh. I'm a developer working at a startup in Hanoi, Vietnam. Coding and writing is fun, so I make this site to share my gists about computer science, data science, and more. It helps me keep my hobby and save information in case I forget. I wish it will be useful for you too.  
 PS: I always looking for collaboration. Feel free to contact me via email brother.rain.1024[at]gmail.com  
 Magizbox Stories  
 Oct 2, 2016: Wow. It's 524th day of my journey. I added some notes in README.md, index.html, changed structure of website. Today I feel like at begin day when I start writing at datayo.wordpress.com blog. In this pass, there are times when I want to make a professional website like tutorialpoints but it doesn't work that way. Because in my heart, I don't want it, I don't to make a professional website. I just love coding, writing and sharing my hobby with people around the world. So today I come back to starting point, I will keep my writing schedule, make some fun stuffs each week.  
 In July 2016, I turn to use HTML and mkdocs, and opensource magizbox.  
 In March 2015, I start writing blog with wordpress.

Bỏ cái này vì thời gian build quá lằng nhằng. Quản lý dependencies các kiểu rất lâu. Muốn có một cái gì đó giúp viết thật nhanh và đơn giản.

**Chào từ hồi datayo.wordpress.com, cái này tồn tại được 489 ngày (1 năm 4 tháng) (từ 01/03/2015 đến 01/07/2016)**

I'm a junior data scientist, working as a researcher in big data team at a company in Vietnam. I love listening music when I'm writing code because it's make me coding better. I love reading books before sleeping because it take me sleep easier and discover the world with data mining via beautiful language R.  
 I write this blog because I want to share my hobbies with everybody. I hope you will enjoy it. Feel free to contact me via twitter @rain1024oremailbrother.rain.1024@gmail.com(Iwillanswerallemailsforsure)foranythingyouwantto  
 In case you find one of my posts could be better, don't hesitate to drop me a line in comment. I'm very appreciated and I will try my best to make it better and better.

Bỏ cái này. Bỏ wordpress. Vì muốn một site interactive hơn.

# Phần I

## Lập trình

## Chương 2

# Giới thiệu

### 2.1 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)

#### 2.1.1 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

## 2.1.2 Data Structure

Number

String

Collection

DateTime

Boolean

Object

## 2.1.3 OOP

Classes Objects

Inheritance

Encapsulation

Abstraction

Polymorphism

For OOP Example: see Python: OOP

## 2.1.4 Networking

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

## 2.1.5 Sample Project Ideas

Guess My Number Game

Create Analog Clock

Create Pong Game

Create flappy bird

## 2.2 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

## 2.3 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

## 2.4 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:

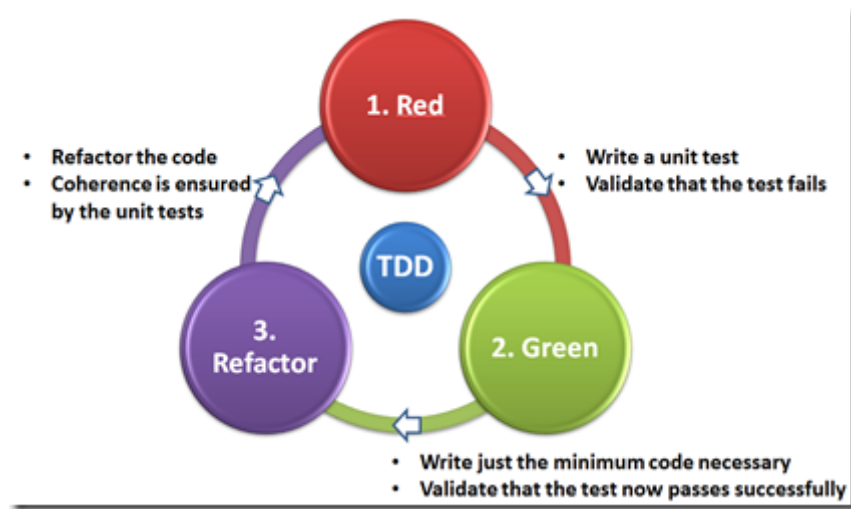
Treats programs as a model of a finite state machine or any other formal automata

Main Characteristics:

State enumeration, control variable, state changes, isomorphism, state transition table

Examples: AsmL

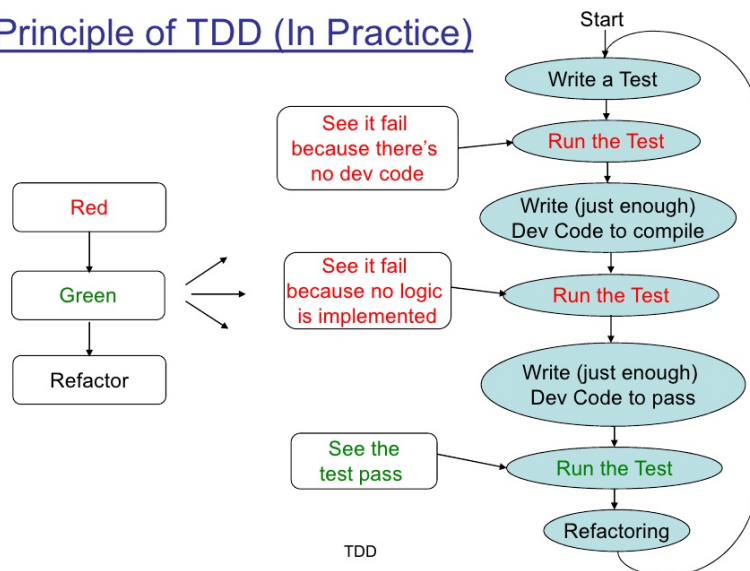
## 2.5 Testing



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)



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.

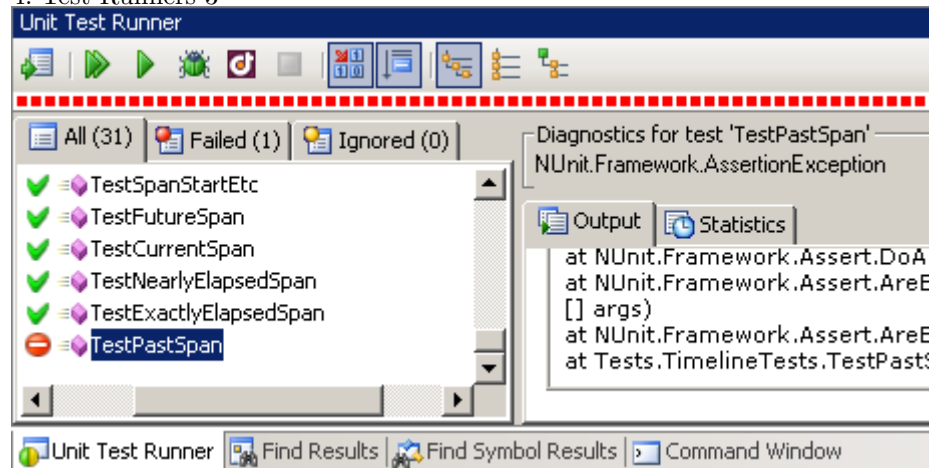
Eliminate duplication Red: (Automated test fail) Green (Automated test pass because dev code has been written) Refactor (Eliminate duplication, Clean the code)

## 3. Assertions Assert Framework

Numeric	Array	String	Exception
12, 34.5	[1, 2, 3] [4, 5, 6]	"hello" "world"	IOException TypeErrorException
areEqual	areEqual	areEqual	assertRaises
greaterThan	contains	startsWith	expected=Exception
lessThan	hasLength	endsWith	fail

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

## 4. Test Runners 3



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.

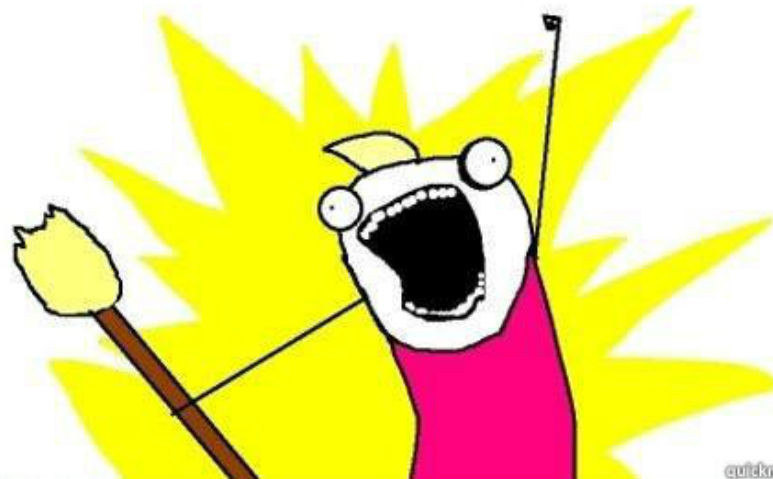
#### 5. Test Frameworks

Language Test Frameworks C++/VisualStudio C++: Test Web Service rest-assured Web UI SeleniumHQ

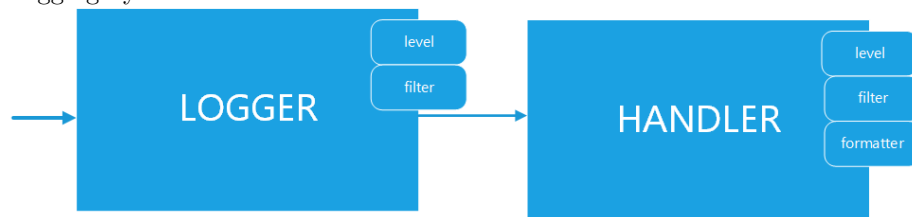
## 2.6 Logging

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

# LOG ALL THE THINGS



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.

**2.7 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.

## 2.8 Lập trình song song

Parallel/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
```

## 2.9 IDE

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. 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 3

# Python

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

### 3.1 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

## 3.2 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
```

### 3.3 Cơ bản

### 3.4 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<sub>v</sub>ar*. Next, the statements block is executed. Each time the block is executed, the next item in the sequence is assigned to the iterating variable.

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](http://www.tutorialspoint.com) "Python Loops". [www.tutorialspoint.com](http://www.tutorialspoint.com)  
 "What does the "yield" keyword do?". [stackoverflow.com](http://stackoverflow.com) "Improve Your Python:  
 'yield' and Generators Explained". [jeffknupp.com](http://jeffknupp.com)

**Vấn đề với mảng**

Random Sampling <sup>1</sup> - 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

### 3.5 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

<sup>1</sup> 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 prime 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<sub>p</sub>primes* implementation above fulfills the requirements, so we tell our boss we're done. She reports so.

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<sub>p</sub>primes* function on a very large list of numbers. In fact, the list is so large that merely creating it would consume

Once we think about this new requirement, it becomes clear that it requires more than a simple change to *get<sub>p</sub>primes*. Clearly, we can't return a list of all the prime numbers from start to infinity.

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?"

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:

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.

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.

This point bears repeating: to get the next value from a generator, we use the same built-in function as for iterators: `next()`.

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.

```
>> def simple_generator_function():>>> yield1>>> yield2>>> yield3Andherearetwosimplewaystous
>> for value in simple_generator_function():>>> print(value)123 >>>
our_generator = simple_generator_function()>>> next(our_generator)1>>>
next(our_generator)2>>> next(our_generator)3Magic?What'sthemagicpart?Gladyouasked!Whenagenera
```

Let's rewrite `getPrimes` as a generator function. Notice that we no longer need the magical infinite range function.

```
def getPrimes(number):
    while True:
        if isPrime(number):
            yield number
            number += 1
```

If a generator function calls `return` or reaches the end of its definition, a `StopIteration` exception is raised. This signals the end of the sequence of values. In our case, the loop is presenting `getPrimes`. If it weren't, the first time `next()` was called we would check if the number is prime and

```

>>> our_generator = simple_generator_function() >>> for value in our_generator: >>>
print(value)

```

```

>>> our_generator has been exhausted... >>> print(next(our_generator))
Traceback (most recent call last):
  File "<ipython - input - 13 - 7e48a609051a>", line 1, in <module>
    next(our_generator)
StopIteration

```

>> however, we can always create a new generator >> by calling the generator function again...

```

>>> new_generator = simple_generator_function() >>> print(next(new_generator))
perfectly valid! Thus, the flow of the generator is not affected by the
StopIteration exception.

```

Visualizing the flow Let's go back to the code that was calling `get_primes` :  
`solve_number_10`.

```

def solve_number_10():
    She *is* working on Project Euler 10, I know it! total = 2
    for next_prime in get_primes(3):
        if next_prime < 2000000:
            total += next_prime
        else:
            print(total)
            return
    It's helpful to visualize how the first few elements are created when we call
    get_primes in solve_number_10.

```

We enter the while loop on line 3. The if condition holds (3 is prime). We yield the value 3 and control to `solve_number_10`. Then, back in `solve_number_10` :  
 The value 3 is passed back to the for loop. The for loop assigns `next_prime` to this value. `next_prime` is added to `total`.  
`def get_primes(number):` while True : if is\_prime(number) : yield number  
 number += 1  
 <<<<<<<<<< Most importantly, number still has the same value it did when we called yield (i.e. 3). Remember, yield returns a value and then continues execution of the function.

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_primes` :

```

def print_successive_primes(iterations, base = 10):
    like normal functions, a generator function can be assigned to a variable.
    prime_generator = get_primes(base)
    missing code...

```

```

def get_primes(number):
    while True:
        if is_prime(number):
            ... what goes here?
    Then the next line of get_primes takes the value of number and yields it.
    yield foo means, "yield foo and, when a value is sent to me, set the next value to that value."
    You can "send" values to a generator.

```

```

def get_primes(number):
    while True:
        if is_prime(number):
            number = yield number
            number += 1
    In this way, we can set number to a different value each time the generator yields.
    We can use this to generate prime numbers.

```

```

def print_successive_primes(iterations, base = 10):
    prime_generator = get_primes(base)
    prime_generator.send(base)
    print(prime_generator.send(base ** power))
    Two things to note here: First, we're reprinting the result of generator.send().
    Second, notice the prime_generator.send(None) line. When you're using send to "start" a generator (that is,
    to get the first value), you should send None.

```

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

## 3.6 Cấu trúc dữ liệu

### 3.6.1 Number

Basic Operation

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

### 3.6.2 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 portions 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

### 3.6.3 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) : smartersearchfunctionThisfunctionsearchesforfirstoccurrenceofREpattern
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() : dogssubfunctionThismethodreplacesalldifferencesoftheREpatternins
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 [a]not, borc/analy[sz]e/analyseanalyzeanalyxe[a]



`g]characterbetweenag/[2-4]/demo1demo2demo3demo4demo5QuantifiersAlternation*`  
`a+a?0ormore,1ormore,0or1/go*gle/goglegoglegooglegooooooglehgle/go+gle/gglegoglegooglegooooooglehgle,`  
start / end of the string `/^abc/` `abc` `/^bc/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@` `fab`, linefeed, carriage return `//` `abc def` `/ab/` `ab` `//` `abc@00A9` unicode es-  
caped `@` `/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=def` `abc=def` `(?:abc)` non-capturing group  
`/(?:abc)3/` `abcabcabc` `abcabc` `(?=abc)` positive lookahead `/t(?:=s)/` `ttssstttss`  
`(?!abc)` negative lookahead `/t(?:!s)/` `ttssstttss` `(?<=abc)` positive lookbehind  
`/(?<=foo)bar/` `foobar` `fuubar` `(?<!abc)` negative lookbehind `/(?<!foo)bar/` `foo-`  
`bar` `fuubar` Related Readings

Online regex tester and debugger: PHP, PCRE, Python, Golang and JavaScript,  
[regex101.com](http://regex101.com) RegExr: Learn, Build, Test RegEx, [regexr.com](http://regexr.com)

### 3.6.4 Datetime

Print current time

```
from datetime import datetime
datetime.now().strftime(' %Y-%m-%d %H:%M:%S')
```

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", "%Y-%m-%d")
```

```
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-MM-DD")`

Create and format a moment with strftime semantics `moment.date(2012, 12, 18).strftime("YYYY-MM-DD")`

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.utcnw().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.utcnw().timezone("US/Eastern").date`

You can also clone a moment, so the original stays unaltered `now = moment.utcnw().timezone("US/Pacific") future = now.clone().add(weeks=2)` Related Readings How to get current time in Python, [stackoverflow](http://stackoverflow.com/questions/11110111/how-to-get-current-time-in-python) Does Python's `time.time()` return the local or UTC timestamp?, [stackoverflow](http://stackoverflow.com/questions/11110111/how-to-get-current-time-in-python) Measure time elapsed in Python?, [stackoverflow](http://stackoverflow.com/questions/11110111/how-to-get-current-time-in-python) momnet, <https://github.com/zachwill/moment>

### 3.6.5 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, you can use
        > args = 'a': 1, 'b': 2
        > s = Struct(**args)
        > s < main.Struct instance at 0x01D6A738 >
        > s.a1 > s.b2
Related Readings
stackoverflow, Convert Python dict to object?
```

## 3.7 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 = id
        self.balance = balance
    def get_balance(self):
        return self.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 `BankAccount` doesn't mean we've created one; we've merely outlined the blueprint to create a `BankAccount` object. To do so, we call the class's

*`__init__` method with the proper number of arguments (minus self, which we'll get to in a moment)*

So, to use the "blueprint" that we created 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 simply says "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.

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__` may make sense for other methods, but what about `__init__`? When we call `__init__`, we're in the process of creating an object, so how

This is why when we call `__init__`, we initialize objects by saying things like `self.id = id`. Remember, since `self` is the instance, this is equivalent to

Be careful what you `__init__`. After `__init__` has finished, the caller can rightly assume that the object is ready to use. That is, after `john = BankAccount`

**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:  $5,000 \times \text{number of wheels}$  a vehicle has. We love buying back

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 = wheels
        self.miles = miles
        self.make = make
        self.model = model
        self.year = year
        self.sold_on = sold_on

    def sale_price(self):
        if self.sold_on is not None:
            return 0.0
        already_sold_return = 5000.0 * self.wheels

    def purchase_price(self):
        if self.sold_on is None:
            return 0.0
        not_yet_sold_return = 8000 - (.10 * self.miles)
        OK, that looks pretty reasonable. Of course, we would likely have a number of other methods on the Car class, but we'll see why these are important in a bit.
```

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 = wheels
        self.miles = miles
        self.make = make
        self.model = model
        self.year = year
        self.sold_on = sold_on
```

```
def sale_price(self) : if self.sold_on is not None : return 0.0 Already sold return 5000.0 *
self.wheels
```

```
def purchase_price(self) : if self.sold_on is None : return 0.0 Not yet sold return 10000 -
(.10 * self.miles) Wow. That's almost identical to the car class. One of the most important rules of programming (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_sale_price = 0
def __init__(self, wheels, miles, make, model, year, sold_on): self.wheels = wheels self.miles = miles self.make = make self.model = model self.year = year
def sale_price(self) : if self.sold_on is not None : return 0.0 Already sold return 5000.0 *
self.wheels
def purchase_price(self) : if self.sold_on is None : return 0.0 Not yet sold return self.base_sale_price -
(.10 * self.miles) Now we can make the Car and Truck class inherit from the Vehicle class by replacing object in the
```

We can now define Car and Truck in a very straightforward way:

```
class Car(Vehicle):
def __init__(self, wheels, miles, make, model, year, sold_on): self.wheels = wheels self.miles = miles self.make = make self.model = model self.year = year
class Truck(Vehicle):
def __init__(self, wheels, miles, make, model, year, sold_on): self.wheels = wheels self.miles = miles self.make = make self.model = model self.year = year
class Struct: def __init__(self, **entries): self.__dict__.update(entries) Then, you can use
> args = 'a': 1, 'b': 2 > s = Struct(**args) > s < main.Struct instance at 0x01D6A738 > > s.a1 > s.b2 Suggested Readings Improve Y
```

### 3.7.1 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 __init__(self, i): self.i = int(i)
def add_one(self) : self.i += 1 ic = IntContainer(2) ic.add_one() print(ic.i) 3 This is a bit of a silly example, but
their ability to bundle data and operations into a single object, which leads to cleaner, more manageable, and more
oriented approach to programming can be very intuitive and powerful.
```

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 add method (which is the machinery underneath the + operator):
```

(Note: We'll use 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 __add__(self, other): print "specializing addition" return super(MyInt, self).__add__(other)
```

i = MyInt(2) print(i + 2) specializing addition 4 Using the + operator on our derived type goes through our

add method, as expected. We see that int really is an object that can be subclassed and extended just like user-defined

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) main.DoNothing We see that our variable is an instance of the class main.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_factory(s) : i = int(s) return i
i = int_factory('100') print(i) 100 This is overly-simplistic, but any function you write in the course of a normal program takes some arguments, does some operations, and creates and returns an object. With the above discussion in mind, though, this is a metafunction :
```

```
def class_factory() : class Foo(object) : pass return Foo
```

```
F = class_factory() f = F() print(type(f)) <class 'main.Foo'> Just as the function int_factory constructs and returns an instance of int, the function class_factory constructs and returns an instance of class.
```

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_factory() : return type('Foo', (), )
```

```
F = class_factory() f = F() print(type(f)) <class 'main.Foo'> In fact, the construct
```

```
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): return self.i
b = Bar() print(b.get_i()) 4
Foo = type('Foo', (), dict(i = 4))
Bar = type('Bar', (Foo,), dict(get_i = lambda self: self.i))
b = Bar() print(b.get_i()) 4
```

*This perhaps seems a bit over-complicated in the case of this contrived example, but 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):
        create a class if it's not specified if 'class_id' not in dct:
            dct['class_id'] = name
        open the specified file for writing if 'file' in dct:
            filename = dct['file']
            dct['file'] = open(filename, 'w')
        we need to call type.new to complete the initialization
        return super(InterfaceMeta, cls).new(cls, name, parents, dct)
```

*Notice that we've modified the new method of the metaclass.*

Now we'll use our InterfaceMeta class to construct and instantiate an Interface object:

```
Interface = InterfaceMeta('Interface', (), dict(file='tmp.txt'))
print(Interface.class_id) print(Interface.file)
interface < open file 'tmp.txt', mode 'w' at 0x21b8810 >
```

*This behaves as we'd expect: the class\_id class variable is created, and the file class variable is replaced with an open file object.*

```
class Interface(object):
    metaclass = InterfaceMeta
    file = 'tmp.txt'
print(Interface.class_id) print(Interface.file)
interface < open file 'tmp.txt', mode 'w' at 0x21b8ae0 >
```

*By defining the metaclass attribute of the class, we've told the class that it should be constructed using InterfaceMeta rather than using type. To make this work, we've overridden the new method of the metaclass. Furthermore, any class derived from Interface will now be constructed using the same metaclass.*

```
class UserInterface(Interface):
    file = 'foo.txt'
print(UserInterface.file) print(UserInterface.class_id)
< open file 'foo.txt', mode 'w' at 0x21b8c00 >
```

*User interface. This simple example shows how metaclasses can be used to create powerful and flexible APIs for programming.*

**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):
    def __new__(cls, name, bases, dct):
        we use __init__ rather than __new__ here because we want to modify attributes of the class after they have been created
        super(DBInterfaceMeta, cls).__init__(name, bases, dct)
        Our metaclass simply adds a registry dictionary if it's not already present, and adds the new class to it
        class DBInterface(object):
            metaclass = DBInterfaceMeta
            registry = {}
        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 'main.FirstInterface'>, 'secondinterface': <class 'main.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  
– Tim Peters

In a way, this is a very unsatisfying answer: it's a bit reminiscent of the wistful and clichéd 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.

### 3.7.2 Design Patterns

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

```

@Singleton class Foo: def __init__(self): print 'Foocreated'
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__’ method that takes only the ‘self’ argument. Also, the decorated class cannot be inherited from.
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) Threadsafe fewer idempotent implementation of singletons. A thread-safe implementation
```

```

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:
            with cls._singleton_lock:
                if not cls._singleton_instance:
                    cls._singleton_instance = cls()
        return cls._singleton_instance

class A(SingletonMixin):
    pass
class B(SingletonMixin):
    pass
if __name__ == '__main__':
    a1=A.instance(),a2=A.instance(),b=B.instance(),b2=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?')

```

### 3.8 File System IO

JSON Write json file with pretty format and unicode

```

import json
import io
data = {
    "menu": {
        "header": "Sample Menu",
        "items": [
            {
                "id": "Open",
                "label": "Open New",
                "value": None
            },
            {
                "id": "Help",
                "label": "About",
                "value": "About Adobe CVG Viewer..."
            }
        ]
    }
}
with io.open("sample.json", "w", encoding="utf8") as f:
    content = json.dumps(data, indent=4, sort_keys=True, ensure_ascii=False)
    f.write(unicode(content))
Result
{
    "menu": {
        "header": "Sample Menu",
        "items": [
            {
                "id": "Open",
                "label": "Open New",
                "value": null
            },
            {
                "id": "Help",
                "label": "About",
                "value": "About Adobe CVG Viewer..."
            }
        ]
    }
}
Read json file
import json
from pprint import pprint
with open('sample.json') as data_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 = ET.Element('catalog')
insert comment
comment = ET.Comment('this is a xml sample file')
root.insert(1, comment)
book = ET.SubElement(root, 'book', id="bk001")
book_data = ET.SubElement(book, 'author')
book_data.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 "Book Information", book.attrib["id"]
print "Author : ", book.find('author').text
print "Title : ", book.find('title').text
Result
Book Information ID : bk001
Author : Gambardella, Matthew
Title : XML Developer's Guide

```



### 3.9 Operating System

File Operations Copy folder 1

```
import shutil
shutil.copyfile("src", "dst")
```

CLI shutil — High-level file operations

### 3.10 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)
```

### 3.11 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_new_thread(function, args[, kwargs])
```

This method call enables a fast and efficient way to create new threads.

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_time(threadName, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print "
```

```
Create two threads as follows try:
thread.start_new_thread(print_time, ("Thread-1", 2,))
thread.start_new_thread(print_time, ("Thread-2", 4,))
except:
    print "Error: unable to start thread"
```

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 init(self,threadID,name,counter):threading.Thread.init(self)self.threadID=threadID
def print_time(threadName,delay,counter) : whilecounter : ifexitFlag :
threadName.exit()time.sleep(delay)print"counter- = 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 init(self,threadID,name,counter):threading.Thread.init(self)self.threadID=threadID
def print_time(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 process-
ing One Thread-2 processing Two Thread-3 processing Three Thread-1 process-
ing Four Thread-2 processing Five Exiting Thread-3 Exiting Thread-1 Exit-
ing 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.
```

### 3.12 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 Mes-  
saging 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 hun-  
guyen/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)publisher.publishesaneventpublisher.publish(event)c
Subscriber('S1')subscriber.subscribe(topic)subscriber.getallneweventsbytimestamp
subscriber.get_events(pydispatcher
```

stackoverflow, Recommended Python publish/subscribe/dispatch module?

### 3.13 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_folder/your_project_name/your_project_name/static/models.pyserializers.pysettings.pyurls.pyviews.py
InstalldependenciespipinstalldjangoipinstalldjangoestframeworkpipinstallmarkdownMarkdownsupport
filterFilteringsupportpipinstalldjango - cors - headersCORSSupportStep2 :
```

Createprojectdjango-adminstartprojectyour\_project\_nameStep3 : Configapps3Add'your\_project\_name','res

```
INSTALLED_APPS = (...'your_project_name''rest_framework',)Step4 : Model,View,Route6Step4.1 :
```

CreateamodelandserializerYoucangotoDjango : Modelfieldreferencepageformorefields.

Step 4.1.1: Create Task class in your\_project\_name/models.pyfilefromdjango.dbimportmodels

```
class Task(models.Model): content = models.CharField(max_length = 30)status =
```

```
models.CharField(max_length = 30)Step4.1.2 : CreateTaskSerializerclassinyour_project_name/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_project_namepythonmanage.py migrateStep4.2 :
```

CreateTaskViewSetclassinyour\_project\_name/views.pyfilefromyour\_project\_name.modelsimportTaskfrom

```
class TaskViewSet(viewsets.ModelViewSet): queryset = Task.objects.all() serializer_class =
```

TaskSerializerStep4.3 : Configroute5Changeyour\_project\_name/urls.pyfile

```
from django.conf.urls import include, url from django.contrib import admin
```

```
from rest_frameworkimportroutersfromyour_project_name.viewsimportTaskViewSet
```

```
router = routers.DefaultRouter() router.register(r'api/tasks', TaskViewSet)
```

```
admin.autodiscover()
```

```
urlpatterns = [ url(r'^admin/', include(admin.site.urls)), url(r'^', include(router.urls)), url(r'^api-
```

```
auth/', include('rest_framework.urls', namespace = 'rest_framework'))]Step5 :
```

```

RunServerpythonmanage.pyrunserverStep6.UseAPIStep6.1: Createanewtaskcurl-
i - XPOST - H"Content - Type : application/json"http : //localhost :
8000/api/tasks-d"content" : "a", "status" : "INIT"'Step6.2: Listalltaskscurlhttp :
//localhost : 8000/api/tasksStep6.3: Getdetailoftask1curlhttp : //localhost :
8000/api/tasks/1Step6.4: Deletetask1curl-i - XDELETEhttp : //localhost :
8000/api/tasks/1Step7: CORSKnownError : No' Access - Control - Allow -
Origin'headerisresentontherequestedresource.Origin'null'isthereforenotallowedaccess.
Step 7.1: Install corsheader app Add module corsheaders to yourproject_name/settings.py
INSTALLED_APPS = (... 'corsheaders', ...)Step7.2AddmiddlewareclassesAddmiddleware_classestoyourp
MIDDLEWARE_CLASSES = (... 'corsheaders.middleware.CorsMiddleware', 'django.middleware.com
AllowAll
Add this line to yourproject_name/settings.py
CORS_ORIGIN_ALLOW_ALL : TrueStep8: httpsYoucanusehttps : //github.com/teddziuba/django-
sslserver
Unicode REST_FRAMEWORK = 'DEFAULT_RENDERER_CLASSES' : ('rest_framework.renderers.
PagingAddthismodulesettingtoyourproject_name/settings.py
REST_FRAMEWORK = 'DEFAULT_PAGINATION_CLASS' : 'rest_framework.pagination.LimitOf
API:
GET <>/?limit=<limit>offset=<offset>
Step 10: Search by field in import this to your viewsets.py
from rest_frameworkimportfilters
add this to your viewsets class
filter_backends = (filters.SearchFilter,)search_fields = ('< 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_delete = models.CASCADE)StartingwithMysqlAddthisdatabasesettingstoyourproject_name/settings.p
DATABASES = 'default': 'ENGINE': 'django.db.backends.mysql', 'NAME':
'[DB_NAME]', 'USER': '[DB_USER]', 'PASSWORD': '[PASSWORD]', 'HOST':
'[HOST]', 'OranIP Address that your DB is hosted on' '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_frameworkimportmixins
class CreateModelMixin(object): """ Create a model instance. """ def cre-
ate(self, request, *args, **kwargs): event = request.data try: event['time'] =
int(time.time()) except Exception, e: print 'Set Time Error' serializer = self.get_serializer(data =
request.data)serializer.is_valid(raise_exception = True)self.perform_create(serializer)headers =
self.get_success_headers(serializer.data)returnResponse(serializer.data, status =
status.HTTP_201_CREATED, headers = headers)
def perform_create(self, serializer): serializer.save()
def get_success_headers(self, data): try : return'Location' : data[api_settings.URL_FIELD_NAME]except
return
class YourViewSet(CreateModelMixin, mixins.RetrieveModelMixin, mixins.UpdateModelMixin,
mixins.DestroyModelMixin, mixins.ListModelMixin, GenericViewSet): queryset
= YourModel.objects.all() serializer_class = YourModelSerializerLoggingsettingsHereisanexample, putthi

```

```
LOGGING = 'version': 1, 'disable_existing_loggers': False, 'formatters' :
'verbose' : 'format' :', 'simple' : 'format' :',, 'filters' : 'special' : '()' : 'project.logging.SpecialFilter', 'foo' :
'console' : 'level' : 'INFO', 'filters' : ['require_debug_true'], 'class' : 'logging.StreamHandler', 'formatter' :
'django' : 'handlers' : ['console'], 'propagate' : True,, 'django.request' : 'handlers' : ['mail_admins'], '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\_clientpackageStep5 : 2mkdircondacdcondagitclonehttps : //github.com/hunguyen1702/condacr.gitREADME.mdStep6 : Editmeta.yamlfileinyour\_packagefolder6.1Followinstructioninsidemeta.yamlStep7 : build : -python -setuptoolsrun : -pythonwith : requirements : build : -python -setuptools -six -certifi -python -dateutilrun : -python -six -certifi -python -dateutilStep7 : cd..condabuildyour\_packageStep8 : mkdirchannelcdchannelcondaconvert--platformall /anaconda/conda-bld/linux-64/your\_package0.1.0-py270.tar.bz2Step9 : Createvirtual-envname : your\_envnamedependencies : -certifi = 2016.2.28 = py270 - openssl = 1.0.2h = 0 - pip = 8.1.2 = py270 - python = 2.7.11 = 0 - python - dateutil = 2.5.3 = py270 - readline = 6.2 = 2 - setuptools = 20.7.0 = py270 - six = 1.10.0 = py270 - tk = 8.5.18 = 0 - wheel = 0.29.0 = py270 - zlib = 1.2.8 = 0 - pip : -urllib3 == 1.15.1Step10 : Install : condainstall - -use -localyour\_packageDjango

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\_framework : mixins

### 3.14 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_root]level = DEBUGhandlers = consoleHandler, fileHandler
[handler_consoleHandler]class = StreamHandlerlevel = DEBUGformatter =
formatterargs = (sys.stdout,)
[handler_fileHandler]class = FileHandlerlevel = DEBUGformatter =
formatterargs = ('logs/app.log','a')
[formatter_formatter]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 mes-
sage') 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: Log-
ging module
Python: Logging cookbook
Python: Logging guide
```

### 3.15 Configuration

```
pyconfiguration
Installation conda install -c rain1024 pyconfiguration Usage Step 1: Create
config.json file
"SERVICE_URL" : "http://api.service.com" Step2: Addthesecodetomain.py file
from pyconfiguration import Configuration Configuration.load('config.json')
print Configuration.SERVICE_URL
> http://api.service.com References: What's the best practice using a set-
tings file 1
What's the best practice using a settings file in Python?
```

### 3.16 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 =
"write report to FILE", metavar = "FILE") parser.add_option("-q", "--quiet", action =
"store_false", dest = "verbose", default = True, help = "don't print status messages to stdout")
(options, args) = parser.parse_args() optparse supports (among other things) :
```

Multiple options in any order. Short and long options. Default values. Generation of a usage help message. Suggest Reading Command Line Arguments In Python

### 3.17 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/PHPUnit/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't happen")
```

```
@unittest.skipIf(mylib.__version__ < (1,3), "not supported in this library version") def test_format(self): Test that work for only a certain version
```

```
@unittest.skipUnless(sys.platform.startswith("win"), "requires Windows") def
```

```
test_windows_support(self) : windowsspecific testing code pass Tox to aim to automate and standardize testing
```

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:\27python = 2.7 conda create -p C:\34python = 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

### 3.18 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-Indentation and code formatting

Automatic indents are inserted on new line. Indent verification and code re-formatting 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 *something* Code folding

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: Ctrl+Shift+V (paste)

```
set VS100COMNTOOLS=
```

## 3.20 Environment

Environment Management 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, Mini-conda doesn't come with any installed packages by default. Note that for mini-conda, 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
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
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
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
```

Remove

If you didnt really want an environment named flowers, just

→ remove it as follows:

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

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 developers environment from their

→ environment.yml file:

```
conda env create -f environment.yml
```

```
# remove environment
```

```
conda remove -n <your_enviroment> --all
```

## 3.21 Module

Create Public Module conda, pypi, github

Step 0/4: Check your package name Go to [https://pypi.python.org/pypi/your\\_package\\_name](https://pypi.python.org/pypi/your_package_name) to see your package

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\_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] :

It will create a directory

| - 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.py Step 2/4

2. Create a .pypirc configuration file in *HOME* directory

[distutils] index-servers = pypi

[pypi] repository=https://pypi.python.org/pypi username=your\_username password =

your\_password 3. Change your MANIFEST.in

recursive-include project\_folder \* 4. Upload your package to PyPI

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_package_folder mkdir conda_skeleton
cd conda_skeleton
pypi your_package This creates a directory named your_package
|- your_package | - bld.bat | - meta.yaml | - build.sh
3. Build your package
conda build your_package
convert to all platform conda convert -f -platform all C:-bld-64_package -
0.1.1 - py27_0.tar.bz2
Upload package to Anaconda
anaconda login anaconda upload linux-32/your_package.tar.bz2
anaconda upload linux-64/your_package.tar.bz2
anaconda upload win-32/your_package.tar.bz2
anaconda upload win-64/your_package.tar.bz2
Create Private Module Step 1: Make your module
1.1 pip install cookiecutter
1.2 cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git
1.3 Fill all necessary information
full_name[Audrey Roy Greenfield] : 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] :
Step 2: Build your module
Change your MANIFEST.in
recursive-include project_folder * Build your module with setup.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/dist
Step 3: Usage your module in the same machine
import your_project_name
In another machine
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 cookiecutter on terminal:
mkdir build cd build pip install cookiecutter
cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git
Fill all necessary information
full_name[Audrey Roy Greenfield] : 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] :
It will create a directory
|- 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.py
Copy your package to a new channel
Add this line to MANIFEST.in
recursive-include project_folder *
Step 2: Build conda package
mkdir conda_channel
cd conda_channel
git clone https://github.com/hunguyen1702/condaBuildLocalTemplate.git
mv condaBuildLocalTemplate/your_package_name.rf.git README.md
Edit the file meta.yaml with the instruction inside it
cd ..
conda build your_package_name
Step 3: Create custom channel and install from local package
Create a channel directory
cd channel
Convert your_package you've built to all platform
conda convert -platform all /anaconda/conda-bld/linux-64/your_package_0.1.0-py27_0.tar.bz2
and this will create :
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
Verify your new channel

```

```
conda search -c file://path/to/channel/ --override-channels
```

If you see your *package's appearance*, so it's work  
 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/
```

## 3.22 Production

Production with docker Base Image: magizbox/conda2.7/  
 Docker Folder  
 your\_app/appconfig/main.py Dockerfile run.sh Dockerfile  
 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

## 3.23 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 dòng 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://cdn.jsdelivr.net/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/541#issuecomment-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

```

## 3.24 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.

## 3.25 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. Vải thật. Làm thẳng đê không biết đầu 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 thẳng readthedocs có phần log cho từng build một. Lăn 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)  


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 bởi 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 tạo (thực sự là rất nhiều tạo). Thấy cái này trong trang Settings



Ồ há. Thằng đàn này cho phép trở 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



Build thử. Build thử thôi. Cảm giác đúng lắm rồi đây. Và... nó chạy. Ahihi



### 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 thêm build thẳng PDF với Epub nữa. Tiết kiệm được bao nhiêu thời gian.

## 3.26 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 đổ lờm khó chịu thật.

## 3.27 Vì sao lại code python?

01/11/2017 Thích python vì nó quá đơn giản (và quá đẹp).

[<sup>1</sup>] : <https://github.com/jupyter/nbconvert/issues/91#issuecomment-283736634>

## Chương 4

# 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/>

### 4.1 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_64.exe` *from the website* <https://cygwin.com/install.html>

Install `setup-x86_64.exe` file

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

## 4.2 Basic Syntax

C/C++ Hello World include <iostream> using namespace std;

```
int main() cout << "hello world";
Convention Naming variable n ame i k e t h i s c l a s s a t a m e m e b e r n ame i k e t h i s
//GargantuanTableIterator * iter = table-> NewIterator(); //for(iter->
Seek("foo"); !iter-> done(); iter-> Next())//process(iter-> key(), iter-> value()); ///delete iter; cl
Used" *" here for concatenation operator. //TODO(Zeke) change this to user relations.
```

## 4.3 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 *Several of the basic types can be modified using the 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; return 0; 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; //example int*ip; //pointertoaninteger double*dp; //pointertoa double float*
fp; //pointertoa float char*ch; //pointertocharacter PointerLab
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`, and `tm`. The types `clock_t`, `size_t` and `time_t` are capable of

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; //seconds of minutes from 0 to 61 int tm_min; //minutes of hour from 0 to 59 int tm_hour; //
```

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_t now =
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

## 4.4 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;
```

## 4.5 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(int argc, _TCHAR* argv[]) {
    int rc; char *error;
    // Open Database
    cout << "Opening MyDb.db ..." << endl;
    sqlite3 *db; rc = sqlite3_open("MyDb.db", &db);
    if(rc) cerr << "Error opening SQLite3 database : " << sqlite3_errmsg(db) << endl;
    // 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 << "Error executing SQL statement : " << error << endl;
    // 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 << "Error executing SQL statement : " << error << endl;
    // 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 << "Error executing SQL statement : " << error << endl;
    // Display Cell Value
    cout << "Cell Value: ";
    cout << results[cellPosition];
    cout << endl;
    // End Line
    cout << endl;
    // Display Separator For Header
    if(0 == rowCtr) {
        for(int colCtr = 0; colCtr < columns; ++colCtr) {
            cout << "Column " << colCtr << ": ";
            cout << results[cellPosition];
            cout << " ";
        }
        cout << endl;
        sqlite3_free_table(results);
    }
    // Close Database
    cout << "Closing MyDb.db ..." << endl;
    rc = sqlite3_close(db);
    if(rc) cerr << "Error closing SQLite3 database : " << error << endl;
    // Wait For User To Close Program
    cout << "Please press any key to exit the program ..." << endl;
    cin.get();
    return 0;
}

```

## 4.6 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 Security Development Lifecycle(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_METHOD function`  
 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.

## 4.7 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 5

# 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.

### 5.1 Installation

Google Chrome Pycharm

### 5.2 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.

### 5.3 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 Enum-
NamesLikeThis methodNamesLikeThis CONSTANT_VALUES_LIKE_THIS foo.namespaceNamesLikeThis.
```

```
3.1 Class Comment /** * Class making something fun and easy. * @param
string arg1 An argument that makes this more interesting. * @param Ar-
ray.<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 My-
Class and returns something. * @param project.MyClass obj Instance of My-
Class which leads to a long * comment that needs to be wrapped to two lines. *
@return boolean Whether something occurred. */ function PR_someMethod(obj) // ... 4. Expression and Statement
```

```
22 "this is an expression" (5 > 6) ? false : true Statements The Simplest kind
of statement 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 access 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 too 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-frame will be destroyed, and all the local variable
such as text will be undefined. 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 "aaaaaaaaa!"; 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)

```
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

## 5.4 Data Structure

### 5.4.1 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

```

### 5.4.2 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 char-
acters 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: Any 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 = /\w/; console.log(word.test("@@")); Any whitespace character
(space, tab, newline, and similar)

```

```

var space = /\s/; 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 = /\D/; console.log(notDigit.test("ww")); console.log(notDigit.test("1a"));
console.log(notDigit.test("1124")); A nonalphanumeric character

```

```

var nonAlphanumericChar = /\W/; console.log(nonAlphanumericChar.test("abc12231"));
console.log(nonAlphanumericChar.test("!@§A nonwhitespace character

```

```

var nonWhiteSpace = /\S/; console.log(nonWhiteSpace.test("abc123")); con-
sole.log(nonWhiteSpace.test("1. abcd")); console.log(nonWhiteSpace.test(" "));

```

Any character except for newline

```

var anything = /.../; console.log(anything.test("abc. ")); console.log(anything.test("acbacd. "));
console.log(anything.test("acba ")); "Using caret character to match any except the ones

```

```

var notBinary = /[01]/; console.log(notBinary.test("01101011100")); console.log(notBinary.test("010210
" Match one or more" * " Match zero or more

```

```

console.log(/+/.test(1234)); console.log(/+/.test());

```

```

console.log(/*/.test(1234)); console.log(/*/.test()) "?" Question mark test a
character exist or not is still ok

```

```

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("wazup"));
// Grouping Subexpressions () using parentheses to make whole group like one character
var cartoonCrying = /boo+(hoo+)+/i; //i to match all Capitalize or normal text
console.log(cartoonCrying.test("Boohooooohooohoo")); console.log(cartoonCrying.test("boohooooohooOO"));
// Matches and group Test is a simplest method, and it only return true or false.
// exec (execute) is another 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("she said 'hello'")); and if the subexpression appear on some more times, then
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 pattern Only one in the list between 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 expression
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+3 console.log(stripComments("x = 10; //ten!")); //x = 10; console.log(stripComments("1/*a*/+/*b*/1")); //11 Search method Search method return the first index if the regular expression match. And return -1 if not found
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("xyzyzy"); //lastIndex update to 5 console.log(pattern.lastIndex);
match = pattern.exec("xyzyzyxxx"); //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);

```

### 5.4.3 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);

### 5.4.4 Datetime

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

### 5.4.5 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);
```

### 5.4.6 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);
```

## 5.5 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 rabbit 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 proto-
    type 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 = func-
    tion(line) console.log("The " + this.type + " rabbit 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"); ; killerRab-
    bit.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 = Ob-
    ject.create(null); map["pizza"] = 0.09; console.log("toString" in map); console.log("pizza"
    in map);
    // POLYMORPHISM // laying out a table: example for polymorphism func-
    tion 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
    function drawTable(rows) var heights = rowHeights(rows); var widths =
    colWidths(rows);
    function drawLine(blocks, lineNo) return blocks.map(function(block) re-
    turn block[lineNo]; ).join(" ");
    function drawRow(row, rowNum) var blocks = row.map(function(cell, col-
    Num) return cell.draw(widths[colNum], heights[rowNum]); ); return blocks[0].map(function(i, lineNo) return
    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.length); , 0); ; TextCell.prototype.minHeight = function() return this.text.length;
TextCell.prototype.minHeight = function() return this.text.length; 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) % 2 == 0) 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 attempt 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 con-
structor?

```

## 5.6 Networking

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

## 5.7 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
```

## 5.8 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.

```

## 5.9 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 = Per-
son("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); con-
sole.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 = func-
tion(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: negative 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 promptDirection(question) var result = prompt(question, ""); if (re-
sult.toLowerCase() == "left") return "L"; if (result.toLowerCase() == "right")
return "R"; throw new Error("Invalid direction: " + result);
function look() if (promptDirection("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 oldCon-
text = 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 = promptDirection("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 Er-
ror()).stack; InputError.prototype = Object.create(Error.prototype); InputEr-
ror.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];
```

## 5.10 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(); ) ); );
```

## 5.11 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_components/jquery/dist/jquery.js" >< /script ><
scriptsrc = "./bower_components/moment/moment.js" >< /script >< scriptsrc =
"./bower_components/underscore/underscore.js" >< /script >< scriptsrc =
"./bower_components/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>
```

## 5.12 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_parties = ["bower_components/jquery/dist/jquery.js", "bower_components/bootstrap/dist/js/b
var modules = [ "modules/your_script.js"];
gulp.watch(third_parties, ['js_thirdparty']); gulp.watch(modules, ['js_modules']);
gulp.task('js_thirdparty', function() return gulp.src(third_parties).pipe(concat('third_party.uglify.js')).pi
gulp.task('js_modules', function() return gulp.src(modules).pipe(concat('modules.uglify.js'))/.pipe(ugl
gulp.task('documentation', function () return gulp.src("./modules/*/*.js")
.pipe(jsdoc('./documentation')); );
gulp.task('default', ['js_thirdparty', 'js_modules']); http : //gulpjs.com/
Deprecated grunt
```

## 5.13 Make Module

Make Module sample modules: underscore, momentjs

Folder Structure |- docs |- test |- src | |- your\_module.js| - .gitignore| -  
 bower.json

# Chương 6

## 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.

### 6.1 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 Runinstalljavasudoupdate-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
```

### 6.2 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 its 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- -> 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 (*bitwise XOR*) Binary XOR Operator copies the bit if it exists in either operand.

Example: (A ^ B) will give 49 which is 0011 0001 4 (*bitwise complement*) Binary Ones Complement Operator is used to perform the bitwise complement.

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

Example C /= A is equivalent to C = C / A 6

Example: C

Example C <<= 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 = bitwise exclusive OR and assignment operator.

Example: C = 2 is same as C = C ^ 2 11 = bitwise inclusive OR and assignment operator.

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 to right  
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
```

## 6.3 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<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?

## 6.4 OOP

### 6.4.1 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 age;
    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;installation/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
```

### 6.4.2 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 En-
capTest class. Normally, these methods are referred as getters and setters. There-
fore, 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](http://www.tutorialspoint.com). N.p., 2016. Web. 10 Dec. 2016.

### 6.4.3 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.

#### 6.4.4 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 sub-type 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 = num-
ber;
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) su-
per(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 Mo-
htashim", "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 Em-
ployee 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 refer-
ence 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](http://www.tutorialspoint.com). N.p., 2016. Web. 10 Dec. 2016.

### 6.4.5 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 = num-
ber;
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 ab-
stract 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 follow-
ing error
```

Employee.java:46: Employee is abstract; cannot be instantiated Employee e = new Employee("George W.", "Houston, TX", 43); <sup>1</sup>*errorInheritingtheAbstractClassWecaninheritthepro*

```
/* File name : Salary.java */ public class Salary extends Employee private
double salary; // Annual salary
public Salary(String name, String address, int number, double salary) su-
per(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.

The `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 semicolon (;) 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](http://www.tutorialspoint.com). N.p., 2016. Web. 10 Dec. 2016.

## 6.5 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

```
javac CopyFile.java
```

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

```
javac CopyFile.java
```

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'

```
javac ReadConsole.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 `makedirs( )` 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.
```

## 6.6 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 FileNotFoundExceptionDemo
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: javacFileNotFoundExceptionDemo.javaFileNotFoundExceptionDemo.java : 8 : error :
unreported exception FileNotFoundException; must be caught or declared to be thrown
FileReader fr = new FileReader(file);
1 errorNoteSince the methods read() and close() of FileReader class throws IOException
```

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 `ArrayIndexOutOfBoundsException` occurs.

```
public class UncheckedDemo
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.UncheckedDemo.main(UncheckedDemo.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 defini-
tion 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 Insuf-
ficientFundsException
```

```
import java.io.*; public class className
```

```
public void withdraw(double amount) throws RemoteException, Insufficient-
FundsException // 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 pro-
duce the following result
```

```
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3 First el-
ement 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 ReadDataDemo
```

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

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 Auto-Closeable 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 pre-defined 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 CheckingAc-
count.
```

```
// File Name BankDemo.java public class BankDemo
public static void main(String [] args) CheckingAccount c = new CheckingAc-
count(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(); Compile all the above three files and run Bank Demo
```

Output

Depositing 500...

Withdrawing 100...

Withdrawing 600...*Sorry, but you are short 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 categories 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\\_exceptions.htm](https://www.tutorialspoint.com/java/java_exceptions.htm).

## 6.7 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.properties

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.

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=St

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/).  
<http://www.tutorialspoint.com/log4j/>. "Java Logging". 2016. [tutorials.jenkov.com](http://tutorials.jenkov.com/java-logging/index.html).  
<http://tutorials.jenkov.com/java-logging/index.html>.

## 6.8 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

## 6.9 Package Manager

Java: Package Manager Gradle

Create your first project with gradle Step 1: Create new project folder

`mkdir gradle_sample` Step 2: *Make folder structure*

`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"`

## 6.10 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™
```

## 6.11 Production

Java: Production (Docker) Production with java

Base Image: `[java]/java`

Docker Folder

`your-app/ app bin your_app.sh lib Docker file run.sh Docker file`

FROM `java:7`

COPY `run.sh run.sh run.sh`

`cd /app/bin chmod u+x your_app.sh ./your_app.sh` Compose

service: build: `./your_appcommand : ' bash run.sh'`



## Chương 7

# PHP

PHP là ngôn ngữ lập trình web dominate tất cả các anh tài khác mà (chắc là) chỉ dụi 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 = "DELETE FROM 'groups' WHERE id = '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 trở 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 `Start debugger`, `Stop debugger`; 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 `Start Listening for PHP Debug Connections`; \* Đặt breakpoint tại dòng thứ 5 \* Refresh lại trang <http://localhost/untitled/add.php>, lúc này, breakpoint sẽ dừng ở dòng 5

## Chương 8

# R

View online <http://magizbox.com/training/r/site/>

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.

### 8.1 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

## 8.2 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.html>performance-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 9

# 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.

### 9.1 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

### 9.2 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/>

### 9.3 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 too 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_SOURCE).iterator.flatMap(entry =>
flatten(ResourcesPlugin.getWorkspace.getRoot.findMember(entry.getPath))) Refactor There's a lot of val
jp.getRawClasspath.filter( .getEntryKind == IClasspathEntry.CPE_SOURCE).defworkspaceRoot =
ResourcesPlugin.getWorkspace.getRoot def filesOfEntry(entry : Set[File]) =
flatten(workspaceRoot.findMember(entry.getPath).sources.iterator.flatMap(filesOfEntryPreferFunction))

```

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 diabolize 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 10

# 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.

### 10.1 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

## 10.2 Basic Syntax

```
Print console.log("Hello World"); Conditional if(you_smart)console.log("learnnodejs");elseconsole.log("goau
0; count < 10; count++)console.log(count);Functionfunctionprint_info(arg1,arg2)console.log(arg1);conso
```

## 10.3 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.jsVerify the Output.

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
```

*nodemain.jsVerifytheOutput.*

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 a socket. 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
nodemain.jsVerifytheOutput.
```

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
```

*nodemain.jsVerifytheOutput.*

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
```

*nodemain.jsVerifytheOutput.*

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
```

*nodemain.jsVerifytheOutput.*

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 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 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
```

*nodemain.jsVerifytheOutput.*

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 hspferdata<sub>a</sub>pachetesttest.txtRem

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 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 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 hsuperfdata_a pachetest.txt
```

## 10.4 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.npmjs.org](http://search.npmjs.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 --version 2.7.1 If you are running an old version of NPM then it is quite easy to update it to the latest version
sudo npm install npm -g /usr/bin/npm -> /usr/lib/node_modules/npm/bin/npm -
cli.js npm@2.7.1 /usr/lib/node_modules/npm Installing Modules There is a simple syntax to install any Node.js module
npm install < ModuleName > For example, following is the command to install a famous Node.js web framework
npm install express Now you can use this module in your js files as following
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_modules directory lying in the folder where Node application is present. Locally deployed
ls -lt total 0 drwxr-xr-x 3 root root 20 Mar 17 02:23 node_modules Alternatively, you can use npm ls command to
```

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.

```
npm install express -g This will produce a similar result but the module will be installed globally. Here, the first
express@4.12.2 /usr/lib/node_modules/express merge-descriptors@1.0.0 utils-
merge@1.0.0 cookie-signature@1.0.6 methods@1.1.1 fresh@0.2.4 cookie@0.1.2 escape-
html@1.0.1 range-parser@1.0.2 content-type@1.0.1 finalhandler@0.3.3 vary@1.0.0 parseurl@1.3.0 content-
disposition@0.5.0 path-to-regexp@0.1.3 depd@1.0.0 qs@2.3.3 on-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-type@0.3.0, mime-types@2.0.10) You can use the following command to check all the modules
npm ls -g Using package.json package.json is present in the root directory of any Node application/module and
{
  "name": "express", "description": "Fast, unopinionated, minimalist web frame-
work", "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": "af-
ter": "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_modules/mocha/bin/mocha --
--require test/support/env --reporter dot --check-leak test/test/acceptance/", "test-
tap": "mocha --require test/support/env --reporter tap --check-leak test/test/acceptance/", "test-
travis": "istanbul cover node_modules/mocha/bin/mocha --report lcovonly --
--require test/support/env --reporter spec --check-leak test/test/acceptance/", "gitHead":
"63ab25579bda70b4927a179b580a9c580b6c7ada", "bugs": "url": "https://github.com/strongloop/express/
express@4.11.2", "shasum": "8df3d5a9ac848585f00a0777601823faecd3b148", "from":
express@*, "npmVersion": "1.4.28", "npmUser": "name": "dougwilson", "email": "doug@somethingd
[name": "tjholowaychuk", "email": "tj@vision-media.ca", "name": "jongleberry", "email": "jonatho
shasum": "8df3d5a9ac848585f00a0777601823faecd3b148", "tarball": "https://registry.npmjs.org/expr
resolved": "https://registry.npmjs.org/express/-/express-4.11.2.tgz", "readme":
"ERROR: No README data found!" Attributes of Package.json name of the package version version of
npm uninstall express Once NPM install the package, you can verify it by looking at the content of /node_m
npm ls Updating a module Update package.json and change the version of the dependency to be updated and run
npm update express Search a module Search a package name using NPM.
npm search express Create a module Creating a module requires package.json to be generated. Let's generate
npm init
```

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* at any time to quit. *name*: (webmaster) You will need to provide all the required information about your mentioned package. *json* file to understand the meaning of various information demanded. Once *package.json* is created, you can run *npm adduser* *Username*: mcmohd *Password*: *Email*: (this is public) mcmohd@gmail.com It is time now to *npm publish* If everything is fine with your module, then it will be published in the repository and will be accessible.

## 10.5 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.jsone two = threefour0 : node1 : /Users/mjr/work/node/process-2.js2 : one3 : two = three4 : four
```



# Chương 11

## 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

### 11.1 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  $m \times n$  matrix  $\text{rand}(3, 2)$   $\text{ans} = 0.13567 \ 0.51230 \ 0.67646$   
 $0.19012 \ 0.76147 \ 0.89694$

Creating a zero  $m \times n$  matrix  $\text{zeros}(3, 2)$   $\text{ans} = 0 \ 0 \ 0 \ 0 \ 0 \ 0$

Creating an  $m \times n$  matrix of ones  $\text{ones}(3, 2)$   $\text{ans} = 1 \ 1 \ 1 \ 1 \ 1 \ 1$

Creating an identity matrix  $\text{eye}(3)$  Diagonal Matrix  $1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1$

Creating a diagonal matrix  $a = [1 \ 2 \ 3]$   $\text{diag}(a)$  Diagonal Matrix  $1 \ 0 \ 0 \ 0 \ 2 \ 0$   
 $0 \ 0 \ 3$  Accessing Matrix Elements Getting the dimension of a matrix  $A = [1 \ 2$   
 $3; 4 \ 5 \ 6]$   $\text{size}(A)$   $\text{ans} = 2 \ 3$

Selecting rows  $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$   $A(1, :)$   $\text{ans} = 1 \ 2 \ 3$   $A(1:2, :)$   $\text{ans} = 1$   
 $2 \ 3 \ 4 \ 5 \ 6$

Selecting columns  $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$   $A(:, 1)$   $\text{ans} = 1 \ 4 \ 7$   $A(:, 1:2)$   $\text{ans}$   
 $= 1 \ 2 \ 4 \ 5 \ 7 \ 8$

Extracting rows and columns by criteria  $A = [1 \ 2 \ 3; 4 \ 5 \ 9; 7 \ 8 \ 9]$   $A(A(:, 3)$   
 $== 9, :)$   $\text{ans} = 4 \ 5 \ 9 \ 7 \ 8 \ 9$

Accessing elements  $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$   $A(1, 1)$   $\text{ans} = 1$   $A(2, 3)$   $\text{ans} =$   
6 Manipulating Shape and Dimensions Converting a matrix into a row vector  
(by column)  $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$   $A = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9$   $A(:)$   $\text{ans} = 1 \ 4 \ 7 \ 2 \ 5$   
 $8 \ 3 \ 6 \ 9$

Converting row to column vectors  $b = [1 \ 2 \ 3]$   $b = 1 \ 2 \ 3$   $b'$   $\text{ans} = 1 \ 2 \ 3$

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 = -0.707110.707110.707110.70711` `eig_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 12

# 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.

## 12.1 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 pro-
gramming 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 pro-
gramming 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 \*h\* 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*, *nottheprogramming*)

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 xxxxxxxx (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 pro-
gramming UNIX a hurdle | | | | | +-----+ DELET-
ING 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 xxxxxxxx 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/html/doc/usr02.html>

## 12.2 Virtual Box

Virtual Box Export and Import VirtualBox VM images? Export Open Virtual-Box 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.

## 12.3 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

# Xác suất

## Chương 13

# Các hàm phân phối thông dụng

Phần này có thêm khảo [Goodfellow u.a. \(2016\)](#) và giáo trình xác suất thống kê của thạc sỹ Trần Thiện Khải, đại học Trà Vinh <sup>1</sup>

17/01/2018 Lòng vòng thế nào hôm nay lại tìm được của bạn Đỗ Minh Hải <sup>2</sup>, rất hay

### 13.0.1 Biến rời rạc

#### Phân phối đều - Discrete Uniform distribution

Là phân phối mà xác suất xuất hiện của các sự kiện là như nhau.  
Biến ngẫu nhiên  $X$  tuân theo phân phối đều rời rạc

$$X \sim \mathcal{U}(a, b)$$

với tham số  $a, b \in \mathbb{Z}; a < b$  là khoảng giá trị của  $X$ , đặt  $n = b - a + 1$

Ta sẽ có:

Định nghĩa	Giá trị
PMF	$p(x) \mid \frac{1}{n}, \forall x \in [a, b]$
CDF - $F(x; a, b)$	$\frac{x - a + 1}{n}, \forall x \in [a, b]$
Kỳ vọng - $E[X]$	$\frac{a + b}{2}$
Phương sai - $Var(X)$	$\frac{n^2 - 1}{12}$

Ví dụ: Lịch chạy của xe buýt tại một trạm xe buýt như sau: chiếc xe buýt đầu tiên trong ngày sẽ khởi hành từ trạm này vào lúc 7 giờ, cứ sau mỗi 15 phút sẽ có một xe khác đến trạm. Giả sử một hành khách đến trạm trong khoảng thời gian từ 7 giờ đến 7 giờ 30. Tìm xác suất để hành khách này chờ:

- Ít hơn 5 phút.
- Ít nhất 12 phút.

**Giải**

<sup>1</sup>[http://www.ctec.tvu.edu.vn/ttkhai/xacsuatthongke\\_dh.htm](http://www.ctec.tvu.edu.vn/ttkhai/xacsuatthongke_dh.htm)

<sup>2</sup><https://dominhhai.github.io/vi/2017/10/prob-com-var>

Gọi  $X$  là số phút sau 7 giờ mà hành khách đến trạm.

Ta có:  $X \sim R[0; 30]$ .

a) Hành khách sẽ chờ ít hơn 5 phút nếu đến trạm giữa 7 giờ 10 và 7 giờ 15 hoặc giữa 7 giờ 25 và 7 giờ 30. Do đó xác suất cần tìm là:

$$P(0 < X < 15) + P(25 < X < 30) = \frac{5}{30} + \frac{5}{30} = \frac{1}{3}$$

b) Hành khách chờ ít nhất 12 phút nếu đến trạm giữa 7 giờ và 7 giờ 3 phút hoặc giữa 7 giờ 15 phút và 7 giờ 18 phút. Xác suất cần tìm là:

$$P(0 < X < 3) + P(15 < X < 18) = \frac{3}{30} + \frac{3}{30} = \frac{1}{5}$$

### Phân phối Béc-nu-li - Bernoulli distribution

Như đã đề cập về phép thử Béc-nu-li rằng mọi phép thử của nó chỉ cho 2 kết quả duy nhất là  $A$  với xác suất  $p$  và  $\bar{A}$  với xác suất  $q = 1 - p$ . Biến ngẫu nhiên  $X$  tuân theo phân phối Béc-nu-li

$$X \sim B(p)$$

với tham số  $p \in \mathbb{R}, 0 \leq p \leq 1$  là xác suất xuất hiện của  $A$  tại mỗi phép thử

Định nghĩa		Giá trị
PMF	$p(x)$	$p(x) \mid p^x(1-p)^{1-x}, x \in \{0, 1\}$
CDF	$F(x; p)$	$\begin{cases} 0 & \text{for } x < 0 \\ 1-p & \text{for } 0 \leq x < 1 \\ 1 & \text{for } x \geq 1 \end{cases}$
Kỳ vọng	$E[X]$	$p$
Phương sai	$Var(X)$	$p(1-p)$

#### Ví dụ

Tham khảo thêm các thuật toán khác tại [Hai \(2018\)](#)

**Phần III**

**Khoa học máy tính**

## Chương 14

# Data Structure and Algorithm

View online <http://magizbox.com/training/danda/site/>

### 14.1 Introduction

Algorithms + Data Structures = Programs

In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently. Data structures can implement one or more particular abstract data types (ADT), which specify the operations that can be performed on a data structure and the computational complexity of those operations. In comparison, a data structure is a concrete implementation of the specification provided by an ADT.

In mathematics and computer science, an algorithm is a self-contained step-by-step set of operations to be performed. Algorithms perform calculation, data processing, and/or automated reasoning tasks.

Software engineering is the study of ways in which to create large and complex computer applications and that generally involve many programmers and designers. At the heart of software engineering is with the overall design of the applications and on the creation of a design that is based on the needs and requirements of end users. While software engineering involves the full life cycle of a software project, it includes many different components - specification, requirements gathering, design, verification, coding, testing, quality assurance, user acceptance testing, production, and ongoing maintenance.

Having an in-depth understanding on every component of software engineering is not mandatory, however, it is important to understand that the subject of data structures and algorithms is concerned with the coding phase. The use of data structures and algorithms is the nuts-and-bolts used by programmers to store and manipulate data.

This article, along with the other examples in this section focuses on the essentials of data structures and algorithms. Attempts will be made to understand how they work, which structure or algorithm is best in a particular situation in an easy to understand environment.

**Data Structures and Algorithms - Defined** A data structure is an arrangement of data in a computer's memory or even disk storage. An example of several common data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables. Algorithms, on the other hand, are used to manipulate the data contained in these data structures as in searching and sorting.

Many algorithms apply directly to a specific data structures. When working with certain data structures you need to know how to insert new data, search for a specified item, and deleting a specific item.

Commonly used algorithms include are useful for:

Searching for a particular data item (or record). Sorting the data. There are many ways to sort data. Simple sorting, Advanced sorting Iterating through all the items in a data structure. (Visiting each item in turn so as to display it or perform some other action on these items)

### 14.1.1 Greedy Algorithm

**Greedy Algorithms** An algorithm is designed to achieve optimum solution for a given problem. In greedy algorithm approach, decisions are made from the given solution domain. As being greedy, the closest solution that seems to provide an optimum solution is chosen.

Greedy algorithms try to find a localized optimum solution, which may eventually lead to globally optimized solutions. However, generally greedy algorithms do not provide globally optimized solutions.

**Counting Coins** This problem is to count to a desired value by choosing the least possible coins and the greedy approach forces the algorithm to pick the largest possible coin. If we are provided coins of 1, 2, 5 and 10 and we are asked to count 18 then the greedy procedure will be

Select one 10 coin, the remaining count is 8

Then select one 5 coin, the remaining count is 3

Then select one 2 coin, the remaining count is 1

And finally, the selection of one 1 coins solves the problem

Though, it seems to be working fine, for this count we need to pick only 4 coins. But if we slightly change the problem then the same approach may not be able to produce the same optimum result.

For the currency system, where we have coins of 1, 7, 10 value, counting coins for value 18 will be absolutely optimum but for count like 15, it may use more coins than necessary. For example, the greedy approach will use  $10 + 1 + 1 + 1 + 1 + 1$ , total 6 coins. Whereas the same problem could be solved by using only 3 coins ( $7 + 7 + 1$ )

Hence, we may conclude that the greedy approach picks an immediate optimized solution and may fail where global optimization is a major concern.

**Examples** Most networking algorithms use the greedy approach. Here is a list of few of them

Travelling Salesman Problem Prim's Minimal Spanning Tree Algorithm Kruskal's Minimal Spanning Tree Algorithm Dijkstra's Minimal Spanning Tree Algorithm Graph - Map Coloring Graph - Vertex Cover Knapsack Problem Job Scheduling Problem

### 14.1.2 Divide and Conquer

In divide and conquer approach, the problem in hand, is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing the subproblems into even smaller sub-problems, we may eventually reach a stage where no more division is possible. Those "atomic" smallest possible sub-problem (fractions) are solved. The solution of all sub-problems is finally merged in order to obtain the solution of an original problem.

Broadly, we can understand divide-and-conquer approach in a three-step process.

**Divide/Break** This step involves breaking the problem into smaller sub-problems. Sub-problems should represent a part of the original problem. This step generally takes a recursive approach to divide the problem until no sub-problem is further divisible. At this stage, sub-problems become atomic in nature but still represent some part of the actual problem.

**Conquer/Solve** This step receives a lot of smaller sub-problems to be solved. Generally, at this level, the problems are considered 'solved' on their own.

**Merge/Combine** When the smaller sub-problems are solved, this stage recursively combines them until they formulate a solution of the original problem. This algorithmic approach works recursively and conquer merge steps works so close that they appear as one.

**Examples** The following computer algorithms are based on divide-and-conquer programming approach

Merge Sort Quick Sort Binary Search Strassen's Matrix Multiplication Closest pair (points) There are various ways available to solve any computer problem, but the mentioned are a good example of divide and conquer approach.

### 14.1.3 Dynamic Programming

Dynamic programming approach is similar to divide and conquer in breaking down the problem into smaller and yet smaller possible sub-problems. But unlike, divide and conquer, these sub-problems are not solved independently. Rather, results of these smaller sub-problems are remembered and used for similar or overlapping sub-problems.

Dynamic programming is used where we have problems, which can be divided into similar sub-problems, so that their results can be re-used. Mostly, these algorithms are used for optimization. Before solving the in-hand sub-problem, dynamic algorithm will try to examine the results of the previously solved sub-problems. The solutions of sub-problems are combined in order to achieve the best solution.

So we can say that

The problem should be able to be divided into smaller overlapping sub-problem. An optimum solution can be achieved by using an optimum solution of smaller sub-problems. Dynamic algorithms use memorization. Comparison In contrast to greedy algorithms, where local optimization is addressed, dynamic algorithms are motivated for an overall optimization of the problem.

In contrast to divide and conquer algorithms, where solutions are combined to achieve an overall solution, dynamic algorithms use the output of a smaller sub-problem and then try to optimize a bigger sub-problem. Dynamic algorithms use memorization to remember the output of already solved sub-problems.



Example The following computer problems can be solved using dynamic programming approach

Fibonacci number series Knapsack problem Tower of Hanoi All pair shortest path by Floyd-Warshall Shortest path by Dijkstra Project scheduling Dynamic programming can be used in both top-down and bottom-up manner. And of course, most of the times, referring to the previous solution output is cheaper than recomputing in terms of CPU cycles.

#### 14.1.4 7 Steps to Solve Algorithm Problems

Today, I viewed the video "7 Steps to Solve Algorithm Problems" by Gayle Laakmann McDowell - the author of Cracking the Coding Interview book. In this video, Gayle describe her method for solve algorithms problems which consists 7 steps: listen carefully, example, brute force, optimize, walk through your algorithms, code and test. In this article, I will summary these steps base on what I learned from this video.

Step 1: Listen carefully Every single detail in a question is necessary to solve it.

The first step is to listen carefully to the problem. So, generally speaking every single detail in a question is necessary to solve that problem - either to solve it all or to solve it optimally. So if there's some detail you haven't used in the question in your algorithm so far think about how you can put that to use because it might be necessary to solve the problem optimally.

Let me give you an example.

You have two arrays, sorted and distinct How did you find the number of elements in common between the two arrays? A lot of people solve this problem and they'll get kind of stuck for awhile and what they'll do is they'll be solving the problem and they'll know the arrays are sorted but they haven't actually used the fact that it's sorted.

This sorting detail - it's not necessary just to find an algorithm but it is necessary to solve the problem optimally.

So remember every single detail in the problem and make sure you use it.

Step 2: Example Make example big, no special cases

The second piece is to come up with a good example, so the last problem that I gave two arrays sorted and distinct compute the number of elements in common, most people's examples look like this.

too small and special case A: 1, 5, 15, 20 B: 2, 5, 13, 30 Yes technically if it's a problem but it's not very useful.

As soon as you glance at this example you notice that there's only one element common and you know exactly what it is and it's obvious because this example is so small and it's actually kind of a special case.

A better example is something like this

larger and avoid special cases A: 1, 5, 15, 20, 30, 37 B: 2, 5, 13, 30, 32, 35, 37, 42 It's much larger and you've avoided some special cases. One of the easiest ways of improving your performance on algorithm questions is just make your examples larger and really avoid special cases.

Step 3: Brute force Better to have a brute force than nothing at all

The third step is to come up with a brute force algorithm. Now I'm not saying you need to go out of your way to come up with something slow, I'm really just saying, hey if the first thing you have is only something really really

slow and terrible that's okay. It is so much better to start off with something slow then to start off with nothing at all. So it's fine if your first algorithm is slow and terrible whatever. However, and this is very very very important, I'm not saying to code the brute force. I'm saying just state your brute force algorithm, state its runtime, and then immediately go to optimizing.

A good chunk of the time on algorithm interview question will often be spent on optimizations. So that's step 4 and spend some good time on it.

Step 4: Optimize The fourth step is optimize and spend some good time on it.

Step 5: Walk through your algorithms Know exactly what you're going to do before coding

what variables data structures? how, why, why do they change? what is the structure of your code Then once you have an optimal algorithm or you're ready to start coding take a step back and just make sure you know exactly what you're going to do in your code.

So many people code prematurely when they aren't really really comfortable with what they're about to do and it ends in disaster. An eighty percent understanding of what you're about to write is really not enough for a whiteboard especially. So take a moment and walk through your algorithm and make sure you know exactly what you're about to do.

Step 6: Code Use space wisely, coding style matters, modularize

Step 6 is to start coding and I'm gonna go into this in a bit of detail. So a couple things to keep in mind particularly when you're coding on a whiteboard. The first couple tips are kind of whiteboard specific but try to write your lines straight. I'm not gonna be judging you on your handwriting and things like that but when people start writing their lines and sharp angles they start to lose track over whether this if statement under this for loop or not. The second thing is use your board space wisely. If you don't need stuff up on the board anymore just erase it. Try to write in this top left corner etc.

Basically give yourself as much space as you possibly can to write your code. If you do run out of space though, it's ok to use arrows, that's fine, I'm really not gonna be judging you on this kind of stuff. So more important things.

Coding style matters (consistent braces, consistent variable naming, consistency spaces, descriptive variables)

Coding style matters even on a whiteboard but on a computer as well, so that means things like braces, naming conventions, or using camel case or underscores, things like that. Those kind of style things absolutely matter. I'm not that concerned over which style you pick, I don't care if you write braces on the same line or the next line but I do care a lot that you have a style and you stick to it. So be consistent in your style. When it comes to variable names, yeah I know it's an annoying to write long variable names on a whiteboard but descriptive variable names are important to good style. So one compromise here is write the good descriptive variable name first and then just ask your interviewer, hey is it okay if I abbreviate this the next time. So that'll be a nice little compromise - you'd show that you care about good variable names but you also don't waste a lot of time.

Modularize (before. not after)

Last thing I want to talk about is modularization. Modularize your code up front and just any little conceptual chunks of code, push that off to another function. So suppose you have three steps in your algorithm - process the first

string, process the second string, and then compare the results. Don't start writing these for loops that walk through each string in the very beginning. Instead write this overall function that wraps these three steps. So step one, step two, step three, and then start drilling in and going into all the details there. Remember any conceptual chunks of code push those off to other functions, don't write them in line.

Step 7: Test Analyse: think about each line, double check things that look weird/risky (for-loop that decrement, math)

Use test cases (smaller test-cases first (faster to run, you will probably be more thorough, edge cases, big test cases)

Then once you're done with the coding you have to start testing your code. One of the mistakes a lot of people do here is they take their big example from step 2 and throw that in as a test case. The problem with that is it's very large so it will take you a long time to run through but also you just used that to develop your code, so if here's an oversight there, the problem will probably repeat itself here.

What's a better step to do, what's a better process to do, is just walk through your code line by line and just think about each line up front not with the test case but just consider, is it really doing the right thing?

Double check anything that looks weird, so for loops that decrement instead of increment and any math at all is a really common place for errors. Just think, look at your code analytically and think what are the most likely places for errors to be and double-check those.

Start with small rather than big

Then once you start with actual test cases start with small test cases rather than big ones. Small test cases work pretty much as effectively as big test cases but they are so much faster to run through, and in fact because they're faster people tend to be much more thorough so you're much more likely to actually find bugs with small test cases than big test cases. So start with small test cases then go in to edge cases after that and then if you have time maybe throw in some big test cases. A couple last techniques with testing. The first one is make sure that when you're testing you're really thinking about what you're doing. A lot of people when they're testing they're just walking through their code almost like they're a bot, and they only look to see if things made sense at the very end when they look at their output. It's much better to really think as you're testing, this way you find the bug as soon as it happens rather than six lines later at the very bottom.

Test your code not your algorithm

The second thing is when you're testing make sure that you're actually testing your code and not your algorithm. An amazing number of people will just take their example and like just walk through it again as though they're just walking through their algorithm but they're never even looking at their code, they're not looking at the exact calculations their code actually did. So make sure that you're really testing your code.

Find bugs

Then the last thing is when you find in a bug, don't panic. Just really think about what caused the bug. A lot of times people will panic and just try to make the first fix that fixes it for that output but they haven't really given it some thought and then they're in a much worse position because if you make the wrong fix to your code, the thing that just fixed the output but didn't fix a

real bug you've not fixed the actual bug, you've made your code more complex, and you potentially introduced a brand new bug and you're in a much worse position. It's much better to just when you find the bug, it's ok, it's not that big of a deal to have a bug it's very normal just really think through what the actual bug, where the actual plug came from.

Remember

think as you test (don't be a bot) test your code, not your algorithm think before you fix bugs. Don't panic! (wrong fixes are worse than no fix) Suggested Reading 7 Steps to Solve Algorithm Problems. Gayle Laakmann McDowell

## 14.2 Data Structures

### 14.2.1 Array

**Arrays** An array is an aggregate data structure that is designed to store a group of objects of the same or different types. Arrays can hold primitives as well as references. The array is the most efficient data structure for storing and accessing a sequence of objects.

Here is the list of most important array features you must know (i.e. be able to program)

copying and cloning insertion and deletion searching and sorting You already know that the Java language has only two data types, primitives and references. Which one is an array? Is it primitive? An array is not a primitive data type - it has a field (and only one), called length. Formally speaking, an array is a reference type, though you cannot find such a class in the Java APIs. Therefore, you deal with arrays as you deal with references. One of the major differences between references and primitives is that you cannot copy arrays by assigning one to another:

`int[] a = {9, 5, 4}; int[] b = a;` The assignment operator creates an alias to the object, like in the picture below

Since these two references `a` and `b` refer to the same object, comparing them with the double equal sign `"=="` will always return true. In the next code example,

`int[] a = {1,2,3}; int[] b = {1,2,3};` `a` and `b` refer to two different objects (though with identical contents). Comparing them with the double equal sign will return false. How would you compare two objects with identical contents? In short, using the equals method. For array comparison, the Java APIs provides the Arrays class.

**The Arrays class** The `java.util.Arrays` class is a convenience class for various array manipulations, like comparison, searching, printing, sorting and others. Basically, this class is a set of static methods that are all useful for working with arrays. The code below demonstrates a proper invocation of equals:

`int[] a = {1,2,3}; int[] b = {1,2,3}; if( Arrays.equals(a, b) ) System.out.println("arrays with identical contents");` Another commonly used method is `toString()` which takes care of printing

`int[] a = {1,2,3}; System.out.println(Arrays.toString(a));` Here is the example of sorting

`int[] a = {3,2,1}; Arrays.sort(a); System.out.println(Arrays.toString(a));` In addition to that, the class has other utility methods for supporting operations over

multidimensional arrays.

Copying arrays There are four ways to copy arrays using a loop structure using `Arrays.copyOf()` using `System.arraycopy()` using `clone()` The first way is very well known to you

```
int[] a = 1, 2, 3; int[] b = new int[a.length]; for(int i= 0; i < a.length; i++)
b[i] = a[i];
```

The next choice is to use `Arrays.copyOf()`

```
int[] a = 1, 2, 3; int[] b = Arrays.copyOf(a, a.length);
```

The second parameter specifies the length of the new array, which could either less or equal or bigger than the original length.

The most efficient copying data between arrays is provided by `System.arraycopy()` method. The method requires five arguments. Here is its signature

```
public static void arraycopy(Object source, int srcIndex, Object destination,
int destIndex, int length)
```

The method copies length elements from a source array starting with the index `srcIndex` to a new array destination at the index `destIndex`. The above code example can be rewritten as it follows

```
int[] a = 1, 2, 3; int[] b = new int[a.length]; System.arraycopy(a, 0, b, 0, 3)
```

And the last copying choice is the use of cloning. Cloning involves creating a new array of the same size and type and copying all the old elements into the new array. The `clone()` method is defined in the `Object` class and its invocation is demonstrated by this code segment

```
int[] a = 1, 2, 3; int[] b = (int[]) a.clone();
```

Note, that casting (`int[]`) is the must.

Examine the code in `ArrayCopyPrimitives.java` for further details.

Insertion and Deletion Arrays in Java have no methods and only one immutable field `length`. Once an array is created, its length is fixed and cannot be changed. What do you do to resize the array? You allocate the array with a different size and copy the contents of the old array to the new array. This code example demonstrates deletion from an array of primitives

```
public char[] delete(char[] data, int pos) if(pos >= 0 pos < data.length)
char[] tmp = new char[data.length-1]; System.arraycopy(data, 0, tmp, 0, pos);
System.arraycopy(data, pos+1, tmp, pos, data.length-pos-1); return tmp; else
return data;
```

The first `arraycopy` copies the elements from index 0 to index `pos-1`, the second `arraycopy` copies the elements from index `pos+1` to `data.length`.

Examine the code in `ArrayDemo.java` for further details.

The `ArrayList` class The `java.util.ArrayList` class supports an idea of a dynamic array - an array that grows and shrinks on demand to accommodate the number of elements in the array. Below is a list of commonly used methods

`add(object)` - adds to the end `add(index, object)` - inserts at the index  
`set(index, object)` - replaces at the index `get(index)` - returns the element at that index  
`remove(index)` - deletes the element at that index `size()` - returns the number of elements  
The following code example will give you a heads up into how some of them are used.

```
/* ADD */ ArrayList<Integer> num = new ArrayList<Integer>(); for(int
i = 0; i < 10; i++) num.add(i); System.out.println(num);
/* REMOVE even integers */ for(int i = 0; i < num.size(); i++) if(num.get(i)%2==0)
num.remove(i); System.out.println(num);
```

Copying arrays of objects This topic is more complex for understanding.. Let us start with a simple loop structure

```
Object[] obj1 = new Integer(10), new StringBuffer("foobar"), new Double(12.95);
Object[] obj2 = new Object[obj1.length]; for(int i = 0; i < obj1.length; i++)
obj2[i] = obj1[i];
```

At the first glance we might think that all data is copied.

In reality, the internal data is shared between two arrays. The figure below illustrates the inner structure

The assignment operator `obj2[i] = obj1[i]` is a crucial part of understanding the concept. You cannot copy references by assigning one to another. The assignment creates an alias rather than a copy. Let us trace down changes in the above picture after execution the following statements

```
obj1[0] = new Integer(5);
and ((StringBuffer) obj1[1]).append('s');
```

As you see, `obj1[0]` and `obj2[0]` now refer to different objects. However, `obj1[1]` and `obj2[1]` refer to the same object (which is "foobars"). Since both arrays shares the data, you must be quite careful when you modify your data, because it might lead to unexpected effects.

The same behavior will take place again, if we use `Arrays.copyOf()`, `System.arraycopy()` and `clone()`. Examine the code example `ArrayCopyReferences.java` for further details.

**Multi-dimensional arrays** In many practical application there is a need to use two- or multi-dimensional arrays. A two-dimensional array can be thought of as a table of rows and columns. This creates a table of 2 rows and 4 columns:

```
int[][] ar1 = new int[2][4];
```

You can create and initialize an array by using nested curly braces. For example, this creates a table of 3 rows and 2 columns:

```
int[][] ar2 = {1,2,3,4,5,6};
```

Generally speaking, a two-dimensional array is not exactly a table - each row in such array can have a different length. Consider this code fragment

```
Object[][] obj = new Integer(1), new Integer(2), new Integer(10), "bozo",
new Double(1.95);
```

The accompanying picture sheds a bit of light on internal representation

From the picture you clearly see that a two-dimensional array in Java is an array of arrays. The array `obj` has two elements `obj[0]` and `obj[1]` that are arrays of length 2 and 3 respectively.

**Cloning 2D arrays** The procedure is even more confusing and less expected. Consider the following code segment

```
Object[][] obj = new Integer(1), new Integer(2), new Integer(10), "bozo", new
Double(1.95);
```

```
Object[][] twin = (Object[][]) obj.clone();
```

The procedure of cloning 2d arrays is virtually the same as cloning an array of references. Unfortunately, built-in `clone()` method does not actually clone each row, but rather creates references to them. Here is a graphical interpretation of the above code

Let us change the value of `obj[1][1]`

```
obj[1][1] = "xyz";
```

This assignment effects the value of `twin[1][1]` as well

Such a copy is called a "shallow" copy. The default behavior of `clone()` is to return a shallow copy of the object. If we want a "deep" copy instead, we must provide our own implementation by overriding `Object's clone()` method.

The idea of a "deep" copy is simple - it makes a distinct copy of each of the object's fields, recursing through the entire object. A deep copy is thus a completely separate object from the original; changes to it don't affect the original, and vice versa. In relevance to the above code, here is a deep clone graphically

Further, making a complete deep copy is not always needed. Consider an array of immutable objects. As we know, immutable objects cannot be modified,

allowing clients to share the same instance without interfering with each other. In this case there is no need to clone them, which leads to the following picture

Always in this course we will create data structures of immutable objects, therefore implementing the clone method will require copying a structure (a shape) and sharing its internal data. We will discuss these issues later on in the course.

Challenges "Arrays: Left Rotation". hackerrank. 2016 References "Array Data Structure". Victor S.Adamchik, CMU. 2009

### 14.2.2 Linked List

A linked list is a sequence of data structures, which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

Link Each link of a linked list can store a data called an element. Next Each link of a linked list contains a link to the next link called Next. LinkedList A Linked List contains the connection link to the first link called First. Representation Linked list can be visualized as a chain of nodes, where every node points to the next node.

As per the above illustration, following are the important points to be considered.

Linked List contains a link element called first. Each link carries a data field(s) and a link field called next. Each link is linked with its next link using its next link. Last link carries a link as null to mark the end of the list. Types of Linked List Following are the various types of linked list.

Simple Linked List Item navigation is forward only. Doubly Linked List Items can be navigated forward and backward. Circular Linked List Last item contains link of the first element as next and the first element has a link to the last element as previous. Basic Operations Following are the basic operations supported by a list.

Insertion Adds an element at the beginning of the list. Deletion Deletes an element at the beginning of the list. Display Displays the complete list. Search Searches an element using the given key. Delete Deletes an element using the given key. Insertion Operation Adding a new node in linked list is a more than one step activity. We shall learn this with diagrams here. First, create a node using the same structure and find the location where it has to be inserted.

Imagine that we are inserting a node B (NewNode), between A (LeftNode) and C (RightNode). Then point B.next to C

NewNode.next > RightNode; It should look like this

Now, the next node at the left should point to the new node.

LeftNode.next > NewNode;

This will put the new node in the middle of the two. The new list should look like this

Similar steps should be taken if the node is being inserted at the beginning of the list. While inserting it at the end, the second last node of the list should point to the new node and the new node will point to NULL.

**Deletion Operation** Deletion is also a more than one step process. We shall learn with pictorial representation. First, locate the target node to be removed, by using searching algorithms.

The left (previous) node of the target node now should point to the next node of the target node

`LeftNode.next > TargetNode.next;`

This will remove the link that was pointing to the target node. Now, using the following code, we will remove what the target node is pointing at.

`TargetNode.next > NULL;`

We need to use the deleted node. We can keep that in memory otherwise we can simply deallocate memory and wipe off the target node completely.

**Reverse Operation** This operation is a thorough one. We need to make the last node to be pointed by the head node and reverse the whole linked list.

First, we traverse to the end of the list. It should be pointing to NULL. Now, we shall make it point to its previous node

We have to make sure that the last node is not the lost node. So we'll have some temp node, which looks like the head node pointing to the last node. Now, we shall make all left side nodes point to their previous nodes one by one.

Except the node (first node) pointed by the head node, all nodes should point to their predecessor, making them their new successor. The first node will point to NULL.

We'll make the head node point to the new first node by using the temp node.

The linked list is now reversed.

### 14.2.3 Stack and Queue

An array is a random access data structure, where each element can be accessed directly and in constant time. A typical illustration of random access is a book - each page of the book can be open independently of others. Random access is critical to many algorithms, for example binary search.

A linked list is a sequential access data structure, where each element can be accessed only in particular order. A typical illustration of sequential access is a roll of paper or tape - all prior material must be unrolled in order to get to data you want.

In this note we consider a subcase of sequential data structures, so-called limited access data structures.

**Stacks** A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. push adds an item to the top of the stack, pop removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top. A stack is a recursive data structure. Here is a structural definition of a Stack:

a stack is either empty or it consists of a top and the rest which is a stack;

**Applications** The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack. Another application is an "undo" mechanism in text editors; this operation is



accomplished by keeping all text changes in a stack. Backtracking. This is a process when you need to access the most recent data element in a series of elements. Think of a labyrinth or maze - how do you find a way from an entrance to an exit? Once you reach a dead end, you must backtrack. But backtrack to where? to the previous choice point. Therefore, at each choice point you store on a stack all possible choices. Then backtracking simply means popping a next choice from the stack.

Language processing: space for parameters and local variables is created internally using a stack. compiler's syntax check for matching braces is implemented by using stack. support for recursion Implementation In the standard library of classes, the data type stack is an adapter class, meaning that a stack is built on top of other data structures. The underlying structure for a stack could be an array, a vector, an ArrayList, a linked list, or any other collection. Regardless of the type of the underlying data structure, a Stack must implement the same functionality. This is achieved by providing a unique interface:

```
public interface StackInterface<AnyType> {
    public void push(AnyType e);
    public AnyType pop();
    public AnyType peek();
    public boolean isEmpty();
```

The following picture demonstrates the idea of implementation by composition.

Another implementation requirement (in addition to the above interface) is that all stack operations must run in constant time  $O(1)$ . Constant time means that there is some constant  $k$  such that an operation takes  $k$  nanoseconds of computational time regardless of the stack size.

#### Array-based implementation

In an array-based implementation we maintain the following fields: an array  $A$  of a default size (1), the variable  $top$  that refers to the top element in the stack and the capacity that refers to the array size. The variable  $top$  changes from  $-1$  to  $capacity - 1$ . We say that a stack is empty when  $top = -1$ , and the stack is full when  $top = capacity - 1$ . In a fixed-size stack abstraction, the capacity stays unchanged, therefore when  $top$  reaches capacity, the stack object throws an exception. See `ArrayStack.java` for a complete implementation of the stack class.

In a dynamic stack abstraction when  $top$  reaches capacity, we double up the stack size.

#### Linked List-based implementation

Linked List-based implementation provides the best (from the efficiency point of view) dynamic stack implementation. See `ListStack.java` for a complete implementation of the stack class.

**Queues** A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle. An excellent example of a queue is a line of students in the food court of the UC. New additions to a line made to the back of the queue, while removal (or serving) happens in the front. In the queue only two operations are allowed enqueue and dequeue. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item. The picture demonstrates the FIFO access. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

**Implementation** In the standard library of classes, the data type queue is an adapter class, meaning that a queue is built on top of other data structures. The underlying structure for a queue could be an array, a Vector, an ArrayList, a LinkedList, or any other collection. Regardless of the type of the underlying data structure, a queue must implement the same functionality. This is achieved by providing a unique interface.

```
interface QueueInterface<AnyType> {
    public boolean isEmpty();
    public AnyType getFront();
    public AnyType dequeue();
    public void enqueue(AnyType e);
    public void clear();
}
```

Each of the above basic operations must run at constant time  $O(1)$ . The following picture demonstrates the idea of implementation by composition.

**Circular Queue** Given an array A of a default size ( 1) with two references back and front, originally set to -1 and 0 respectively. Each time we insert (enqueue) a new item, we increase the back index; when we remove (dequeue) an item - we increase the front index. Here is a picture that illustrates the model after a few steps:

As you see from the picture, the queue logically moves in the array from left to right. After several moves back reaches the end, leaving no space for adding new elements

However, there is a free space before the front index. We shall use that space for enqueueing new items, i.e. the next entry will be stored at index 0, then 1, until front. Such a model is called a wrap around queue or a circular queue

Finally, when back reaches front, the queue is full. There are two choices to handle a full queue: a) throw an exception; b) double the array size.

The circular queue implementation is done by using the modulo operator (denoted

See ArrayQueue.java for a complete implementation of a circular queue.

**Applications** The simplest two search techniques are known as Depth-First Search (DFS) and Breadth-First Search (BFS). These two searches are described by looking at how the search tree (representing all the possible paths from the start) will be traversed.

**Depth-First Search with a Stack** In depth-first search we go down a path until we get to a dead end; then we backtrack or back up (by popping a stack) to get an alternative path.

**Create a stack** Create a new choice point Push the choice point onto the stack while (not found and stack is not empty) Pop the stack Find all possible choices after the last one tried Push these choices onto the stack Return **Breadth-First Search with a Queue** In breadth-first search we explore all the nearest possibilities by finding all possible successors and enqueue them to a queue.

**Create a queue** Create a new choice point Enqueue the choice point onto the queue while (not found and queue is not empty) Dequeue the queue Find all possible choices after the last one tried Enqueue these choices onto the queue Return We will see more on search techniques later in the course.

**Arithmetic Expression Evaluation** An important application of stacks is in parsing. For example, a compiler must parse arithmetic expressions written using infix notation:

$1 + ((2 + 3) * 4 + 5) * 6$  We break the problem of parsing infix expressions into two stages. First, we convert from infix to a different representation

called postfix. Then we parse the postfix expression, which is a somewhat easier problem than directly parsing infix.

Converting from Infix to Postfix. Typically, we deal with expressions in infix notation

$2 + 5$  where the operators (e.g.  $+$ ,  $*$ ) are written between the operands (e.g. 2 and 5). Writing the operators after the operands gives a postfix expression 2 and 5 are called operands, and the  $+$  is operator. The above arithmetic expression is called infix, since the operator is in between operands. The expression

$2\ 5\ +$  Writing the operators before the operands gives a prefix expression

$+2\ 5$  Suppose you want to compute the cost of your shopping trip. To do so, you add a list of numbers and multiply them by the local sales tax (7.25

$70 + 150 * 1.0725$  Depending on the calculator, the answer would be either 235.95 or 230.875. To avoid this confusion we shall use a postfix notation

$70\ 150\ +\ 1.0725\ *$  Postfix has the nice property that parentheses are unnecessary.

Now, we describe how to convert from infix to postfix.

Read in the tokens one at a time If a token is an integer, write it into the output If a token is an operator, push it to the stack, if the stack is empty. If the stack is not empty, you pop entries with higher or equal priority and only then you push that 1. token to the stack. If a token is a left parentheses '(', push it to the stack If a token is a right parentheses ')', you pop entries until you meet '('. When you finish reading the string, you pop up all tokens which are left there. Arithmetic precedence is in increasing order: '+', '-', '\*', '/'; Example. Suppose we have an infix expression:  $2+(4+3*2+1)/3$ . We read the string by characters. '2' - send to the output. '+' - push on the stack. '(' - push on the stack. '4' - send to the output. '+' - push on the stack. '3' - send to the output. '\*' - push on the stack. '2' - send to the output. Evaluating a Postfix Expression. We describe how to parse and evaluate a postfix expression.

We read the tokens in one at a time. If it is an integer, push it on the stack If it is a binary operator, pop the top two elements from the stack, apply the operator, and push the result back on the stack. Consider the following postfix expression

$5\ 9\ 3\ +\ 4\ 2\ *\ * \ 7\ +\ *$  Here is a chain of operations

Stack Operations Output ————— push(5); 5 push(9); 5 9 push(3); 5 9 3 push(pop() + pop()) 5 12 push(4); 5 12 4 push(2); 5 12 4 2 push(pop() \* pop()) 5 12 8 push(pop() \* pop()) 5 96 push(7) 5 96 7 push(pop() + pop()) 5 103 push(pop() \* pop()) 515 Note, that division is not a commutative operation, so  $2/3$  is not the same as  $3/2$ .

Challenges Stacks: Balanced Brackets Queues: A Tale of Two Stacks References "Stacks and Queues". Victor S.Adamchik, CMU. 2009

#### 14.2.4 Tree

Binary Tree Fundamentally, a binary tree is composed of nodes connected by edges (with further restrictions discussed below). Some binary tree,  $tt$ , is either empty or consists of a single root element with two distinct binary tree child elements known as the left subtree and the right subtree of  $tt$ . As the name binary suggests, a node in a binary tree has a maximum of 2 children.

The following diagrams depict two different binary trees:

Here are the basic facts and terms to know about binary trees:

The convention for binary tree diagrams is that the root is at the top, and the subtrees branch down from it. A node's left and right subtrees are referred to as children, and that node can be referred to as the parent of those subtrees. A non-root node with no children is called a leaf. Some node *aa* is an ancestor of some node *bb* if *bb* is located in a left or right subtree whose root node is *aa*. This means that the root node of binary tree *tt* is the ancestor of all other nodes in the tree. If some node *aa* is an ancestor of some node *bb*, then the path from *aa* to *bb* is the sequence of nodes starting with *aa*, moving down the ancestral chain of children, and ending with *bb*. The depth (or level) of some node *aa* is its distance (i.e., number of edges) from the tree's root node. Simply put, the height of a tree is the number of edges between the root node and its furthest leaf. More technically put, it's  $1 + \max(\text{height}(\text{leftSubtree}), \text{height}(\text{rightSubtree}))$  (i.e., one more than the maximum of the heights of its left and right subtrees). Any node has a height of 11, and the height of an empty subtree is 11. Because the height of each node is  $1 + 1 +$  the maximum height of its subtrees and an empty subtree's height is 11, the height of a single-element tree or leaf node is 00. Let's apply some of the terms we learned above to the binary tree on the right:

The root node is AA.

The respective left and right children of AA are BB and EE. The left child of BB is CC. The respective left and right children of EE are FF and DD.

Nodes CC, FF, and DD are leaves (i.e., each node is a leaf).

The root is the ancestor of all other nodes, BB is an ancestor of CC, and EE is an ancestor of FF and DD.

The path between AA and CC is ABCABC. The path between AA and FF is AEFAEF. The path between AA and DD is A EDED.

The depth of root node AA is 00. The depth of nodes BB and EE is 11. The depth of nodes CC, FF, and DD, is 22.

The height of the tree,  $\text{height}(t)$ , is 22. We calculate this recursively as  $\text{height}(t) = 1 + (\max(\text{height}(\text{root.leftChild}), \text{height}(\text{root.rightChild})))$

Because this is long and complicated when expanded, we'll break it down using an image of a slightly simpler version of *tt* whose height is still 22:

Binary Search Tree A Binary Search Tree (BST), *tt*, is a binary tree that is either empty or satisfies the following three conditions:

Each element in the left subtree of *tt* is less than or equal to the root element of *tt* (i.e.,  $\max(\text{leftTree}(t).value) \leq t.value$ ).

Each element in the right subtree of *tt* is greater than the root element of *tt* (i.e.,  $\max(\text{rightTree}(t).value) > t.value$ ).

Both *leftTree(t)* and *rightTree(t)* are BSTs.

You can essentially think of it as a regular binary tree where for each node parent having a *leftChild* and *rightChild*, *leftChild.parent* and *rightChild.parent*. In the first diagram at the top of this article, the binary tree of integers on the left side is a binary search tree.

Advantages and Drawbacks Searching for elements is very fast. We know that each node has a maximum of two children and we know that the items are always in the left subtree and the » items are always in the right subtree. To search for an element, we simply need to compare the value we want against the value stored in the root node of the current subtree and work our way down the appropriate child subtrees until we either find the value we're looking for or

we hit null (i.e., an empty subtree) which indicates the item is not in the BST. Inserting or searching for a node in a balanced tree is  $O(\log n)O(\log n)$  because you're discarding half of the possible values each time you go left or right.

It can easily become unbalanced. Depending on the insertion order, the tree can very easily become unbalanced (which makes for longer search times). For example, if we create a new tree where the sequence of inserted nodes is 213456213456, we end up with the following unbalanced tree:

Observe that the root's left subtree only has one node, whereas the root's right subtree has four nodes. For this reason, inserting or searching for a node in an unbalanced tree is  $O(n)O(n)$ .

Challenges "Trees: Is This a Binary Search Tree?". hackerrank. 2016 References "Binary Trees and Binary Search Trees". AllisonP, hackerrank. 2016

### 14.2.5 Binary Search Tree

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties

The left sub-tree of a node has a key less than or equal to its parent node's key. The right sub-tree of a node has a key greater than to its parent node's key. Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as

*left<sub>s</sub>ubtree(keys)node(key)right<sub>s</sub>ubtree(keys)RepresentationBSTisacollectionofnodesarrangedinaway*

Following is a pictorial representation of BST

We observe that the root node key (27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

Basic Operations Following are the basic operations of a tree

Search Searches an element in a tree. Insert Inserts an element in a tree.

Pre-order Traversal Traverses a tree in a pre-order manner. In-order Traversal Traverses a tree in an in-order manner. Post-order Traversal Traverses a tree in a post-order manner. Node Define a node having some data, references to its left and right child nodes.

struct node { int data; struct node \*leftChild; struct node \*rightChild; }; Search Operation Whenever an element is to be searched, start searching from the root node. Then if the data is less than the key value, search for the element in the left subtree. Otherwise, search for the element in the right subtree. Follow the same algorithm for each node.

Algorithm

```
struct node* search(int data) { struct node *current = root; printf("Visiting elements: "); while(current->data != data) { if(current != NULL) { printf(" //go to left tree if(current->data > data) current = current->leftChild; //else go to right tree else current = current->rightChild;
```

```
//not found if(current == NULL) return NULL; return current; } Insert Operation Whenever an element is to be inserted, first locate its proper location. Start searching from the root node, then if the data is less than the key value, search for the empty location in the left subtree and insert the data. Otherwise, search for the empty location in the right subtree and insert the data.
```

Algorithm

```

void insert(int data) struct node *tempNode = (struct node*) malloc(sizeof(struct
node)); struct node *current; struct node *parent;
tempNode->data = data; tempNode->leftChild = NULL; tempNode->rightChild
= NULL;
//if tree is empty if(root == NULL) root = tempNode; else current =
root; parent = NULL;
while(1) parent = current;
//go to left of the tree if(data < parent->data) current = current->leftChild;
//insert to the left
if(current == NULL) parent->leftChild = tempNode; return; //go to right
of the tree else current = current->rightChild;
//insert to the right if(current == NULL) parent->rightChild = tempNode;
return;

```

### 14.3 Heaps

A heap is just what it sounds like — a pile of values organized into a binary tree-like structure adhering to some ordering property. When we add elements to a heap, we fill this tree-like structure from left to right, level by level. This makes heaps really easy to implement in an array, where the value for some index  $i$ 's left child is located at index  $2i+1$  and the value for its right child is at index  $2i+2$  (using zero-indexing). Here are the two most fundamental heap operations:

**add:** Insert an element into the heap. You may also see this referred to as **push**. **poll:** Retrieve and remove the root element of the heap. You may also see this referred to as **pop**. **Max Heap** This type heap orders the maximum value at the root.

When we add the values 12341234 to a Max heap, it looks like this:

When we poll the same Max heap until it's empty, it looks like this:

**Min Heap** This type of heap orders the minimum value at the root.

When we add the values 12341234 to a Min heap, it looks like this:

When we poll the same Min heap until it's empty, it looks like this:

**Applications** The heap data structure has many applications.

**Heapsort:** One of the best sorting methods being in-place and with no quadratic worst-case scenarios. **Selection algorithms:** A heap allows access to the min or max element in constant time, and other selections (such as median or  $k$ th-element) can be done in sub-linear time on data that is in a heap. **Graph algorithms:** By using heaps as internal traversal data structures, run time will be reduced by polynomial order. Examples of such problems are Prim's minimal-spanning-tree algorithm and Dijkstra's shortest-path algorithm. **Priority Queue:** A priority queue is an abstract concept like "a list" or "a map"; just as a list can be implemented with a linked list or an array, a priority queue can be implemented with a heap or a variety of other methods. **Order statistics:** The Heap data structure can be used to efficiently find the  $k$ th smallest (or largest) element in an array. **Challenges** "Heaps: Find the Running Median". hackerrank. 2016 **References** "Heaps". AllisonP, hackerrank. 2016 "Heap (data structure)". wikipedia. 2016

## 14.4 Sort

### 14.4.1 Introduction

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats. Following are some of the examples of sorting in real-life scenarios

**Telephone Directory** The telephone directory stores the telephone numbers of people sorted by their names, so that the names can be searched easily. **Dictionary** The dictionary stores words in an alphabetical order so that searching of any word becomes easy. **In-place Sorting and Not-in-place Sorting** Sorting algorithms may require some extra space for comparison and temporary storage of few data elements. These algorithms do not require any extra space and sorting is said to happen in-place, or for example, within the array itself. This is called in-place sorting. Bubble sort is an example of in-place sorting.

However, in some sorting algorithms, the program requires space which is more than or equal to the elements being sorted. Sorting which uses equal or more space is called not-in-place sorting. Merge-sort is an example of not-in-place sorting.

**Stable and Not Stable Sorting** If a sorting algorithm, after sorting the contents, does not change the sequence of similar content in which they appear, it is called stable sorting.

If a sorting algorithm, after sorting the contents, changes the sequence of similar content in which they appear, it is called unstable sorting.

Stability of an algorithm matters when we wish to maintain the sequence of original elements, like in a tuple for example.

**Adaptive and Non-Adaptive Sorting Algorithm** A sorting algorithm is said to be adaptive, if it takes advantage of already 'sorted' elements in the list that is to be sorted. That is, while sorting if the source list has some element already sorted, adaptive algorithms will take this into account and will try not to re-order them.

A non-adaptive algorithm is one which does not take into account the elements which are already sorted. They try to force every single element to be re-ordered to confirm their sortedness.

**Important Terms** Some terms are generally coined while discussing sorting techniques, here is a brief introduction to them

#### Increasing Order

A sequence of values is said to be in increasing order, if the successive element is greater than the previous one. For example, 1, 3, 4, 6, 8, 9 are in increasing order, as every next element is greater than the previous element.

#### Decreasing Order

A sequence of values is said to be in decreasing order, if the successive element is less than the current one. For example, 9, 8, 6, 4, 3, 1 are in decreasing order, as every next element is less than the previous element.

#### Non-Increasing Order

A sequence of values is said to be in non-increasing order, if the successive element is less than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values. For example, 9, 8, 6, 3, 3, 1 are in non-increasing order, as every next element is less than or equal to (in case of 3) but not greater than any previous element.

#### Non-Decreasing Order

A sequence of values is said to be in non-decreasing order, if the successive element is greater than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values. For example, 1, 3, 3, 6, 8, 9 are in non-decreasing order, as every next element is greater than or equal to (in case of 3) but not less than the previous one.

### 14.4.2 Bubble Sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of  $O(n^2)$  where  $n$  is the number of items.

How Bubble Sort Works? We take an unsorted array for our example. Bubble sort takes  $O(n^2)$  time so we're keeping it short and precise.

Bubble sort starts with very first two elements, comparing them to check which one is greater.

In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

We find that 27 is smaller than 33 and these two values must be swapped.

The new array should look like this

Next we compare 33 and 35. We find that both are in already sorted positions.

Then we move to the next two values, 35 and 10.

We know then that 10 is smaller 35. Hence they are not sorted.

We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this

To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this

Notice that after each iteration, at least one value moves at the end.

And when there's no swap required, bubble sorts learns that an array is completely sorted.

Now we should look into some practical aspects of bubble sort.

Algorithm We assume list is an array of  $n$  elements. We further assume that swap function swaps the values of the given array elements.

```
begin BubbleSort(list)
  for all elements of list if list[i] > list[i+1] swap(list[i], list[i+1]) end if end for
  return list
```

end BubbleSort Pseudocode We observe in algorithm that Bubble Sort compares each pair of array element unless the whole array is completely sorted in an ascending order. This may cause a few complexity issues like what if the array needs no more swapping as all the elements are already ascending.



To ease-out the issue, we use one flag variable swapped which will help us see if any swap has happened or not. If no swap has occurred, i.e. the array requires no more processing to be sorted, it will come out of the loop.

Pseudocode of BubbleSort algorithm can be written as follows

```

procedure bubbleSort( list : array of items )
  loop = list.count;
  for i = 0 to loop-1 do: swapped = false
    for j = 0 to loop-1 do:
      /* compare the adjacent elements */ if list[j] > list[j+1] then /* swap them
      */ swap( list[j], list[j+1] ) swapped = true end if
    end for
    /*if no number was swapped that means array is sorted now, break the
    loop.*/
    if(not swapped) then break end if
  end for

```

end procedure return list Implementation One more issue we did not address in our original algorithm and its improvised pseudocode, is that, after every iteration the highest values settles down at the end of the array. Hence, the next iteration need not include already sorted elements. For this purpose, in our implementation, we restrict the inner loop to avoid already sorted values.

### 14.4.3 Insertion Sort

This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be 'insert'ed in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort.

The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are of  $(n^2)$ , where  $n$  is the number of items.

How Insertion Sort Works? We take an unsorted array for our example.

Insertion sort compares the first two elements.

It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.

Insertion sort moves ahead and compares 33 with 27.

And finds that 33 is not in the correct position.

It swaps 33 with 27. It also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14, and 27 is greater than 14. Hence, the sorted sub-list remains sorted after swapping.

By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10.

These values are not in a sorted order.

So we swap them.

However, swapping makes 27 and 10 unsorted.

Hence, we swap them too.

Again we find 14 and 10 in an unsorted order.

We swap them again. By the end of third iteration, we have a sorted sub-list of 4 items.

This process goes on until all the unsorted values are covered in a sorted sub-list. Now we shall see some programming aspects of insertion sort.

Algorithm Now we have a bigger picture of how this sorting technique works, so we can derive simple steps by which we can achieve insertion sort.

Step 1 If it is the first element, it is already sorted. return 1; Step 2 Pick next element Step 3 Compare with all elements in the sorted sub-list Step 4 Shift all the elements in the sorted sub-list that is greater than the value to be sorted Step 5 Insert the value Step 6 Repeat until list is sorted Pseudocode procedure insertionSort( A : array of items ) int holePosition int valueToInsert  
 for i = 1 to length(A) inclusive do:  
   /\* select value to be inserted \*/ valueToInsert = A[i] holePosition = i  
   /\*locate hole position for the element to be inserted \*/  
   while holePosition > 0 and A[holePosition-1] > valueToInsert do: A[holePosition]  
 = A[holePosition-1] holePosition = holePosition -1 end while  
   /\* insert the number at hole position \*/ A[holePosition] = valueToInsert  
 end for  
 end procedure

#### 14.4.4 Selection Sort

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of  $O(n^2)$ , where  $n$  is the number of items.

How Selection Sort Works? Consider the following depicted array as an example.

For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

So we replace 14 with 10. After one iteration 10, which happens to be the minimum value in the list, appears in the first position of the sorted list.

For the second position, where 33 is residing, we start scanning the rest of the list in a linear manner.

We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values.

After two iterations, two least values are positioned at the beginning in a sorted manner.

The same process is applied to the rest of the items in the array.

Following is a pictorial depiction of the entire sorting process

Now, let us learn some programming aspects of selection sort.

Algorithm Step 1 Set MIN to location 0 Step 2 Search the minimum element in the list Step 3 Swap with value at location MIN Step 4 Increment MIN to point to next element Step 5 Repeat until list is sorted Pseudocode procedure selection sort list : array of items n : size of list

for i = 1 to n - 1 /\* set current element as minimum \*/ min = i

```

/* check the element to be minimum */
for j = i+1 to n if list[j] < list[min] then min = j; end if end for
/* swap the minimum element with the current element*/ if indexMin != i
then swap list[min] and list[i] end if
end for
end procedure

```

#### 14.4.5 Merge Sort

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being  $O(n \log n)$ , it is one of the most respected algorithms.

Merge sort first divides the array into equal halves and then combines them in a sorted manner.

How Merge Sort Works? To understand merge sort, we take an unsorted array as the following

We know that merge sort first divides the whole array iteratively into equal halves unless the atomic values are achieved. We see here that an array of 8 items is divided into two arrays of size 4.

This does not change the sequence of appearance of items in the original. Now we divide these two arrays into halves.

We further divide these arrays and we achieve atomic value which can no more be divided.

Now, we combine them in exactly the same manner as they were broken down. Please note the color codes given to these lists.

We first compare the element for each list and then combine them into another list in a sorted manner. We see that 14 and 33 are in sorted positions. We compare 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.

In the next iteration of the combining phase, we compare lists of two data values, and merge them into a list of found data values placing all in a sorted order.

After the final merging, the list should look like this

Now we should learn some programming aspects of merge sorting.

Algorithm Merge sort keeps on dividing the list into equal halves until it can no more be divided. By definition, if it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

Step 1 if it is only one element in the list it is already sorted, return. Step 2 divide the list recursively into two halves until it can no more be divided. Step 3 merge the smaller lists into new list in sorted order. Pseudocode We shall now see the pseudocodes for merge sort functions. As our algorithms point out two main functions divide merge.

Merge sort works with recursion and we shall see our implementation in the same way.

```

procedure mergesort( var a as array ) if ( n == 1 ) return a
var l1 as array = a[0] ... a[n/2] var l2 as array = a[n/2+1] ... a[n]
l1 = mergesort( l1 ) l2 = mergesort( l2 )
return merge( l1, l2 ) end procedure
procedure merge( var a as array, var b as array )

```

```

var c as array
while ( a and b have elements ) if ( a[0] > b[0] ) add b[0] to the end of c
remove b[0] from b else add a[0] to the end of c remove a[0] from a end if end
while
    while ( a has elements ) add a[0] to the end of c remove a[0] from a end while
    while ( b has elements ) add b[0] to the end of c remove b[0] from b end
while
return c
end procedure

```

#### 14.4.6 Shell Sort

Shell sort is a highly efficient sorting algorithm and is based on insertion sort algorithm. This algorithm avoids large shifts as in case of insertion sort, if the smaller value is to the far right and has to be moved to the far left.

This algorithm uses insertion sort on a widely spread elements, first to sort them and then sorts the less widely spaced elements. This spacing is termed as interval. This interval is calculated based on Knuth's formula as

Knuth's Formula  $h = h/3 + 1$

where

hh is interval with initial value 1 This algorithm is quite efficient for medium-sized data sets as its average and worst case complexity are of  $O(n^2)$ , where nn is the number of items.

How Shell Sort Works? Let us consider the following example to have an idea of how shell sort works. We take the same array we have used in our previous examples. For our example and ease of understanding, we take the interval of 4. Make a virtual sub-list of all values located at the interval of 4 positions. Here these values are 35, 14, 33, 19, 42, 27 and 10, 44

We compare values in each sub-list and swap them (if necessary) in the original array. After this step, the new array should look like this

Then, we take interval of 2 and this gap generates two sub-lists - 14, 27, 35, 42, 19, 10, 33, 44

We compare and swap the values, if required, in the original array. After this step, the array should look like this

Finally, we sort the rest of the array using interval of value 1. Shell sort uses insertion sort to sort the array.

Following is the step-by-step depiction

We see that it required only four swaps to sort the rest of the array.

Algorithm Following is the algorithm for shell sort.

Step 1 Initialize the value of h Step 2 Divide the list into smaller sub-list of equal interval h Step 3 Sort these sub-lists using insertion sort Step 3 Repeat until complete list is sorted Pseudocode Following is the pseudocode for shell sort.

```

procedure shellSort() A : array of items
/* calculate interval */ while interval < A.length /3 do: interval = interval *
3 + 1 end while
while interval > 0 do:
for outer = interval; outer < A.length; outer ++ do:
/* select value to be inserted */ valueToInsert = A[outer] inner = outer;

```

```

/*shift element towards right*/ while inner > interval -1 A[inner - interval]
>= valueToInsert do: A[inner] = A[inner - interval] inner = inner - interval end
while
/* insert the number at hole position */ A[inner] = valueToInsert
end for
/* calculate interval*/ interval = (interval -1) /3;
end while
end procedure

```

#### 14.4.7 Quick Sort

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Quick sort partitions an array and then calls itself recursively twice to sort the two resulting subarrays. This algorithm is quite efficient for large-sized data sets as its average and worst case complexity are of  $(n^2)$ , where  $n$  is the number of items.

Partition in Quick Sort Following animated representation explains how to find the pivot value in an array.

The pivot value divides the list into two parts. And recursively, we find the pivot for each sub-lists until all lists contains only one element.

Quick Sort Pivot Algorithm Based on our understanding of partitioning in quick sort, we will now try to write an algorithm for it, which is as follows.

Step 1 Choose the highest index value has pivot Step 2 Take two variables to point left and right of the list excluding pivot Step 3 left points to the low index Step 4 right points to the high Step 5 while value at left is less than pivot move right Step 6 while value at right is greater than pivot move left Step 7 if both step 5 and step 6 does not match swap left and right Step 8 if left right, the point where they met is new pivot Quick Sort Pivot Pseudocode The pseudocode for the above algorithm can be derived as

```

function partitionFunc(left, right, pivot) leftPointer = left rightPointer =
right - 1
while True do while A[++leftPointer] < pivot do //do-nothing end while
while rightPointer > 0 A[--rightPointer] > pivot do //do-nothing end while
if leftPointer >= rightPointer break else swap leftPointer,rightPointer end if
end while
swap leftPointer,right return leftPointer
end function

```

Quick Sort Algorithm Using pivot algorithm recursively, we end up with smaller possible partitions. Each partition is then processed for quick sort. We define recursive algorithm for quicksort as follows

Step 1 Make the right-most index value pivot Step 2 partition the array using pivot value Step 3 quicksort left partition recursively Step 4 quicksort right partition recursively Quick Sort Pseudocode To get more into it, let see the pseudocode for quick sort algorithm

```

procedure quickSort(left, right)
if right-left <= 0 return else pivot = A[right] partition = partitionFunc(left,
right, pivot) quickSort(left,partition-1) quickSort(partition+1,right) end if

```

end procedure

## 14.5 Search

### 14.5.1 Linear Search

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.

Algorithm Linear Search ( Array A, Value x)

Step 1: Set i to 1 Step 2: if  $i > n$  then go to step 7 Step 3: if  $A[i] = x$  then go to step 6 Step 4: Set i to  $i + 1$  Step 5: Go to Step 2 Step 6: Print Element x Found at index i and go to step 8 Step 7: Print element not found Step 8: Exit  
Pseudocode procedure *linear<sub>s</sub>earch(list, value)*

```

for each item in the list
  if match item == value
    return the item's location
  end if
end for
end procedure

```

### 14.5.2 Binary Search

Binary search is a fast search algorithm with run-time complexity of  $(\log n)$ . This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the subarray reduces to zero.

How Binary Search Works? For a binary search to work, it is mandatory for the target array to be sorted. We shall learn the process of binary search with a pictorial example. The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

First, we shall determine half of the array by using this formula

$\text{mid} = \text{low} + (\text{high} - \text{low}) / 2$  Here it is,  $0 + (9 - 0) / 2 = 4$  (integer value of 4.5). So, 4 is the mid of the array.

Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

We change our low to  $\text{mid} + 1$  and find the new mid value again.

$\text{low} = \text{mid} + 1$   $\text{mid} = \text{low} + (\text{high} - \text{low}) / 2$  Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.

Hence, we calculate the mid again. This time it is 5.

We compare the value stored at location 5 with our target value. We find that it is a match.

We conclude that the target value 31 is stored at location 5.

Binary search halves the searchable items and thus reduces the count of comparisons to be made to very less numbers.

**Pseudocode** The pseudocode of binary search algorithms should look like this

```

Procedure binary_search(A: sorted array, n: size of array, x: value to be searched)
Set lowerBound = 1 Set upperBound = n
while x not found if upperBound < lowerBound EXIT: x does not exist.
set midPoint = lowerBound + ( upperBound - lowerBound ) / 2
if A[midPoint] < x set lowerBound = midPoint + 1
if A[midPoint] > x set upperBound = midPoint - 1
if A[midPoint] = x EXIT: x found at location midPoint
end while
end procedure

```

### 14.5.3 Interpolation Search

Interpolation search is an improved variant of binary search. This search algorithm works on the probing position of the required value. For this algorithm to work properly, the data collection should be in a sorted form and equally distributed.

Binary search has a huge advantage of time complexity over linear search. Linear search has worst-case complexity of  $(n)$  whereas binary search has  $(\log n)$ .

There are cases where the location of target data may be known in advance. For example, in case of a telephone directory, if we want to search the telephone number of Morpheus. Here, linear search and even binary search will seem slow as we can directly jump to memory space where the names start from 'M' are stored.

**Positioning in Binary Search** In binary search, if the desired data is not found then the rest of the list is divided in two parts, lower and higher. The search is carried out in either of them.

Even when the data is sorted, binary search does not take advantage to probe the position of the desired data.

**Position Probing in Interpolation Search** Interpolation search finds a particular item by computing the probe position. Initially, the probe position is the position of the middle most item of the collection.

If a match occurs, then the index of the item is returned. To split the list into two parts, we use the following method

$$\text{mid} = \text{Lo} + ((\text{Hi} - \text{Lo}) / (\text{A}[\text{Hi}] - \text{A}[\text{Lo}])) * (\text{X} - \text{A}[\text{Lo}]) \text{ where}$$

A = list Lo = Lowest index of the list Hi = Highest index of the list A[n] = Value stored at index n in the list If the middle item is greater than the item, then the probe position is again calculated in the sub-array to the right of the middle item. Otherwise, the item is searched in the subarray to the left of the middle item. This process continues on the sub-array as well until the size of subarray reduces to zero.

Runtime complexity of interpolation search algorithm is  $O(\log(\log n))(\log(\log n))$  as compared to  $O(\log n)(\log n)$  of BST in favorable situations.

Algorithm As it is an improvisation of the existing BST algorithm, we are mentioning the steps to search the 'target' data value index, using position probing

Step 1 Start searching data from middle of the list. Step 2 If it is a match, return the index of the item, and exit. Step 3 If it is not a match, probe position. Step 4 Divide the list using probing formula and find the new middle. Step 5 If data is greater than middle, search in higher sub-list. Step 6 If data is smaller than middle, search in lower sub-list. Step 7 Repeat until match. Pseudocode

A Array list N Size of A X Target Value

Procedure *Interpolation<sub>search</sub>*()

Set Lo 0 Set Mid -1 Set Hi N-1

While X does not match

if Lo equals to Hi OR A[Lo] equals to A[Hi] EXIT: Failure, Target not found  
end if

Set Mid = Lo + ((Hi - Lo) / (A[Hi] - A[Lo])) \* (X - A[Lo])

if A[Mid] = X EXIT: Success, Target found at Mid else if A[Mid] < X Set Lo to Mid+1 else if A[Mid] > X Set Hi to Mid-1 end if end if

End While

End Procedure

#### 14.5.4 Hash Table

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data. Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.

Hashing Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values. Consider an example of hash table of size 20, and the following items are to be stored. Item are in the (key,value) format.

(1,20) (2,70) (42,80) (4,25) (12,44) (14,32) (17,11) (13,78) (37,98) Sr. No.  
Key Hash Array Index 1 1 1 2 2 2 3 42 42 4 4 4 5 12 12 6 14 4 7 17 7 8 13 3 9 37  
7 Linear Probing As we can see, it may happen that the hashing technique is used to create an already used index of the array. In such a case, we can search the next empty location in the array by looking into the next cell until we find an empty cell. This technique is called linear probing.

Sr. No. Key Hash Array Index After Linear Probing, Array Index 1 1 1 2 2 2 3 42 42 4 4 4 5 12 12 6 14 14 7 17 17 8 13 13 9 37 37 Basic Operations Following are the basic primary operations of a hash table.

Search Searches an element in a hash table. Insert inserts an element in a hash table. delete Deletes an element from a hash table. DataItem Define a data item having some data and key, based on which the search is to be conducted in a hash table.



struct DataItem int data; int key; ; Hash Method Define a hashing method to compute the hash code of the key of the data item.

int hashCode(int key) return key Search Operation Whenever an element is to be searched, compute the hash code of the key passed and locate the element using that hash code as index in the array. Use linear probing to get the element ahead if the element is not found at the computed hash code.

Example

```
struct DataItem *search(int key) //get the hash int hashIndex = hash-
Code(key);
//move in array until an empty while(hashArray[hashIndex] != NULL)
if(hashArray[hashIndex]->key == key) return hashArray[hashIndex];
//go to next cell ++hashIndex;
//wrap around the table hashIndex
```

return NULL; Insert Operation Whenever an element is to be inserted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing for empty location, if an element is found at the computed hash code.

Example

```
void insert(int key,int data) struct DataItem *item = (struct DataItem*)
malloc(sizeof(struct DataItem)); item->data = data; item->key = key;
//get the hash int hashIndex = hashCode(key);
//move in array until an empty or deleted cell while(hashArray[hashIndex]
!= NULL hashArray[hashIndex]->key != -1) //go to next cell ++hashIndex;
//wrap around the table hashIndex
```

hashArray[hashIndex] = item; Delete Operation Whenever an element is to be deleted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing to get the element ahead if an element is not found at the computed hash code. When found, store a dummy item there to keep the performance of the hash table intact.

Example

```
struct DataItem* delete(struct DataItem* item) int key = item->key;
//get the hash int hashIndex = hashCode(key);
//move in array until an empty while(hashArray[hashIndex] !=NULL)
if(hashArray[hashIndex]->key == key) struct DataItem* temp = hashAr-
ray[hashIndex];
//assign a dummy item at deleted position hashArray[hashIndex] = dum-
myItem; return temp;
//go to next cell ++hashIndex;
//wrap around the table hashIndex
return NULL;
```

## 14.6 Graph

### 14.6.1 Graph Data Structure

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

Formally, a graph is a pair of sets  $(V, E)$ , where  $V$  is the set of vertices and  $E$  is the set of edges, connecting the pairs of vertices. Take a look at the following graph

In the above graph,

$V = \{a, b, c, d, e\}$

$E = \{ab, ac, bd, cd, de\}$

Definitions Mathematical graphs can be represented in data structure. We can represent a graph using an array of vertices and a two-dimensional array of edges. Before we proceed further, let's familiarize ourselves with some important terms

**Vertex** Each node of the graph is represented as a vertex. In the following example, the labeled circle represents vertices. Thus, A to G are vertices. We can represent them using an array as shown in the following image. Here A can be identified by index 0. B can be identified using index 1 and so on. **Edge** Edge represents a path between two vertices or a line between two vertices. In the following example, the lines from A to B, B to C, and so on represents edges. We can use a two-dimensional array to represent an array as shown in the following image. Here AB can be represented as 1 at row 0, column 1, BC as 1 at row 1, column 2 and so on, keeping other combinations as 0. **Adjacency** Two nodes or vertices are adjacent if they are connected to each other through an edge. In the following example, B is adjacent to A, C is adjacent to B, and so on. **Path** Path represents a sequence of edges between the two vertices. In the following example, ABCD represents a path from A to D.

**Basic Operations** Following are basic primary operations of a Graph

**Add Vertex** Adds a vertex to the graph. **Add Edge** Adds an edge between the two vertices of the graph. **Display Vertex** Displays a vertex of the graph.

### 14.6.2 Depth First Traversal

Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

As in the example given above, DFS algorithm traverses from A to B to C to D first then to E, then to F and lastly to G. It employs the following rules.

**Rule 1** Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack. **Rule 2** If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.) **Rule 3** Repeat Rule 1 and Rule 2 until the stack is empty. **Algorithms Step Traversal Description** 1. Initialize the stack. 2. Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order. 3. Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both Sand D are adjacent to A but we are concerned for unvisited nodes only. 4. Visit D and mark it as visited and put onto the stack. Here, we have B and C nodes, which are adjacent to D and both are unvisited. However, we shall again choose in an alphabetical order. 5. We choose B, mark it as visited and put onto the stack. Here B does not have any unvisited adjacent node. So, we pop B from the stack. 6. We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find D to be on the top of the stack. 7. Only unvisited adjacent

node is from D is C now. So we visit C, mark it as visited and put it onto the stack.

### 14.6.3 Breadth First Traversal

Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

As in the example given above, BFS algorithm traverses from A to B to E to F first then to C and G lastly to D. It employs the following rules.

Rule 1 Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it \* in a queue. Rule 2 If no adjacent vertex is found, remove the first vertex from the queue. Rule 3 Repeat Rule 1 and Rule 2 until the queue is empty. Algorithms Step Traversal Description 1. Initialize the stack. 2. Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order. 3. Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both S and D are adjacent to A but we are concerned for unvisited nodes only. 4. Visit D and mark it as visited and put onto the stack. Here, we have B and C nodes, which are adjacent to D and both are unvisited. However, we shall again choose in an alphabetical order. 5. We choose B, mark it as visited and put onto the stack. Here B does not have any unvisited adjacent node. So, we pop B from the stack. 6. We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find D to be on the top of the stack. 7. Only unvisited adjacent node is from D is C now. So we visit C, mark it as visited and put it onto the stack. At this stage, we are left with no unmarked (unvisited) nodes. But as per the algorithm we keep on dequeuing in order to get all unvisited nodes. When the queue gets emptied, the program is over

## 14.7 String

String manipulation is a basic operation of many algorithms and utilities such as data validation, text parsing, file conversions and others. The Java APIs contain three classes that are used to work with character data:

Character – A class whose instances can hold a single character value. String – An immutable class for working with multiple characters. StringBuffer and StringBuilder – Mutable classes for working with multiple characters. The String and StringBuffer classes are two you will use the most in your programming assignments. You use the String class in situations when you want to prohibit data modification; otherwise you use the StringBuffer class.

The String class In Java Strings can be created in two different ways. Either using a new operator

```
String demo1 = new String("This is a string");
```

```
char[] demo2 = {'s','t','r','i','n','g'}; String str = new String(demo2); or using a string literal
```

```
String demo3 = "This is a string";
```

The example below demonstrates differences between these initializations

```
String s1 = new String("Fester"); String s2 = new String("Fester"); String
s3 = "Fester"; String s4 = "Fester"; Then
```

`s1 == s2` returns false `s1 == s3` returns false `s3 == s4` returns true Because of the importance strings in real life, Java stores (at compile time) all strings in a special internal table as long as you create your strings using a string literal `String s3 = "Fester"`. This process is called canonicalization - it replaces multiple string objects with a single object. This is why in the above example `s3` and `s4` refer to the same object. Also note that creating strings like `s3` and `s4` is more efficient. Review the code example `StringOptimization.java` that demonstrates time comparisons between these two ways of string creation.

Here are some important facts you must know about strings:

1. A string is not an array of characters. Therefore, to access a particular character in a string, you have to use the `charAt()` method. In this code snippet we get the fourth character which is 't':

```
String str = "on the edge of history"; char ch = str.charAt(3);
```

2. The `toString()` method is used when we need a string representation of an object.

The method is defined in the `Object` class. For most important classes that you create, you will want to override `toString()` and provide your own string representation.

3. Comparing strings content using `==` is the most common mistake beginners do. You compare the content using either `equals()` or `compareTo()` methods.

**Basic String methods** The `String` class contains an enormous amount of useful methods for string manipulation. The following table presents the most common `String` methods:

`str.charAt(k)` returns a char at position `k` in `str`. `str.substring(k)` returns a substring from index `k` to the end of `str`. `s.substring(k, n)` returns a substring from index `k` to index `n-1` of `str`. `str.indexOf(s)` returns an index of the first occurrence of `String s` in `str`. `str.indexOf(s, k)` returns an index of `String s` starting an index `k` in `str`. `str.startsWith(s)` returns true if `str` starts with `s`. `str.startsWith(s, k)` returns true if `str` starts with `s` at index `k`. `str.equals(s)` returns true if the two strings have equal values. `str.equalsIgnoreCase(s)` same as above ignoring case. `str.compareTo(s)` compares two strings. `s.compareToIgnoreCase(t)` same as above ignoring case. Examine the code in `BasicStringDemo.java` for further details.

**The `StringBuffer` class** In many cases when you deal with strings you will use methods available in the companion `StringBuffer` class. This mutable class is used when you want to modify the contents of the string. It provides an efficient approach to dealing with strings, especially for large dynamic string data. `StringBuffer` is similar to `ArrayList` in a way that the memory allocated to an object is automatically expanded to take up additional data.

Here is an example of reversing a string using string concatenation

```
public static String reverse1(String s) String str = "";
for(int i = s.length() - 1; i >= 0; i--) str += s.charAt(i);
return str; and using a StringBuffer's append
public static String revers2(String s) StringBuffer sb = new StringBuffer();
for(int i = s.length() - 1; i >= 0; i--) sb.append(s.charAt(i));
return sb.toString();
```

Another way to reverse a string is to convert a `String` object into a `StringBuffer` object, use the `reverse` method, and then convert it back to a string:

public static String reverse3(String s) return new StringBuffer(s).reverse().toString();  
The performance difference between these two classes is that StringBuffer is faster than String when performing concatenations. Each time a concatenation occurs, a new string is created, causing excessive system resource consumption.

Review the code example StringOverhead.java that demonstrates time comparisons of concatenation on Strings and StringBuffer.

**StringTokenizer** This class (from java.util package) allows you to break a string into tokens (substrings). Each token is a group of characters that are separated by delimiters, such as an empty space, a semicolon, and so on. So, a token is a maximal sequence of consecutive characters that are not delimiters. Here is an example of the use of the tokenizer (an empty space is a default delimiter):

```
String s = "Nothing is as easy as it looks"; StringTokenizer st = new StringTokenizer(s); while (st.hasMoreTokens()) { String token = st.nextToken(); System.out.println("Token [" + token + "]"); } Here, hasMoreTokens() method checks if there are more tokens available from the string, and nextToken() method returns the next token from the string tokenizer.
```

The set of delimiters (the characters that separate tokens) may be specified in the second argument of StringTokenizer. In the following example, StringTokenizer has a set of two delimiters: an empty space and an underscore:

```
String s = "Every solution breeds new problems"; StringTokenizer st = new StringTokenizer(s, " "); while (st.hasMoreTokens()) { String token = st.nextToken(); System.out.println("Token [" + token + "]"); } Here, the second argument of StringTokenizer is a string containing the delimiters.
```

**Character Classes**

[abc] a, b, or c (simple class) [^abc] Any character except a, b, or c (negation) [a-zA-Z] a through z, or A through Z, inclusive (range) [a-d[m-p]] a through d, or m through p : [a-dm-p] (union) [a-z[def]] d, e, or f (intersection) [a-z[^bc]] a through z, except for b and c : [ad-z] (subtraction) [a-z[m-p]] a through z, and not m through p : [a-lq-z] (subtraction) d any digit from 0 to 9

w any word character (a-z, A-Z, 0-9 and \_)

W any non-word character

s any whitespace character? appearing once or not at all \* appearing zero or more times +

appearing one or more times The Java String class has several methods that allow you to perform an operation using a regular expression.

**The matches() method** The matches("regex") method returns true or false depending whether the string can be matched entirely by the regular expression "regex". For example,

```
"abc".matches("abc") returns True, but
```

```
"abc".matches("bc") returns False. In the following code examples we match
```

all strings that start with any number of dots (denoted by \*), followed by "abc" and end with one or more underscores (denoted by +).

```
String regex = ".*"+"abc"+"+";
```

```
"..abc".matches(regex);
```

```
"abc".matches(regex);
```

"abc".matches(regex); There replaceAll() method The method replaceAll("regex", "replacement") replaces all occurrences of the regular expression "regex" in the string "replacement" with the string "replacement".

**String replaceAll()** String str = "Nothing is as easy as it looks!"; str = str.replaceAll("[a-zA-Z]", ""); The pattern "[a-zA-Z]" describes all letters (in upper and lower cases). Next we negate this pattern, to get all non-letters "[^a-zA-Z]".

In the next example, we replace a sequence of characters by "-".

```
String str = "aabfoaaaabfoabfoob"; str = str.replaceAll("a*b", "-"); The
```

star "\*" in the pattern "ab" denotes that character "a" may be repeated zero or more times. The output: "-foo-foo-foo-";

The `split()` method The `split("regex")` splits the string at each "regex" match and returns an array of strings where each element is a part of the original string between two "regex" matches.

In the following example we break a sentence into words, using an empty space as a delimiter:

`String s = "Nothing is as easy as it looks"; String[] st = s.split(" ");` Tokens are stored in an array of strings and could be easily accessible using array indexes. In the next code example, we choose two delimiters: either an empty space or an underscore:

`String s = "Everysolutionbreedsnewproblems"; String[] st = s.split("_| ");` What if a string contains several tokens, that denotes a repetitive pattern

`String s = "Everysolutionbreedsnewpproblems"; String[] st = s.split("_+ ");` It's important to observe that `split()` might return empty tokens.

`String[] st = "Tomorrow".split("r");` we have three tokens, where the second token is empty string. That is so because `split()` returns tokens between two "regex" matches.

One of the widely use of `split()` is to break a given text file into words. This could be easily done by means of the metacharacter "\*" (any non-word character), which allows you to perform a "whole words only" search using a regular expression. A "word character" is either an alphabet character (a-z and A-Z) or a digit (0-9) or a underscore.

`"Let's go, Steelers!!!"`.`split("W");` returns the following array of tokens

[Let, s, go, Steelers] Examine the code in `Split.java` for further details.

Pattern matching Pattern matching in Java is based on use of two classes

Pattern - compiled representation of a regular expression. Matcher - an engine that performs match operations. A typical invocation is the following, first we create a pattern

`String seq = "CCCAA"; Pattern p = Pattern.compile("C*A*");` In this example we match all substrings that start with any number of Cs followed by any number of As. Then we create a `Matcher` object that can match any string against our pattern

`Matcher m = p.matcher(seq);` Finally, we do actual matching

`boolean res = m.matches();` The `Matcher` class has another widely used method, called `find()`, that finds next substring that matches a given pattern. In the following example we count the number of matches "ACC"

`String seq = "CGTATCCCACAGCACCACACCCAACAACCCA"; Pattern p = Pattern.compile("A1C2");` `Matcher m = p.matcher(seq); int count = 0;` `while( m.find() ) count++;` `System.out.println("there are " + count + " ACC");` Examine the code example `Matching.java` for further details.

Pattern matching in Computational Biology

The DNA (the genetic blueprint) of any species is composed of about 4 billion ACGT nucleotides. DNA forms a double helix that has two strands of DNA binding and twisting together. In pattern matching problems we ignore the fact that DNA forms a double helix, and think of it only as a single strand. The other strand is complimentary. Knowing one strand allows uniquely determine the other one. Thus, DNA is essentially a linear molecule that looks like a string composed out of only four characters A, C, G, and T:

CGTATCCCACAGCACCACACCCAACAACCC Each nucleotides (also called a base) strongly binds to no more than two other bases. These links provides a linear model of DNA strand. The particular order of ACGT nucleotides is extremely important. Different orders generate humans, animals, corn, and other organisms. The size of the genome (a genome is all the DNA in an organism) does not necessarily correlate with the complexity of the organism it belongs to. Humans have less than a third as many genes as were expected.

Pattern matching in computational biology arises from the need to know characteristics of DNA sequences, such as

- find the best way to align two sequences.
- find any common subsequences
- determine how well a sequence fits into a given model.

Comparing various DNA sequencesn provide many uses. Current scientific theories suggest that very similar DNA sequences have a common ancestor. The more similar two sequences are, the more recently they evolved from a single ancestor. With such knowledge, for example, we can reconstruct a phylogenetic tree (known as a "tree of life".) that shows how long ago various organisms diverged and which species are closely related.

Challenges Strings: Making Anagrams References "Strings". Victor S.Adamchik, CMU. 2009

### 14.7.1 Tries

Introduction There are many algorithms and data structures to index and search strings inside a text, some of them are included in the standard libraries, but not all of them; the trie data structure is a good example of one that isn't.

Let word be a single string and let dictionary be a large set of words. If we have a dictionary, and we need to know if a single word is inside of the dictionary the tries are a data structure that can help us. But you may be asking yourself, "Why use tries if set and hash tables can do the same?"

There are two main reasons:

The tries can insert and find strings in  $O(L)O(L)$  time (where  $L$  represent the length of a single word). This is much faster than set , but is it a bit faster than a hash table. The set and the hash tables can only find in a dictionary words that match exactly with the single word that we are finding; the trie allow us to find words that have a single character different, a prefix in common, a character missing, etc.

The tries can be useful in TopCoder problems, but also have a great amount of applications in software engineering. For example, consider a web browser. Do you know how the web browser can auto complete your text or show you many possibilities of the text that you could be writing? Yes, with the trie you can do it very fast. Do you know how an orthographic corrector can check that every word that you type is in a dictionary? Again a trie. You can also use a trie for suggested corrections of the words that are present in the text but not in the dictionary.

What is a Tree? You may read about how wonderful the tries are, but maybe you don't know yet what the tries are and why the tries have this name. The word trie is an infix of the word "retrieval" because the trie can find a single word in a dictionary with only a prefix of the word. The main idea of the trie data structure consists of the following parts:

The trie is a tree where each vertex represents a single word or a prefix. The root represents an empty string (""), the vertexes that are direct sons of the root represent prefixes of length 1, the vertexes that are 2 edges of distance from the root represent prefixes of length 2, the vertexes that are 3 edges of distance from the root represent prefixes of length 3 and so on. In other words, a vertex that are  $k$  edges of distance of the root have an associated prefix of length  $k$ . Let  $v$  and  $w$  be two vertexes of the trie, and assume that  $v$  is a direct father of  $w$ , then  $v$  must have an associated prefix of  $w$ . The next figure shows a trie with the words "tree", "trie", "algo", "assoc", "all", and "also."

Note that every vertex of the tree does not store entire prefixes or entire words. The idea is that the program should remember the word that represents each vertex while lower in the tree.

**Coding a Trie** The tries can be implemented in many ways, some of them can be used to find a set of words in the dictionary where every word can be a little different than the target word, and other implementations of the tries can provide us with only words that match exactly with the target word. The implementation of the trie that will be exposed here will consist only of finding words that match exactly and counting the words that have some prefix. This implementation will be pseudo code because different coders can use different programming languages.

We will code these 4 functions:

**addWord.** This function will add a single string word to the dictionary. **countPrefixes.** This function will count the number of words in the dictionary that have a string prefix as prefix. **countWords.** This function will count the number of words in the dictionary that match exactly with a given string word. Our trie will only support letters of the English alphabet. We need to also code a structure with some fields that indicate the values stored in each vertex. As we want to know the number of words that match with a given string, every vertex should have a field to indicate that this vertex represents a complete word or only a prefix (for simplicity, a complete word is considered also a prefix) and how many words in the dictionary are represented by that prefix (there can be repeated words in the dictionary). This task can be done with only one integer field words.

Because we want to know the number of words that have like prefix a given string, we need another integer field prefixes that indicates how many words have the prefix of the vertex. Also, each vertex must have references to all his possible sons (26 references). Knowing all these details, our structure should have the following members:

structure Trie integer words; integer prefixes; reference edges[26]; And we also need the following functions:

initialize(vertex) addWord(vertex, word); integer countPrefixes(vertex, prefix); integer countWords(vertex, word); First of all, we have to initialize the vertexes with the following function:

initialize(vertex) vertex.words=0 vertex.prefixes=0 for  $i=0$  to 26 edges[i]=NoEdge

The addWord function consists of two parameters, the vertex of the insertion and the word that will be added. The idea is that when a string word is added to a vertex vertex, we will add word to the corresponding branch of vertex cutting the leftmost character of word. If the needed branch does not exist, we will have to create it. All the TopCoder languages can simulate the process of cutting a character in constant time instead of creating a copy of the original string or



moving the other characters.

```
addWord(vertex, word) if isEmpty(word) vertex.words=vertex.words+1 else
vertex.prefixes=vertex.prefixes+1 k=firstCharacter(word) if(notExists(edges[k]))
edges[k]=createEdge() initialize(edges[k]) cutLeftmostCharacter(word) addWord(edges[k],
word) The functions countWords and countPrefixes are very similar. If we are
finding an empty string we only have to return the number of words/prefixes
associated with the vertex. If we are finding a non-empty string, we should to
find in the corresponding branch of the tree, but if the branch does not exist,
we have to return 0.
```

```
countWords(vertex, word) k=firstCharacter(word) if isEmpty(word) return
vertex.words else if notExists(edges[k]) return 0 else cutLeftmostCharacter(word)
return countWords(edges[k], word);
```

```
countPrefixes(vertex, prefix) k=firstCharacter(prefix) if isEmpty(word) re-
turn vertex.prefixes else if notExists(edges[k]) return 0 else cutLeftmostChar-
acter(prefix) return countWords(edges[k], prefix) Complexity Analysis In the
introduction you may read that the complexity of finding and inserting a trie is
linear, but we have not done the analysis yet. In the insertion and finding notice
that lowering a single level in the tree is done in constant time, and every time
that the program lowers a single level in the tree, a single character is cut from
the string; we can conclude that every function lowers L levels on the tree and
every time that the function lowers a level on the tree, it is done in constant
time, then the insertion and finding of a word in a trie can be done in O(L)
time. The memory used in the tries depends on the methods to store the edges
and how many words have prefixes in common.
```

Other Kinds of Tries We used the tries to store words with lowercase letters, but the tries can be used to store many other things. We can use bits or bytes instead of lowercase letters and every data type can be stored in the tree, not only strings. Let flow your imagination using tries! For example, suppose that you want to find a word in a dictionary but a single letter was deleted from the word. You can modify the countWords function:

```
countWords(vertex, word, missingLetters) k=firstCharacter(word) if isEmpty(word)
return vertex.word else if notExists(edges[k]) and missingLetters=0 return 0 else
if notExists(edges[k]) cutLeftmostCharacter(word) return countWords(vertex,
word, missingLetters-1) Here we cut a character but we don't go lower in the
tree else We are adding the two possibilities: the first character has been deleted
plus the first character is present r=countWords(vertex, word, missingLetters-1)
cutLeftmostCharacter(word) r=r+countWords(edges[k], word, missingLetters)
return r The complexity of this function may be larger than the original, but it
is faster than checking all the subsets of characters of a word.
```

Challenges "Tries: Contacts". hackerrank. 2016 References "Using Tries – Topcoder". Topcoder.com. N.p., 2016. Web. 11 Oct. 2016.

### 14.7.2 Suffix Array and suffix tree

A suffix tree T is a natural improvement over trie used in pattern matching problem, the one defined over a set of substrings of a string s. The idea is very simple here. Such a trie can have a long paths without branches. If we only can reduce these long paths into one jump, we will reduce the size of the trie significantly, so this is a great first step in improving the complexity of

operations on such a tree. This reduced trie defined over a subset of suffixes of a string  $s$  is called a suffix tree of  $s$

For better understanding, let's consider the suffix tree  $T$  for a string  $s = \text{abakan}$ . A word  $\text{abakan}$  has 6 suffixes  $\text{abakan}$ ,  $\text{bakan}$ ,  $\text{akan}$ ,  $\text{kan}$ ,  $\text{an}$ ,  $\text{n}$  and its suffix tree looks like this:

There is a famous algorithm by Ukkonen for building suffix tree for  $s$  in linear time in terms of the length of  $s$ . However, because it may look quite complicated at first sight, many people are discouraged to learn how it works. Fortunately, there is a great, I mean an excellent, description of Ukkonen's algorithm given on StackOverflow. Please refer to it for better understanding what a suffix tree is and how to build it in linear time.

Suffix trees can solve many complicated problems, because it contain so many information about the string itself. For example, in order to know how many times a pattern  $P$  occurs in  $s$ , it is sufficient to find  $P$  in  $T$  and return the size of a subtree corresponding to its node. Another well known application is finding the number of distinct substrings of  $s$ , and it can be solved easily with suffix tree, while the problem looks very complicated at first sight.

The post I linked from StackOverflow is so great, that you simply must read it. After that, you will be able to identify problems solvable with suffix trees easily.

If you want to know more about when to use a suffix tree, you should read this paper about the applications of suffix trees.

Suffix Array Suffix array is a very nice array based structure. Basically, it is a lexicographically sorted array of suffixes of a string  $s$ . For example, let's consider a string  $s = \text{abakan}$ . A word  $\text{abakan}$  has 6 suffixes  $\text{abakan}$ ,  $\text{bakan}$ ,  $\text{akan}$ ,  $\text{kan}$ ,  $\text{an}$ ,  $\text{n}$  and its suffix tree looks like this:

Of course, in order to reduce space, we do not store the exact suffixes. It is sufficient to store their indices.

Suffix arrays, especially combined with LCP table (which stands for Longest Common Prefix of neighboring suffixes table), are very very useful for solving many problems. I recommend reading this nice programming oriented paper about suffix arrays, their applications and related problems by Stanford University.

Suffix arrays can be built easily in  $O(n \log^2 n)$  time, where  $n$  is the length of  $s$ , using the algorithm proposed in the paper from the previous paragraph. This time can be improved to  $O(n \log n)$  using linear time sorting algorithm.

However, there is so extraordinary, cool and simple linear time algorithm for building suffix arrays by Kärkkäinen and Sanders, that reading it is a pure pleasure and you cannot miss it.

Correspondence between suffix tree and suffix array

It is also worth to mention, that a suffix array can be constructed directly from a suffix tree in linear time using DFS traversal. Suffix tree can be also constructed from the suffix array and LCP table as described here.

### 14.7.3 Knuth-Morris-Pratt Algorithm

The problem:

given a (short) pattern and a (long) text, both strings, determine whether the pattern appears somewhere in the text.

We'll go through the Knuth-Morris-Pratt (KMP) algorithm, which can be thought of as an efficient way to build these automata. I also have some working C++ source code which might help you understand the algorithm better.

First let's look at a naive solution.

suppose the text is in an array: `char T[n]` and the pattern is in another array: `char P[m]`. One simple method is just to try each possible position the pattern could appear in the text.

Naive string matching:

```
for (i=0; T[i] != "; i++) for (j=0; T[i+j] != " P[j] != " T[i+j]==P[j]; j++)
; if (P[j] == ") found a match
```

There are two nested loops; the inner one takes  $O(m)$  iterations and the outer one takes  $O(n)$  iterations so the total time is the product,  $O(mn)$ . This is slow; we'd like to speed it up.

In practice this works pretty well – not usually as bad as this  $O(mn)$  worst case analysis. This is because the inner loop usually finds a mismatch quickly and move on to the next position without going through all  $m$  steps. But this method still can take  $O(mn)$  for some inputs. In one bad example, all characters in `T[]` are "a"s, and `P[]` is all "a"'s except for one "b" at the end. Then it takes  $m$  comparisons each time to discover that you don't have a match, so  $mn$  overall.

Here's a more typical example. Each row represents an iteration of the outer loop, with each character in the row representing the result of a comparison (X if the comparison was unequal). Suppose we're looking for pattern "nano" in text "banananobano".

```
0 1 2 3 4 5 6 7 8 9 10 11 T: b a n a n a n o b a n o
i=0: X i=1: X i=2: n a n X i=3: X i=4: n a n o i=5: X i=6: n X i=7: X i=8:
X i=9: n X i=10: X
```

Some of these comparisons are wasted work! For instance, after iteration  $i=2$ , we know from the comparisons we've done that `T[3]="a"`, so there is no point comparing it to "n" in iteration  $i=3$ . And we also know that `T[4]="n"`, so there is no point making the same comparison in iteration  $i=4$ .

**Skipping outer iterations** The Knuth-Morris-Pratt idea is, in this sort of situation, after you've invested a lot of work making comparisons in the inner loop of the code, you know a lot about what's in the text. Specifically, if you've found a partial match of  $j$  characters starting at position  $i$ , you know what's in positions `T[i]...T[i+j-1]`. You can use this knowledge to save work in two ways. First, you can skip some iterations for which no match is possible. Try overlapping the partial match you've found with the new match you want to find:

```
i=2: n a n i=3: n a n o
```

Here the two placements of the pattern conflict with each other – we know from the  $i=2$  iteration that `T[3]` and `T[4]` are "a" and "n", so they can't be the "n" and "a" that the  $i=3$  iteration is looking for. We can keep skipping positions until we find one that doesn't conflict:

```
i=2: n a n i=4: n a n o
```

Here the two "n"'s coincide. Define the overlap of two strings  $x$  and  $y$  to be the longest word that's a suffix of  $x$  and a prefix of  $y$ . Here the overlap of "nan" and "nano" is just "n". (We don't allow the overlap to be all of  $x$  or  $y$ , so it's not "nan"). In general the value of  $i$  we want to skip to is the one corresponding to the largest overlap with the current partial match:

String matching with skipped iterations:

```
i=0; while (i<n) for (j=0; T[i+j] != " P[j] != " T[i+j]==P[j]; j++) ; if (P[j]
==") found a match; i = i + max(1, j-overlap(P[0..j-1],P[0..m]));
```

Skipping inner iterations The other optimization that can be done is to skip some iterations in

the inner loop. Let's look at the same example, in which we skipped from  $i=2$  to  $i=4$ :

$i=2$ : n a n  $i=4$ : n a n o In this example, the "n" that overlaps has already been tested by the  $i=2$  iteration. There's no need to test it again in the  $i=4$  iteration. In general, if we have a nontrivial overlap with the last partial match, we can avoid testing a number of characters equal to the length of the overlap. This change produces (a version of) the KMP algorithm:

KMP, version 1:

```
i=0; o=0; while (i<n) for (j=o; T[i+j] != P[j]; j++)
; if (P[j] == T[i+j]) found a match; o = overlap(P[0..j-1], P[0..m]); i = i + max(1, j-o);
```

The only remaining detail is how to compute the overlap function. This is a function only of  $j$ , and not of the characters in  $T[]$ , so we can compute it once in a preprocessing stage before we get to this part of the algorithm. First let's see how fast this algorithm is.

**KMP time analysis** We still have an outer loop and an inner loop, so it looks like the time might still be  $O(mn)$ . But we can count it a different way to see that it's actually always less than that. The idea is that every time through the inner loop, we do one comparison  $T[i+j] == P[j]$ . We can count the total time of the algorithm by counting how many comparisons we perform. We split the comparisons into two groups: those that return true, and those that return false. If a comparison returns true, we've determined the value of  $T[i+j]$ . Then in future iterations, as long as there is a nontrivial overlap involving  $T[i+j]$ , we'll skip past that overlap and not make a comparison with that position again. So each position of  $T[]$  is only involved in one true comparison, and there can be  $n$  such comparisons total. On the other hand, there is at most one false comparison per iteration of the outer loop, so there can also only be  $n$  of those. As a result we see that this part of the KMP algorithm makes at most  $2n$  comparisons and takes time  $O(n)$ .

**KMP and finite automata** If we look just at what happens to  $j$  during the algorithm above, it's sort of like a finite automaton. At each step  $j$  is set either to  $j+1$  (in the inner loop, after a match) or to the overlap  $o$  (after a mismatch). At each step the value of  $o$  is just a function of  $j$  and doesn't depend on other information like the characters in  $T[]$ . So we can draw something like an automaton, with arrows connecting values of  $j$  and labeled with matches and mismatches.

The difference between this and the automata we are used to is that it has only two arrows out of each circle, instead of one per character. But we can still simulate it just like any other automaton, by placing a marker on the start state ( $j=0$ ) and moving it around the arrows. Whenever we get a matching character in  $T[]$  we move on to the next character of the text. But whenever we get a mismatch we look at the same character in the next step, except for the case of a mismatch in the state  $j=0$ .

So in this example (the same as the one above) the automaton goes through the sequence of states:

```
j=0 mismatch T[0] != "n" j=0 mismatch T[1] != "n" j=0 match T[2] == "n"
j=1 match T[3] == "a" j=2 match T[4] == "n" j=3 mismatch T[5] != "o" j=1
match T[5] == "a" j=2 match T[6] == "n" j=3 match T[7] == "o" j=4 found
match j=0 mismatch T[8] != "n" j=0 mismatch T[9] != "n" j=0 match T[10] ==
"n" j=1 mismatch T[11] != "a" j=0 mismatch T[11] != "n"
```

This is essentially the same sequence of comparisons done by the KMP pseudocode above. So this automaton provides an equivalent definition of the KMP algorithm. As one

student pointed out in lecture, the one transition in this automaton that may not be clear is the one from  $j=4$  to  $j=0$ . In general, there should be a transition from  $j=m$  to some smaller value of  $j$ , which should happen on any character (there are no more matches to test before making this transition). If we want to find all occurrences of the pattern, we should be able to find an occurrence even if it overlaps another one. So for instance if the pattern were "nana", we should find both occurrences of it in the text "nanana". So the transition from  $j=m$  should go to the next longest position that can match, which is simply  $j=\text{overlap}(\text{pattern}, \text{pattern})$ . In this case  $\text{overlap}(\text{"nana"}, \text{"nana"})$  is empty (all suffixes of "nana" use the letter "a", and no prefix does) so we go to  $j=0$ .

Alternate version of KMP The automaton above can be translated back into pseudo-code, looking a little different from the pseudo-code we saw before but performing the same comparisons.

KMP, version 2:

```
j = 0; for (i = 0; i < n; i++) for (;;) // loop until break if (T[i] == P[j])
// matches? j++; // yes, move on to next state if (j == m) // maybe that
// was the last state found a match; j = overlap[j]; break; else if (j == 0) break;
// no match in state j=0, give up else j = overlap[j]; // try shorter partial
match
```

The code inside each iteration of the outer loop is essentially the same as the function `match` from the C++ implementation I've made available. One advantage of this version of the code is that it tests characters one by one, rather than performing random access in the `T[]` array, so (as in the implementation) it can be made to work for stream-based input rather than having to read the whole text into memory first. The `overlap[j]` array stores the values of `overlap(pattern[0..j-1], pattern)`, which we still need to show how to compute.

Since this algorithm performs the same comparisons as the other version of KMP, it takes the same amount of time,  $O(n)$ . One way of proving this bound directly is to note, first, that there is one true comparison (in which  $T[i] == P[j]$ ) per iteration of the outer loop, since we break out of the inner loop when this happens. So there are  $n$  of these total. Each of these comparisons results in increasing  $j$  by one. Each iteration of the inner loop in which we don't break out of the loop results in executing the statement  $j = \text{overlap}[j]$ , which decreases  $j$ . Since  $j$  can only decrease as many times as it's increased, the total number of times this happens is also  $O(n)$ .

Computing the overlap function Recall that we defined the overlap of two strings  $x$  and  $y$  to be the longest word that's a suffix of  $x$  and a prefix of  $y$ . The missing component of the KMP algorithm is a computation of this overlap function: we need to know `overlap(P[0..j-1], P)` for each value of  $j > 0$ . Once we've computed these values we can store them in an array and look them up when we need them. To compute these overlap functions, we need to know for strings  $x$  and  $y$  not just the longest word that's a suffix of  $x$  and a prefix of  $y$ , but all such words. The key fact to notice here is that if  $w$  is a suffix of  $x$  and a prefix of  $y$ , and it's not the longest such word, then it's also a suffix of `overlap(x, y)`. (This follows simply from the fact that it's a suffix of  $x$  that is shorter than `overlap(x, y)` itself.) So we can list all words that are suffixes of  $x$  and prefixes of  $y$  by the following loop:

```
while (x != empty) x = overlap(x, y); output x;
```

Now let's make another definition: say that `shorten(x)` is the prefix of  $x$  with one fewer character. The next simple observation to make is that `shorten(overlap(x, y))` is still a prefix of  $y$ , but is also a suffix of `shorten(x)`. So we can find `overlap(x, y)` by adding one

more character to some word that's a suffix of `shorten(x)` and a prefix of `y`. We can just find all such words using the loop above, and return the first one for which adding one more character produces a valid overlap:

Overlap computation:

`z = overlap(shorten(x),y)` while (last char of `x` != `y[length(z)]`) if (`z = empty`) return `overlap(x,y) = empty` else `z = overlap(z,y)` return `overlap(x,y) = z` So this gives us a recursive algorithm for computing the overlap function in general. If we apply this algorithm for `x`=some prefix of the pattern, and `y`=the pattern itself, we see that all recursive calls have similar arguments. So if we store each value as we compute it, we can look it up instead of computing it again. (This simple idea of storing results instead of recomputing them is known as dynamic programming; we discussed it somewhat in the first lecture and will see it in more detail next time.) So replacing `x` by `P[0..j-1]` and `y` by `P[0..m-1]` in the pseudocode above and replacing recursive calls by lookups of previously computed values gives us a routine for the problem we're trying to solve, of computing these particular overlap values. The following pseudocode is taken (with some names changed) from the initialization code of the C++ implementation I've made available. The value in `overlap[0]` is just a flag to make the rest of the loop simpler. The code inside the for loop is the part that computes each overlap value.

KMP overlap computation:

`overlap[0] = -1; for (int i = 0; pattern[i] != "; i++) overlap[i + 1] = overlap[i] + 1; while (overlap[i + 1] > 0 pattern[i] != pattern[overlap[i + 1] - 1]) overlap[i + 1] = overlap[overlap[i + 1] - 1] + 1; return overlap;` Let's finish by analyzing the time taken by this part of the KMP algorithm. The outer loop executes `m` times. Each iteration of the inner loop decreases the value of the formula `overlap[i+1]`, and this formula's value only increases by one when we move from one iteration of the outer loop to the next. Since the number of decreases is at most the number of increases, the inner loop also has at most `m` iterations, and the total time for the algorithm is  $O(m)$ . The entire KMP algorithm consists of this overlap computation followed by the main part of the algorithm in which we scan the text (using the overlap values to speed up the scan). The first part takes  $O(m)$  and the second part takes  $O(n)$  time, so the total time is  $O(m+n)$ .

## Chương 15

# Object Oriented Programming

View online [http://magizbox.com/training/object\\_oriented\\_programming/site/](http://magizbox.com/training/object_oriented_programming/site/)

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another. There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes, which typically also determine their type.

Many of the most widely used programming languages (such as C++, Java, Python etc.) are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include Java, C++, C, Python, PHP, Ruby, Perl, Delphi, Objective-C, Swift, Scala, Common Lisp, and Smalltalk.

### 15.1 OOP

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OO programming, computer programs are designed by making them out of objects that interact with one another.[1][2] There is significant diversity in object-oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type. 1. A First Look Procedural vs Object Oriented 1

Procedural Approach

Focus is on procedures All data is shared: no protection More difficult to modify Hard to manage complexity Advantages of Object Orientation

People think in terms of object OO models map to reality OO models are: Easy to develop Easy to understand. 2. Principles encapsulation, inheritance, abstraction, polymorphism 2

Fundamental Principles of OOP In order for a programming language to be object-oriented, it has to enable working with classes and objects as well as the implementation and use of the fundamental object-oriented principles and concepts: inheritance, abstraction, encapsulation and polymorphism.

### 2.1 Encapsulation 3 4 5

Encapsulation is the packing of data and functions into a single component. The features of encapsulation are supported using classes in most object-oriented programming languages, although other alternatives also exist. It allows selective hiding of properties and methods in an object by building an impenetrable wall to protect the code from accidental corruption.

What it do? We will learn to hide unnecessary details in our classes and provide a clear and simple interface for working with them.

Example: A popular example you'll hear for encapsulation is driving a car. Do you need to know exactly how every aspect of a car works (engine, carburettor, alternator, and so on)? No - you need to know how to use the steering wheel, brakes, accelerator, and so on.

### 2.2 Inheritance 6 7

Inheritance is when an object or class is based on another object (prototypal inheritance) or class (class-based inheritance), using the same implementation (inheriting from an object or class) specifying implementation to maintain the same behavior (realizing an interface; inheriting behavior).

inherit everything, add data or functionality, override functions, super

What it do? We will explain how class hierarchies improve code readability and enable the reuse of functionality.

Example: A real-world example of inheritance is genetic inheritance. We all receive genes from both our parents that then define who we are. We share qualities of both our parents, and yet at the same time are different from them.

Example: we might classify different kinds of vehicles according to the inheritance hierarchy. Moving down the hierarchy, each kind of vehicle is both more specialized than its parent (and all of its ancestors) and more general than its children (and all of its descendants). A wheeled vehicle inherits properties common to all vehicles (it holds one or more people and carries them from place to place) but has an additional property that makes it more specialized (it has wheels). A car inherits properties common to all wheeled vehicles, but has additional, more specialized properties (four wheels, an engine, a body, and so forth). The inheritance relationship can be viewed as an is-a relationship. In this relationship, the objects become more specialized the lower in the hierarchy you go.

Look at the image above you will get a point.8 Yes, the derived class can access base class properties and still the derived class has its own properties.

### 2.3 Abstraction

In computer science, abstraction is a technique for managing complexity of computer systems. It works by establishing a level of complexity on which a person interacts with the system, suppressing the more complex details below the current level. The programmer works with an idealized interface (usually well



defined) and can add additional levels of functionality that would otherwise be too complex to handle.

What it do? We will learn how to work through abstractions: to deal with objects considering their important characteristics and ignore all other details.

Example: You'll never buy a "device", but always buy something more specific : iPhone, Samsung Galaxy, Nokia 3310... Here, iPhone, Samsung Galaxy and Nokia 3310 are concrete things, device is abstract.

#### 2.4 Polymorphism 9

Polymorphism is the provision of a single interface to entities of different types. A polymorphic type is one whose operations can also be applied to values of some other type, or types.

What it do? We will explain how to work in the same manner with different objects, which define a specific implementation of some abstract behavior.

Example: All animal can speak, but dogs woof, cats meow, and ducks quack  
There are two types of polymorphism

Overloading (compile time polymorphism): methods have the same name but different parameters. Overriding (run time polymorphism): the implementation given in base class is replaced with that in sub class.

Example 10: Let us Consider Car example for discussing the polymorphism. Take any brand like Ford, Honda, Toyota, BMW, Benz etc., Everything is of type Car. But each have their own advanced features and more advanced technology involved in their move behavior.

### 3. Concepts Learn Object Oriented Programming though Mario Game

[embed]<https://www.youtube.com/watch?v=HBbzYKMfx5Y>[/embed]

How Mario get 1up

#### 3.1. Object 11

Objects are key to understanding object-oriented technology. Look around right now and you'll find many examples of real-world objects: your dog, your desk, your television set, your bicycle. In mario world, Mario is an object.

Goomba is an object. Koopa is also an object. Even a coin and a pile are objects

Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. An object stores its state in fields (variables in some programming languages) and exposes its behavior through methods (functions in some programming languages). Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication. Hiding internal state and requiring all interaction to be performed through an object's methods is known as data encapsulation — a fundamental principle of object-oriented programming. In Mario world, Mario has some fields like position (which indicate where Mario stands), state (which indicate whether Mario alive), and some methods like walk , fire or jump.

Goomba has some fields like position (which indicate where Goomba stands), state (which indicate whether Goomba die), and direction (which indicate the direction Goomba moves). Goomba has move method, and jumped<sub>o</sub>nmethod(*which occurs when it is jumped on by*

Mario Objects, real scene

#### 3.2 Class 12

In the real world, you'll often find many individual objects all of the same kind. There may be thousands of other bicycles in existence, all of the same make and model. Each bicycle was built from the same set of blueprints and therefore contains the same components. In object-oriented terms, we say that

your bicycle is an instance of the class of objects known as bicycles. A class is the blueprint from which individual objects are created.

In Mario world, each coin object come from Coin class, and every Koomba come from Koomba class

### 3.3. Inheritance 13

Inheritance is a mechanism in OOP to design two or more entities that are different but share many common features.

Feature common to all classes are defined in the superclass The classes that inherit common features from the superclass are called subclasses In Mario World, Goomba and Koopa is in

AND MANY, MANY MORE

### 3.4. Association, Aggregation and Composition 13

Association:

Whenever two objects are related with each other the relationship is called association between object

Aggregation:

Aggregation is specialized form of association. In aggregation objects have their own life-cycle but there is ownership and child object can not belongs to another parent object. But this is only an ownership not the life-cycle control of child control object.

Example: Student and Teacher, Person and address

Composition

Composition is again specialize form of aggregation and we can call this as 'life and death' relationship. It is a strong type of aggregation. Child object does not have their life-cycle and if parent object is deleted, all child object will also be deleted.

Example: House and room

### 3.5 Polymorphism 13

Polymorphism indicates the meaning of "many forms"

Polymorphism present a method that can have many definitions. Polymorphism is related to "over loading" and "over ridding".

Overloading indicates a method can have different definitions by defining different type of parameters.

```
[code] getPrice(): void getPrice(string name): void [/code]
```

### 3.6 Abstraction 13

Abstraction is the process of modelling only relevant features

Hide unnecessary details which are irrelevant for current purpose. Reduces complexity and aids understanding.

Abstraction provides the freedom to defer implementation decisions by avoiding commitments to details.

### 3.7 Interface 13

An interface is a contract consisting of group of related function prototypes whose usage is defined but whose implementation is not:

An interface definition specifies the interface's member functions, called methods, their return types, the number and types of parameters and what they must do.

These is no implementation associated with an interface.

## 4. Coupling and Cohesion 13

4.1 Coupling Coupling defines how dependent one object on another object (that is uses).

Coupling is a measure of strength of connection between any two system components. The more any one components knows about other components, the tighter (worse) the coupling is between those components.

4.2 Cohesion Cohesion defines how narrowly defined an object is. Functional cohesion refers measures how strongly objects are related.

Cohesion is a measure of how logically related the parts of an individual components are to each other, and to the overall components. The more logically related the parts of components are to each other higher (better) the cohesion of that components.

4.3 Object Oriented Design Low coupling and tight cohesion is good object oriented design.

Challenge Object Task 1: With boiler plate code, make an gif image (32x32) Mario fire ball and jump to get coins

5. NEXT Design Principles Design Patterns

## 15.2 UML

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system.

<http://www.yuml.me/> Use UML with IntelliJ: UML Designer Architecture

1

Design of a system consists of classes, interfaces and collaboration. UML provides class diagram, object diagram to support this. Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support implementation perspective. Process defines the flow of the system. So the same elements as used in Design are also used to support this perspective. Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective. Modelling Types 2

Diagrams Usecase Diagram 3 4 5

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Use case diagrams depict:

Use cases. A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse. (example) Actors. An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures. (example) Associations. Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of

the initial invocation of the relationship or to indicate the primary actor within the use case. The arrowheads are typically confused with data flow and as a result I avoid their use. (example) Extend: Extend is a directed relationship that specifies how and when the behavior defined in usually supplementary (optional) extending use case can be inserted into the behavior defined in the extended use case. (example) Include is a directed relationship between two use cases which is used to show that behavior of the included use case (the addition) is inserted into the behavior of the including (the base) use case. (example) System boundary boxes (optional). You can draw a rectangle around the use cases, called the system boundary box, to indicate the scope of your system. Anything within the box represents functionality that is in scope and anything outside the box is not. System boundary boxes are rarely used, although on occasion I have used them to identify which use cases will be delivered in each major release of a system. (example) Packages (optional). Packages are UML constructs that enable you to organize model elements (such as use cases) into groups. Packages are depicted as file folders and can be used on any of the UML diagrams, including both use case diagrams and class diagrams. I use packages only when my diagrams become unwieldy, which generally implies they cannot be printed on a single page, to organize a large diagram into smaller ones. (example) Class Diagram 6

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

### 3.3.1 UML Association 9 10

#### Association

Association is reference based relationship between two classes. Here a class A holds a class level reference to class B. Association can be represented by a line between these classes with an arrow indicating the navigation direction. In case arrow is on the both sides, association has bidirectional navigation.

#### Aggregation

Aggregation (shared aggregation) is a "weak" form of aggregation when part instance is independent of the composite:

the same (shared) part could be included in several composites, and if composite is deleted, shared parts may still exist. Shared aggregation is shown as binary association decorated with a hollow diamond as a terminal adornment at the aggregate end of the association line. The diamond should be noticeably smaller than the diamond notation for N-ary associations. Shared aggregation is shown as binary association decorated with a hollow diamond.

#### Composition

Composition (composite aggregation) is a "strong" form of aggregation. Composition requirements/features listed in UML specification are:

it is a whole/part relationship, it is binary association part could be included in at most one composite (whole) at a time, and if a composite (whole) is deleted, all of its composite parts are "normally" deleted with it. Note, that UML does not define how, when and specific order in which parts of the composite are created. Also, in some cases a part can be removed from a composite before the composite is deleted, and so is not necessarily deleted as part of the composite.

#### Aggregation vs Composition

## Sequence Diagram 7

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

A sequence diagram shows object interactions arranged in time sequence.

It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

## Activity Diagram 8

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

UML - Architecture

UML - Modeling Types

UML - Use Case Diagrams

Use Case Diagram

UML Association Between Actor and Use Case

Class diagram

Sequence diagram

Activity diagram

Aggregation

UML Class Diagram: Association, Aggregation and Composition

Lecture Notes on Object-Oriented Programming: Object Oriented Aggregation

## 15.3 SOLID

**SOLID Principles** In computer programming, SOLID (single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion) is a mnemonic acronym introduced by Michael Feathers for the "first five principles" named by Robert C. Martin in the early 2000s that stands for five basic principles of object-oriented programming and design. The intention is that these principles, when applied together, will make it more likely that a programmer will create a system that is easy to maintain and extend over time. The principles of SOLID are guidelines that can be applied while working on software to remove code smells by providing a framework through which the programmer may refactor the software's source code until it is both legible and extensible. It is part of an overall strategy of agile and Adaptive Software Development.

"Dependency Management is an issue that most of us have faced. Whenever we bring up on our screens a nasty batch of tangled legacy code, we are experiencing the results of poor dependency management. Poor dependency management leads to code that is hard to change, fragile, and non-reusable."

Uncle Bob talk about several different design smells in the PPP book, all relating to dependency management. On the other hand, when dependencies are well managed, the code remains flexible, robust, and reusable. So dependency

management, and therefore these principles, are at the foundation of the -ilities that software developers desire.

SRP - Single Responsibility A class should have one, and only one, reason to change.

A class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)

Original Paper

OCP - Open/Closed You should be able to extend a classes behavior, without modifying it.

Software entities ... should be open for extension, but closed for modification."

Original Paper

LSP - Liskov Substitution Derived classes must be substitutable for their base classes.

Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program. See also design by contract.

Original Paper

ISP - Interface Segregation Make fine grained interfaces that are client specific.

Many client-specific interfaces are better than one general-purpose interface.

Original Paper

DIP - Dependency Inversion Depend on abstractions, not on concretions.

One should "depend upon abstractions, not concretions."

Original Paper

References The Principles of OOD

## 15.4 Design Patterns

Design Patterns

Creational design patterns These design patterns are all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.

Structural design patterns These design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

Behavioral design patterns These design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects.

Design Pattern QA Examples of GoF Design Patterns in Java's core libraries  
Dependency Injection vs Factory Pattern What is Inversion of Control? What is so bad about singletons? What is the basic difference between Factory and Abstract Factory Patterns? When would you use the Builder Pattern? What is the difference between Builder Design pattern and Factory Design pattern? How do the Proxy, Decorator, Adapter, and Bridge Patterns differ? Abstract Factory Pattern Creates an instance of several families of classes Intuitive 1

Volkswagen Transparent Factory in Dresden

What is it? 2 The abstract factory pattern provides a way to encapsulate a group of individual factories that have a common theme without specifying their concrete classes.

In normal usage, the client software creates a concrete implementation of the abstract factory and then uses the generic interface of the factory to create the concrete objects that are part of the theme. The client doesn't know (or care) which concrete objects it gets from each of these internal factories, since it uses only the generic interfaces of their products.

This pattern separates the details of implementation of a set of objects from their general usage and relies on object composition, as object creation is implemented in methods exposed in the factory interface.

Design

Example Code

The most interesting factories in the world Abstract factory pattern Observer Pattern Intuitive

Definition 1 The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributed event handling systems. The Observer pattern is also a key part in the familiar model-view-controller (MVC) architectural pattern. The observer pattern is implemented in numerous programming libraries and systems, including almost all GUI toolkits. Structure 2

Subject

knows its observers. Any number of Observer objects may observe a subject. provides an interface for attaching and detaching Observer objects Observer

defines an updating interface for objects that should be notified of changes in a subject. ConcreteSubject

stores state of interest to ConcreteObserver objects. sends a notification to its observers when its state changes. ConcreteObserver

maintains a reference to a ConcreteSubject object. stores state that should stay consistent with the subject's. implements the Observer updating interface to keep its state consistent with the subject's. Examples Example 1: Blog Manager Application

In this application, each user is an Observer, each blog is a Subject. When a blog post a new article (state change), user get an update. When users get update, they update their articles.

```
[code lang="java"] Blog sportBlog = new Blog("SPORT"); User user1 =
new User("Fan1"); User user2 = new User("Fan2");
sportBlog.attach(user1); sportBlog.attach(user2);
sportBlog.post(new Article("football")); sportBlog.post(new Article("swimming"));
user1.getArticles(); user2.getArticles();
sportBlog.detach(user1);' [/code]
```

Real Implementations Broadcast Receiver 3 4 on Android

More Articles <http://javapapers.com/design-patterns/observer-design-pattern/>  
Comparison Observer/Observable pattern vs Publisher/Subscriber pattern 5  
Observer/Observable pattern is mostly implemented in a synchronous way, i.e. the observable calls the appropriate method of all its observers when some event

occurs. The Publisher/Subscriber pattern is mostly implemented in an asynchronous way (using message queue). In the Observer/Observable pattern, the observers are aware of the observable. Whereas, in Publisher/Subscriber, publishers and subscribers don't need to know each other. They simply communicate with the help of message queues. Observer pattern

Broadcast Receiver

Design Patterns: Elements of Reusable Object-Oriented Software

Which design patterns are used on Android?

stackoverflow, Difference between Observer, Pub/Sub, and Data Binding



# Chương 16

# Database

View online [http://magizbox.com/training/computer\\_science/site/database/](http://magizbox.com/training/computer_science/site/database/)

## 16.1 Introduction

Relational DBMS: Oracle, MySQL, SQLite

Key-value Stores: Redis, Memcached

Document stores: MongoDB

Graph: Neo4j

Wide column stores: Cassandra, HBase

Design and Modeling (a.k.a Data Definition) 1.1 Schema A database schema of a database system is its structure described in a formal language supported by the database management system (DBMS) and refers to the organization of data as a blueprint of how a database is constructed (divided into database tables in the case of Relational Databases). The formal definition of database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real world entities are modeled in the database.

1.1.1 Type In computer science and computer programming, a data type or simply type is a classification identifying one of various types of data, such as real, integer or Boolean, that determines the possible values for that type; the operations that can be done on values of that type; the meaning of the data; and the way values of that type can be stored.

TEXT, INT, ENUM, TIMESTAMP

1.2 Cardinality (a.k.a Relationship) Foreign key, Primary key

1.2 Indexing A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns

of a database table, providing the basis for both rapid random lookups and efficient access of ordered records. Why Indexing is important?

Indexing in MySQL

CREATE INDEX NameIndex ON Employee (name) SELECT \* FROM Employee WHERE name = 'Ashish' 2. Data Manipulation Create - Read - Update - Delete Create or add new entries Read, retrieve, search, or view existing entries \* Update or edit existing entries \* Delete/deactivate existing entries /\* create \*/ CREATE TABLE Guests ( id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY, firstname VARCHAR(30) NOT NULL, lastname VARCHAR(30) NOT NULL, email VARCHAR(50) NOT NULL ) ENGINE=InnoDB; create(insert)\*/INSERT INTO Guests(firstname, lastname, email) VALUES('John', 'Doe', 'john@example.com'); read\*/SELECT \* FROM Guests WHERE id = 1/\*update\*/UPDATE Guests SET lastname = 'Doe' WHERE id = 1/\*delete\*/DELETE FROM Guests WHERE id = 1 3. Data Retrieve Transaction 3.1 Data

Get user id, user name and number of post of this user

SELECT user.id, user.name, COUNT(post.\*) AS posts FROM user LEFT OUTER JOIN post ON post.owner\_id = user.id GROUP BY user.id; Select user who only order onetime.

SELECT name, COUNT(name) AS c FROM orders GROUP BY name HAVING c = 1; Calculate the longest period (in days) that the company has gone without a hiring or firing anyone.

SELECT x.date, MIN(y.date) y\_date, DATEDIFF(MIN(y.date), x.date) days FROM (SELECT hire\_date x.date GROUP BY x.date ORDER BY days DESC LIMIT 1; Data Retrieve API

API Description get get single item Get dog by id

Dog.get(1) find find items

@see collection.find()

Find dog name "Max"

Dog.find("name": "Max") sort sort items

@see cursor.sort

Get 10 older dogs

Dog.find().sort("age", limit: 10) aggregate sum, min, max items

@see collection.aggregate

Get sum of dogs' age

Dog.find().aggregate( "sum\_age" : sum: "age" ) 3.2 Transaction A transaction

symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database. Example: Transfer 900 from Account

Bob to Alice

start transaction select balance from Account where Account\_Number = 'Bob'; select balance from Account where Account\_Number = 'Alice'; update Account set balance = balance - 900 where Account\_Number = 'Bob'; update Account set balance = balance + 900 where Account\_Number = 'Alice'; commit; // if all sql queries succeed rollback; // if any of sql queries failed or error occurs

In computer science, ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably. In the context of databases, a single logical operation on the data is called a transaction.

For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction. ![[16]

4. Backup and Restore Sometimes it is desired to bring a database back to a previous state (for many reasons, e.g., cases when the database is found corrupted due to a software error, or if it has been updated with erroneous data).

To achieve this a backup operation is done occasionally or continuously, where each desired database state (i.e., the values of its data and their embedding in database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When this state is needed, i.e., when it is decided by a database administrator to bring the database back to this state (e.g., by specifying this state by a desired point in time when the database was in this state), these files are utilized to restore that state.

5. Migration In software engineering, schema migration (also database migration, database change management) refers to the management of incremental, reversible changes to relational database schemas. A schema migration is performed on a database whenever it is necessary to update or revert that database's schema to some newer or older version. Example: Android Migration by droid-migrate

droid-migrate init -d my\_databasedroid--migrategenerateupdroid--migrategeneratedownExample : DatabaseSeedingwithLaravel

6. Active record pattern | Object-Relational Mapping (ORM) Object-relational mapping in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to create their own ORM tools.

Example

```
php
employee = newEmployee();employee->setName("Joe"); employee-> save(); Android
public class User @DatabaseField(id = true) String username; @DatabaseField String password; @DatabaseField String email; @DatabaseField String alias; public User() Implementations
Android: [ormlite-android] PHP: [Eloquent]
```

## 16.2 SQL

```
SQL SELECT * FROM WORLD
INSERT INTO
SELECT * FROM girls
```

## 16.3 MySQL

MySQL

MySQL is an open-source relational database management system (RDBMS); in July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source client-server model RDBMS. It is named after co-founder Michael Widenius's daughter, My. The SQL abbreviation stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned

by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL: Docker Docker Run `docker pull mysql` `docker run -d -p 3306:3306 --env MYSQL_ROOT_PASSWORD=docker --env MYSQL_DATABASE=docker --env MYSQL_USER=docker --env MYSQL_PASSWORD=docker` `mysqlNote : OnWindows, view your 0.0.0.0 IP by running below command line (or you can turn on Kitema)`

Docker Compose Step 1: Clone Docker Project

`git clone https://github.com/magizbox/docker-mysql.git` `mv docker-mysql`

mysql Step 2: Docker Compose

version: "2"

services: mysql: build: ./mysql/. ports: - 3306:3306 environment: - MYSQL\_ROOT\_PASSWORD=docker - MYSQL\_DATABASE=docker - MYSQL\_USER=docker - MYSQL\_PASSWORD=docker volumes: - ./data/mysql : /var/lib/mysql Dockerfile `Verify doc machinels NAME ACTIVE DRIVER STATE URL SWARM default * virtual box Running tcp : //192.168.99.100 : 2376 You can add phpmyadmin to see mysql works`

phpmyadmin: image: phpmyadmin/phpmyadmin links: - mysql environment:

- PMA\_ARBITRARY = 1 ports: - 80 : 80 See it works

Go to localhost Login with Server=mysql, Username=docker, Password=docker

## 16.4 Redis

Redis is an open source (BSD licensed), in-memory data structure store, used as database, cache and message broker. 1

It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries.

Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

Redis: Client Python Client `pipy/redis`

Installation

`pip install redis` Usage

`import redis r = redis.StrictRedis(host='localhost', port=6379, db=0)` `r.set('foo', 'bar')` `-> True`

`r.get('foo')` `-> 'bar'`

`r.delete('foo')`

after delete `r.get('foo')` `-> None` Java Client <https://redislabs.com/redis-java>

Redis: Docker Docker Run `docker run -d -p 6379:6379 redis` Docker Compose

version: "2"

services: redis: image: redis ports: - 6379:6379 Redis.io

## 16.5 MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

MongoDB provides high performance data persistence. In particular,

Support for embedded data models reduces I/O activity on database system. Indexes support faster queries and can include keys from embedded documents and arrays. MongoDB is 1 in the Document Store according to db-engines

Client Mongo Shell The mongo shell is an interactive JavaScript interface to MongoDB and is a component of the MongoDB package. You can use the mongo shell to query and update data as well as perform administrative operations.

Start Mongo

Once you have installed and have started MongoDB, connect the mongo shell to your running MongoDB instance. Ensure that MongoDB is running before attempting to launch the mongo shell.

mongo Interact with mongo via shell

Show list database > show dbs

Create or use a database > use <database<sub>n</sub>ame >> *usetestexample*

List collection > show collections

Create or use a collection > db.<collection<sub>n</sub>ame >> *db.new\_collectionexample*

Read document > db.new<sub>c</sub>ollection.find()

Insert new document > db.new<sub>c</sub>ollection.insertOne("a" : "b")

Update document > db.new<sub>c</sub>ollection.update("a" : "b", set: "a": "bcd")

Remove document > db.new<sub>c</sub>ollection.remove("a" : "b") *PyMongo—PythonClientPyMongois a Python*

Installation We recommend using pip to install pymongo on all platforms:

```
pip install pymongo Usage import pymongo create connection client = pymongo.MongoClient('127.0.0.1', 27017) -> MongoClient(host=['127.0.0.1:27017'], documentclass = dict, tzaware = False, connect = True)
```

```
create database db = client.dbtest -> Database(MongoClient(host = ['127.0.0.1 : 27017'], documentclass = dict, tzaware = False, connect = True), u'dbtest')
```

```
create collection (collection is the same with table in SQL) collection = db.newcollection
```

```
insert document to collection (document is the same with rows in SQL)
```

```
db.collection.insertone("c" : "d") -> < pymongo.results.InsertOneResult at 0x7f7eab3c9f00 >
```

```
read document of collection db.newcollection.findone("c" : "d") -> u'd' : ObjectId('589a8195f23e627a9')
```

```
update documents of collection db.newcollection.update("c" : "d", set: "c":
```

```
"def" ) -> u'n': 1, u'nModified': 1, u'ok': 1.0, 'updatedExisting': True
```

```
remove document of collection db.newcollection.remove("c" : "def") -> u'n' : 1, u'ok' : 1.0 Docker Docker Run Run images and share port
```

```
docker run -p 27017:27017 mongo:latest
```

## Chương 17

# Hệ điều hành

Những phần mềm không thể thiếu

\* Trình duyệt Google Chrome (với các extensions Scihub, Mendeley Desktop, Adblock) \* Adblock extension \* Terminal (Oh-my-zsh) \* IDE Pycharm để code python \* Quản lý phiên bản code Git \* Bộ gõ ibus-unikey trong Ubuntu hoặc unikey (Windows) (Ctrl-Space để chuyển đổi ngôn ngữ) \* CUDA (lập trình trên GPU)

**\*\*Xem thông tin hệ thống\*\***

Phiên bản ‘ubuntu 16.04’

```
sudo apt-get install sysstat
```

Xem hoạt động (

“ mpstat -A “

CPU của mình có bao nhiêu core, bao nhiêu siblings

“ cat /proc/cpuinfo

processor : 23 vendor\_id : GenuineIntelcpu family : 6model : 62modelname :

Intel(R) Xeon(R) CPU E5-2430v2@2.50GHzstepping : 4microcode : 0x428cpu MHz :

1599.707cachesize : 15360KBphysicalid : 1siblings : 12coreid : 5cpucore : 5

6apicid : 43initialapicid : 43fpu : yesfpu\_exception : yescpuidlevel : 13wp :

yesflags : fpuvmdepsetscmsrpaemccecx8apicsepmtrrpgemcacrmoovpatpse36clflushdtsacpimxfxsrssesse

5005.20clflushsize : 64cache\_alignment : 64addresssizes : 46bitsphysical, 48bitsvirtualpowermanagement :

“

Kết quả cho thấy cpu của 6 core và 12 siblings

## Chương 18

# Ubuntu

**\*\*Chuyện terminal\*\***

Terminal là một câu chuyện muôn thưở của bất kì ông coder nào thích customize, đẹp, tiện (và bug kinh hoàng). Hiện tại mình đang thấy combo này khá ổn Terminal (Ubuntu) (Color: Black on white, Build-in schemes: Tango) + zsh + oh-my-zsh (fishy-custom theme). Những features hay ho

\* Làm việc tốt trên cả Terminal (white background) và embedded terminal của Pycharm (black background) \* Hiển thị folder dạng ngắn (chỉ ký tự đầu tiên) \* Hiển thị branch của git ở bên phải

![Imgur](https://i.imgur.com/q53vQdH.png)

**\*\*Chuyện bộ gõ\*\***

Làm sao để khởi động lại ibus, thỉnh thoảng lại chết bất đắc kì tử <sup>[1]</sup> “ibus – daemonibusrestart”

**\*\*Chuyện lỗi login loop\*\***

Phiên bản: ‘ubuntu 16.04’

27/12/2017: Lại dính lỗi không thể login. Lần này thì lại phải xóa bạn KDE đi. Kể cũng hơn buồn. Nhưng nhất quyết phải enable được tính năng Windows Spreading (hay đại loại thế). Hóa ra khi ubuntu bị lỗi không có launcher hay toolbar là do bạn unity plugin chưa được enable. Oái. Sao người hiền lành như mình suốt ngày bị mấy lỗi vớ vẩn thế không biết.

20/11/2017: Hôm nay đen thật, dính lỗi login loop. Fix mãi mới được. Thôi cũng kệ. Cảm giác bạn KDE này đỡ bị lỗi ibus-unikey hơn bạn GNOME. Hôm nay cũng đổi bạn zsh theme. Chọn mãi chẳng được bạn nào ổn ổn, nhưng không thể chịu được kiểu suggest lỗi nữa rồi. Đôi khi thấy default vẫn là tốt nhất.

21/11/2017: Sau một ngày trải nghiệm KDE, cảm giác giao diện mượt hơn GNOME. Khi overview windows với nhiều màn hình tốt và trực quan hơn. Đặc biệt là không bị lỗi ibus nữa. Đổi terminal cũng cảm giác ổn ổn. Không bị lỗi suggest nữa.

<sup>[1]</sup> : <https://askubuntu.com/questions/389903/ibus-doesnt-seem-to-restart>

## Chương 19

# Networking

View online [http://magizbox.com/training/computer\\_science/site/networking/](http://magizbox.com/training/computer_science/site/networking/)

TCP/IP TCP/IP is the protocol that has run the Internet for 30 years.

How TCP/IP works

Read More

Happy 30th Anniversary, Internet and TCP/IP!!! P2P Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.[1] Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model in which the consumption and supply of resources is divided. Emerging collaborative P2P systems are going beyond the era of peers doing similar things while sharing resources, and are looking for diverse peers that can bring in unique resources and capabilities to a virtual community thereby empowering it to engage in greater tasks beyond those that can be accomplished by individual peers, yet that are beneficial to all the peers.

bridge vs NAT When you create a new virtual machine, you have one of many options when it comes to choosing your network connectivity. Two common options are to use either bridged networking or network address translation (NAT). So, what exactly does that look like? Take a look at the figure below.

NAT: In this diagram, the vertical line next to the firewall represents the production network and you can see that 192.168.1.1 is the IP address of the company's firewall that connects them to the Internet. There is also a virtual host with three virtual machines running inside it. The big red circle represents the virtual adapter to which NAT-based virtual machines connect (172.16.1.1). You can see that there are two such virtual machines with IP addresses of 172.16.1.2 and 172.16.1.3. When you configure a virtual machine as using NAT, it doesn't see the production network directly. In fact, all traffic coming from the virtual machine will share the VM host's IP address. Behind the scenes, traffic from the virtual machines is routed on the virtual host and sent out via the host's physical adapter and, eventually, to the Internet.

bridge: The third virtual machine (192.168.1.3) is configured in "bridged"



mode which basically means that the virtual network adapter in that virtual machine is bridged to the production network and that virtual machine operates as if it exists directly on the production network. In fact, this virtual machine won't even be able to see the two NAT-based virtual machines since they're on different networks.

Read more: NAT vs. bridged network: A simple diagram

## Chương 20

# UX - UI

View online [http://magizbox.com/training/computer\\_science/site/ux/](http://magizbox.com/training/computer_science/site/ux/)

1. Design Principles UI Design Do's and Don'ts Android Design Principles
2. Design Trends 2.1 Material Design 1 components

We challenged ourselves to create a visual language for our users that synthesizes the classic principles of good design with the innovation and possibility of technology and science. This is material design. This spec is a living document that will be updated as we continue to develop the tenets and specifics of material design.

Tools

materialpalette.com Icon: fa2png UI Components

Data Binding Transclusion Directive - Fragments

Messaging Intent Android 1

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

Intents are objects of the `android.content.Intent` type. Your code can send them to the Android system defining the components you are targeting. For example, via the `startActivity()` method you can define that the intent should be used to start an activity.

An intent can contain data via a `Bundle`. This data can be used by the receiving component.

Style Theme Android Development: Explaining Styles and Themes, <https://m.youtube.com/watch?v=M>

Responsive Design Support Multi Screen 2

Intent Android

Support Multi Screen

## Chương 21

# Service-Oriented Architecture

View online [http://magizbox.com/training/computer\\_science/site/software\\_architecture/](http://magizbox.com/training/computer_science/site/software_architecture/)

A service-oriented architecture (SOA) is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation are independent of any vendor, product or technology. 2

Generally accepted view 1 Boundaries are explicit Services are autonomous Services share schema and contract, not class Service compatibility is based on policy Microservices In computing, microservices is a software architecture style in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. These services are small building blocks, highly decoupled and focussed on doing a small task, facilitating a modular approach to system-building. One of concepts which integrates microservices as a software architecture style is dew computing. 1

Properties 2 Each running in its own process Communicating with lightweight mechanisms, often an HTTP resource API Build around business capabilities Independently deployable fully automated deployment Maybe in a different programming language and use different data storage technologies Monolith vs Microservice Monolith Microservice Simplicity Partial Deployment Consistency Availability Inter-module refactoring Preserve Modularity Multiple Platforms Benefits 4 Their small size enables developers to be most productive. It's easy to comprehend and test each service. You can correctly handle failure of any dependent service. They reduce impact of correlated failures. Web Service RESTful API

REST Client Sense (Beta)

A JSON aware developer console to ElasticSearch.

API Document and Client Generator <http://swagger.io/swagger-editor/>

API Client CRUD Pet

API Client Method URL Body Return Body Method GET /pets [Pet] PetApi.list()

POST /pets/ Pet PetApi.create(pet) GET /pets/pet;dPetPetApi.get(pet;d)PUT /pets/pet;dPetPetPetA

CRUD Store

GET /stores StoreApi.list() ... ... Relationships

Many to many

Example [<https://api.facebook.com/method/links.getStats?url=Microservices>]

Slide 11/42, Micro-services

Martin Fowler, Microservices, youtube

Rick E. Osowski, Microservices in action, Part 1: Introduction to microservices, IBM developerworks

## Chương 22

# License

View online [http://magizbox.com/training/computer\\_science/site/licenses/](http://magizbox.com/training/computer_science/site/licenses/)  
Licenses More Licenses  
More Open Source Licenses Choose A License Top 20 Open Source Licenses

## Chương 23

# Semantic Web

View online [http://magizbox.com/training/semantic<sub>w</sub>eb/site/](http://magizbox.com/training/semantic_web/site/)

The Semantic Web is an extension of the Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats and exchange protocols on the Web, most fundamentally the Resource Description Framework (RDF).

According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries". The term was coined by Tim Berners-Lee for a web of data that can be processed by machines. While its critics have questioned its feasibility, proponents argue that applications in industry, biology and human sciences research have already proven the validity of the original concept.

### 23.1 Web 3.0

Tim Berners-Lee has described the semantic web as a component of "Web 3.0".

People keep asking what Web 3.0 is. I think maybe when you've got an overlay of scalable vector graphics – everything rippling and folding and looking misty – on Web 2.0 and access to a semantic Web integrated across a huge space of data, you'll have access to an unbelievable data resource ...

—Tim Berners-Lee, 2006

"Semantic Web" is sometimes used as a synonym for "Web 3.0", though the definition of each term varies.

### 23.2 RDF

### 23.3 SPARQL

SPARQL (pronounced "sparkle", a recursive acronym for SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. It was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web. On 15 January

2008, SPARQL 1.0 became an official W3C Recommendation, and SPARQL 1.1 in March, 2013.

SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

A SPARQL query

Anatomy of a query

SPARQL has four query forms. These query forms use the solutions from pattern matching to form result sets or RDF graphs. The query forms are:

SELECT Returns all, or a subset of, the variables bound in a query pattern match. CONSTRUCT Returns an RDF graph constructed by substituting variables in a set of triple templates. ASK Returns a boolean indicating whether a query pattern matches or not. DESCRIBE Returns an RDF graph that describes the resources found. Example

Query Result Data filename: ex008.rq

PREFIX ab: <http://learningsparql.com/ns/addressbook>

SELECT ?person WHERE ?person ab:homeTel "(229) 276-5135" Offline query example GET CRAIG EMAILS PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns> PREFIX owl: <http://www.w3.org/2002/07/owl> PREFIX xsd: <http://www.w3.org/2001/XMLSchema> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema> PREFIX : <http://www.semanticweb.org/lananh/ontologies/2016/10/untitled-ontology-3>

SELECT ?craigEmail WHERE :craig :email ?craigEmail . Online query example PREFIX ab: <http://learningsparql.com/ns/addressbook>

SELECT ?craigEmail WHERE ab:craig ab:email ?craigEmail . Query in dbpedia.org Example

SELECT \* WHERE ?a ?b ?c . LIMIT 20

**Phần IV**

**Khoa học dữ liệu**



## Chương 24

# Học máy

- Vấn đề với HMM và CRF?
- Học MLE và MAP?

**\*\*Có bao nhiêu thuật toán Machine Learning?\***

Có rất nhiều thuật toán Machine Learning, bài viết [Điểm qua các thuật toán Machine Learning hiện đại](<https://ongxuanhong.wordpress.com/2015/10/22/diem-qua-cac-thuat-toan-machine-learning-hien-dai/>) của Ông Xuân Hồng tổng hợp khá nhiều thuật toán. Theo đó, các thuật toán Machine Learning được chia thành các nhánh lớn như ‘regression’, ‘bayesian’, ‘regularization’, ‘decision tree’, ‘instance based’, ‘dimensionality reduction’, ‘clustering’, ‘deep learning’, ‘neural networks’, ‘associated rule’, ‘ensemble’... Ngoài ra thì còn có các cheatsheet của [sklearn]([http://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)).

Việc biết nhiều thuật toán cũng giống như ra đường mà có nhiều lựa chọn về xe cộ. Tuy nhiên, quan trọng là có task để làm, sau đó thì cập nhật SOTA của task đó để biết các công cụ mới.

**\*\*Xây dựng model cần chú ý điều gì?\***

Khi xây dựng một model cần chú ý đến vấn đề tối ưu hóa tham số (có thể sử dụng [GridSearchCV]([sklearn.model\\_selection.GridSearchCV](https://sklearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)))

Bài phát biểu này có vẻ cũng rất hữu ích [PYCON UK 2017: Machine learning libraries you’d wish you’d known about](<https://www.youtube.com/watch?v=nDF78FOhpI>).

\* [DistrictDataLabs/yellowbrick](<https://github.com/DistrictDataLabs/yellowbrick>) (giúp visualize model được train bởi sklearn) \* [marcotcr/lime](<https://github.com/marcotcr/lime>) (giúp inspect classifier) \* [TeamHG-Memex/eli5](<https://github.com/TeamHG-Memex/eli5>) (cũng giúp inspect classifier, hỗ trợ nhiều model như xgboost, crfsuite, đặc biệt có TextExplainer sử dụng thuật toán từ eli5) \* [rhiever/tpot](<https://github.com/rhiever/tpot>) (giúp tối ưu hóa pipeline) \* [dask/dask](<https://github.com/dask/dask>) (tính toán song song và lập lịch)

Ghi chú về các thuật toán trong xử lý ngôn ngữ tự nhiên tại [underthesea.flow/wiki](<https://github.com/magizbox/underthesea.flow/wiki/Develop>)

Framework để train, test hiện tại vẫn rất thoải mái sklearn. tensorboard cung cấp phần log cũng khá hay.

[Câu trả lời hay](<https://www.quora.com/What-are-the-most-important-machine-learning-techniques-to-master-at-this-time/answer/Sean-McClure-3?srid=5O2u>)

cho câu hỏi [Những kỹ thuật machine learning nào quan trọng nhất để master?](<https://www.quora.com/What-are-the-most-important-machine-learning-techniques-to-master-at-this-time>), đặc biệt là dẫn đến bài [The State of ML and Data Science 2017](<https://www.kaggle.com/surveys/2017>) của Kaggle.

**\*\*Tài liệu học PGM\*\***

[Playlist youtube](<https://www.youtube.com/watch?v=WPSQfOkb1M8&list=PL50E6E80E8525B59C>) khóa học Probabilistic Graphical Models của cô Daphne Koller. Ngoài ra còn có một [tutorial](<http://mensxmachina.org/files/software/demos/bayesnetdemo.html>) dễ hơi ở đầu về tạo Bayesian network

**\*\*[Chưa biết] Tại sao Logistic Regression lại là Linear Model?\*\***

Trong quyển Deep Learning, chương 6, trang 165, tác giả có viết

““ Linear models, such as logistic regression and linear regression, are appealing because they can be efficiently and reliably, either in closed form or with convex optimization ““

Mình tự hỏi tại sao logistic regression lại là linear, trong khi nó có sử dụng hàm logit (nonlinear)? Tìm hiểu hóa ra cũng có bạn hỏi giống mình trên [stats.stackexchange.com](<https://stats.stackexchange.com/questions/93569/why-is-logistic-regression-a-linear-classifier>). Ngoài câu trả lời trên stats.stackexchange, đọc một số cái khác [Generalized Linear Models, SPSS Statistics 22.0.0]([https://www.ibm.com/support/knowledgecenter/IT23241\\_22.0.0/IntroductiontoGeneralizedLinearModels\\_AnalysisofDiscreteData\\_PennsylvaniaStateUniversity](https://www.ibm.com/support/knowledgecenter/IT23241_22.0.0/IntroductiontoGeneralizedLinearModels_AnalysisofDiscreteData_PennsylvaniaStateUniversity)](<https://onlinecourses.science.psu.edu/stat504/node/216>)*engvnchahiulm*.

Hiện tại chỉ hiểu là các lớp model này chỉ có thể hoạt động trên các tập linear separable, có lẽ do việc map input  $x$ , luôn có một liên kết linear  $latex wx$ , trước khi đưa vào hàm non-linear.

**\*\*Các tập dữ liệu thú vị\*\***

**\*Iris dataset\***: dữ liệu về hoa iris

Là một ví dụ cho bài toán phân loại

**\*Weather problem\***: dữ liệu thời tiết. Có thể tìm được ở trong quyển Data

Mining: Practical Machine Learning Tools and Techniques

Là một ví dụ cho bài toán cây quyết định

Deep Learning

**\*\*Tài liệu Deep Learning\*\***

Lang thang thế nào lại thấy trang này [My Reading List for Deep Learning!]([https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/DL\\_Reading\\_List.pdf](https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/DL_Reading_List.pdf))*camtanhMicrosoft.TrongỞ, (Ờngnhìn)cDee*

Các layer trong deep learning [2]

Sparse Layers

**\*\*nn.Embedding\*\***(<http://pytorch.org/docs/master/nn.html#embedding>) ([hướng dẫn]([http://pytorch.org/tutorials/beginner/nlp/word\\_embeddings\\_tutorial.html](http://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html)))*grepcode* : [Shawn1993/cnn-text-classification-pytorch](<https://github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.py#L18>)*(Engvaitrnhmtlookuptable, mapmtwordvidenseve*


Convolution Layers

**\*\*nn.Conv1d\*\***(<http://pytorch.org/docs/master/nn.html#conv1d>), **\*\*nn.Conv2d\*\***(<http://pytorch.org/docs/master/nn.html#conv2d>), **\*\*nn.Conv3d\*\***(<http://pytorch.org/docs/master/nn.html#conv3d>) [1]*grepcode* : [Shawn1993/cnn-text-classification-pytorch](<https://github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.py#L20-L24>), [galsang/CNN-sentence-classification-pytorch](<https://github.com/galsang/CNN-sentence-classification-pytorch/blob/master/model.py#L36-L38>)

Các tham số trong Convolution Layer

\* `kernel_size` (*hay* `filter_size`)

Đối với NLP, `kernel_sizethnbnregion_size*word_dim` (*Ờiviconv1d*) *hay* (`region_size, word_dim`) *Ờiviconv2d*

Quá trình tạo feature map đối với region size bằng 2  
 (<https://media.giphy.com/media/l2QE2y1UQP7vIgit/giphy.gif>)  
 \* `'in_channels', 'out_channels' (labeled '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://d3kbpzbmcyntmx.cloudfront.net/wp-content/uploads/2015/11/Screenshot-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_dim'`

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

 (<http://d3kbpzbmcyntmx.cloudfront.net/wp-content/uploads/2015/11/Screenshot-2015-11-06-at-8.03.47-AM.png>)

Description	Values
Google word2vec	filter region size   (3, 4, 5)   feature maps   100
activation function	ReLU   pooling   1-max pooling   dropout rate   0.5
<i>latex</i> $s = 22$ 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>)

**\*\*BTS\*\***

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning [tại đây] ([https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM\\_TlUGxydQ-Qi1JOPFr8/edit?usp=sharing](https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM_TlUGxydQ-Qi1JOPFr8/edit?usp=sharing))

[<sup>1</sup>] : [Understanding Convolutional Neural Networks for NLP] (<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp>) [<sup>2</sup>] : [<http://pytorch.org/docs/master/nn.html>] (<http://pytorch.org/docs/master/nn.html>)

## Chương 25

# Học sâu

### 25.1 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.

### 25.2 Các layer trong deep learning

#### 25.2.1 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

#### 25.2.2 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>



\* *'in\_channels'*, *'out\_channels'* (*lslnsg'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.



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<sub>dim</sub>' đối với 'nn.conv1d'

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,



Description	Values
Google word2vec	input word vectors
filter region size	(3, 4, 5)
feature maps	100
activation function	ReLU
pooling	1-max pooling
dropout rate	0.5
l <sub>2</sub> 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>)

\*\*BTS\*\*

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning [tại đây]([https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM\\_TlUGxydQ-Qi1JOPFr8/edit?usp=sharing](https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM_TlUGxydQ-Qi1JOPFr8/edit?usp=sharing))

[<sup>1</sup>] : [UnderstandingConvolutionalNeuralNetworksforNLP](<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp>)[<sup>2</sup>] : [<http://pytorch.org/docs/master/nn.html>](<http://pytorch.org/docs/master/nn.html>)

## Chương 26

# Xử lý ngôn ngữ tự nhiên

**\*\*05/01/2018\*\***: "điên đầu" với Sphinx và HTK

HTK thì đã bỏ rồi vì quá lằng nhằng.

Sphinx thì setup được đối với dữ liệu nhỏ rồi. Nhưng không thể làm nó hoạt động với dữ liệu của VIVOS. Chắc hôm nay sẽ switch sang Kaldi vậy.

**\*\*26/12/2017\*\***: Automatic Speech Recognition 100

Sau mấy ngày "vật lộn" với code base của Truong Do, thì cuối cùng cũng produce voice được. Cảm giác rất thú vị. Quyết định làm luôn ASR. Tìm mãi chẳng thấy code base đâu (chắc do lĩnh vực mới nên không có kinh nghiệm). May quá lại có bạn frankydotid có project về nhận diện tiếng Indonesia ở [github](<https://github.com/frankydotid/Indonesian-Speech-Recognition>). Trong README.md bạn đấy bảo là phải cần đọc HTK Book. Tốt quá đang cần cơ bản.

**\*\*20/12/2017\*\***: Text to speech 100

Cảm ơn project rất hay của [bạn Truong Do ở vais]([https://vais.vn/vi/tai-ve/hts\\_for\\_vietnamese/](https://vais.vn/vi/tai-ve/hts_for_vietnamese/)), *nukhngcprojectnychcmnhphimtrtnhiuthigianmicOcphinbntexttospeechOutin*.

Tóm lại thì việc sinh ra tiếng nói từ text gồm 4 giai đoạn

1. Sinh ra features từ file wav sử dụng tool sptk 2. Tạo một lab, trong đó có dữ liệu huấn luyện (những đặc trưng của âm thanh được trích xuất từ bước 1), text đầu vào 3. Sử dụng htk để train dữ liệu từ thư mục lab, đầu ra là một model 4. Sử dụng model để sinh ra output với text đầu vào, dùng *hts\_engineOdecode, ktquOcwav files*.

Phù. 4 bước đơn giản thế này thôi mà không biết. Lọc cả internet ra mãi chẳng hiểu, cuối cùng file phân tích file 'train.sh' của bạn Truong Do mới hiểu. Ahihi

**\*\*24/11/2017\*\***: Nhánh của Trí tuệ nhân tạo mà hiện tại mình đang theo đuổi. Project hiện tại là [underthesea](<https://github.com/magizbox/underthesea>). Với mục đích là xây dựng một toolkit cho xử lý ngôn ngữ tự nhiên tiếng Việt.

## Chương 27

# Nhận dạng tiếng nói

Trong hệ thống nhận dạng tiếng nói, tín hiệu âm thanh được thu thập như những mẫu phù hợp cho quá trình xử lý của máy tính và được đưa vào quá trình nhận diện. Đầu ra của hệ thống là một câu phụ đề của câu nói.

Nhận dạng tiếng nói là một nhiệm vụ phức tạp và hệ thống tốt nhất trong nhận dạng tiếng nói rất phức tạp. Có rất nhiều cách tiếp cận cho mỗi thành phần. Trong phần này, người viết chỉ muốn đưa ra một cái nhìn tổng thể về nhận dạng tiếng nói, các khó khăn chính, các thành phần cơ bản, chức năng và tương tác của chúng trong một hệ thống nhận dạng tiếng nói.

Các thành phần của hệ thống nhận dạng tiếng nói



Trong bước thứ nhất, trích rút thông tin *\*Feature Extraction\**, các mẫu tín hiệu được tham số hóa. Mục tiêu là trích xuất ra một tập các tham số (đặc trưng) từ tín hiệu có nhiều thông tin hữu ích nhất cho quá trình phân loại. Các đặc trưng chính được trích xuất với điều kiện *\*thích nghi\** với các sự thay đổi của âm thanh và *\*nhảy cảm\** với các nội dung ngôn ngữ.

Trong module phân loại, các vector đặc trưng được ánh xạ với các pattern, được gọi là *\*mô hình âm học\** (acoustic model). Mô hình học thường là HMM được train với toàn bộ từ, hay âm như là một đơn vị ngôn ngữ.

*\*Từ điển phát âm\** (pronunciation dictionary) định nghĩa cách kết hợp âm cho các ký tự. Nó có thể chứa cách phát âm khác nhau cho cùng một từ. Bảng 1 hiển thị chính xác một từ điển. Từ (grapheme) ở cột bên trái ứng với cách phát âm (các âm) ở cột bên phải (các ký tự âm trong bảng được dùng phổ biến đối với tiếng Anh)

word   pronunciation	INCREASE	ih n
k r iy s	INCREASED	ih n k r iy s t
INCREASES	ih n k r iy s ah z	
INCREASING	ih n k r iy s ih ng	
INCREASINGLY	ih n k r iy s ih ng l iy	
INCRECIBLE	ih n k r eh d ah b ah l	

*\*Mô hình ngôn ngữ\** (language model) chứa các thông tin về cú pháp. Mục tiêu để dự đoán khả năng một từ xuất hiện sau các từ khác trong một ngôn ngữ. Nói cách khác, xác suất để một từ  $k$  xảy ra sau khi  $k-1$  từ sau đó được định nghĩa bởi  $latexP(w_k|w_{k-1}, w_{k-2}, ..., w_1)$

**\*\*Mô hình hóa sub-word với HMMs\*\***

Trong các hệ thống ASR, HMMs được dùng để biểu diễn các đơn vị dưới từ (ví dụ như âm). Với ngôn ngữ, thông thường có 40 âm. Số lượng âm phụ thuộc

vào từ điển được sử dụng. Số lượng âm phụ thuộc vào từ điển được sử dụng. Mô hình từ có thể được xây dựng bằng cách kết hợp các mô hình dưới từ.

Trong thực tế, khi nhận dạng một âm phụ thuộc rất nhiều vào các âm bên cạnh. Do đó, mô hình âm phụ thuộc ngữ cảnh (\*context dependence\*) được sử dụng rất phổ biến. Mô hình \*biphone\* chú ý đến âm bên trái hoặc âm bên phải, mô hình \*triphone\* chú ý đến cả hai phía, với một âm, các mô hình khác nhau được sử dụng trong ngữ cảnh khác nhau. Hình dưới thể hiện các mô hình monophone, biphone và triphone của từ \*bat\* (b ae t)



Quá trình huấn luyện

**\*\*Huấn luyện các mô hình monophone\*\***

Một mô hình monophone là một mô hình âm học, trong đó không chứa thông tin ngữ cảnh về các âm trước và sau. Nó được sử dụng như thành phần cơ bản cho các mô hình triphone - mô hình sử dụng những thông tin về ngữ cảnh.

Việc huấn luyện sử dụng framework Gaussian Mixture Model/Hidden Markov Model.

**\*\*Đóng hàng âm thanh trong mô hình âm học\*\***

Các tham số trong mô hình âm học được tính toán trong quá trình huấn luyện; tuy nhiên, quá trình này có thể được tối ưu hóa bởi việc lặp lại quá trình huấn luyện và đóng hàng. Còn lại là huấn luyện Viterbi (liên quan đến phương pháp này, nhưng dùng nhiều khối lượng tính toán hơn là thuật toán Forward-Backward và Expectation Maximization). Bằng cách đóng hàng âm thanh - phụ đề với mô hình âm học hiện tại, các thuật toán huấn luyện có thể sử dụng kết quả này để cải thiện và hiệu chỉnh tham số của mô hình. Do đó, mỗi quá trình huấn luyện sẽ theo bởi một bước đóng hàng trong đó âm thanh và văn bản được đóng hàng lại.

**\*\*Huấn luyện các mô hình triphone\*\***

Trong khi các mô hình monophone đơn giản biểu diễn các đặc trưng âm thanh như một đơn âm, trong khi các âm vị sẽ thay đổi đáng kể phụ thuộc vào ngữ cảnh. Mô hình triphone thể hiện một âm trong ngữ cảnh với hai âm bên cạnh.

Đến đây, một vấn đề là không phải tất cả các đơn vị triphone được thể hiện trong dữ liệu huấn luyện. Có tất cả (of phonemes)<sup>3</sup> *triphone, nhngchcmttpthcstntitrongdliu.Hnna, ccOnvxyr*

**\*\*Đóng hàng các mô hình âm học và huấn luyện lại các mô hình triphone\*\***

Lặp lại các bước dòng hàng âm thanh và huấn luyện các mô hình triphone với các thuật toán huấn luyện để hiệu chỉnh mô hình. Các phương pháp phổ biến là delta+delta-delta, LDA-MLLT và SAT. Các giải thuật đóng hàng bao gồm đóng hàng cho từng người nói và FMLLR.

**\*\*Các thuật toán huấn luyện\*\***

Huấn luyện delta+delta-delta tính các đặc trưng delta và double-delta, hay các hệ số động, để thêm vào các đặc trưng MFCC. Delta và delta-delta là các đặc trưng số học, tính các đạo hàm bậc 1 và 2 của tín hiệu. Do đó, phép tính toán này thường được thực hiện trên một window của các đặc trưng vector. Trong khi một window của hai đặc trưng vector có thể hiệu quả, nó là các xấp xỉ thô (giống như delta-difference là một xấp xỉ thô của đạo hàm). Đặc trưng delta được tính toán trong các window của các đặc trưng cơ bản, trong khi delta-delta được tính toán trong các window của đặc trưng delta.

LDA-MLLT viết tắt của Linear Discriminant Analysis - Maximum Likelihood Linear Transform. Linear Discriminant Analysis lấy các đặc trưng vector



và xây dựng các trạng thái HMM, nhưng giảm thiểu không gian vector. Maximum Likelihood Linear Transform lấy các đặc trưng được giảm từ LDA, và thực hiện các biến đổi đối với từng người nói. MLLT sau đó thực hiện một bước chuẩn hóa, để giảm sự khác biệt giữa các người nói.

SAT viết tắt của Speaker Adaptive Training. SAT cũng thực hiện các chuẩn hóa đối với người nói bằng cách thực hiện biến đổi trên mỗi người nói. Kết quả của quá trình này đồng nhất và chuẩn hóa hơn, cho phép mô hình có thể sử dụng những tham số này để giảm thiểu sự biến đổi của âm, đối với từng người nói hoặc môi trường thu.

**\*\*Các thuật toán đóng hàng\*\***

Thuật toán dòng hàng luôn luôn cố định, trong đó các kịch bản chấp nhận các loại đầu vào âm học khác nhau. Dòng hàng đối với từng người nói, sẽ tách biệt thông tin giữa các người nói trong quá trình đóng hàng.

fMLLR viết tắt của Feature Space Maximum Likelihood Linear Regression. Sau quá trình huấn luyện SAT, các mô hình âm học không huấn luyện trên các đặc trưng ban đầu, mà đối với các đặc trưng chuẩn hóa theo người nói. Với quá trình đóng hàng, xóa bỏ sự khác biệt giữa người nói (bằng cách nghịch đảo ma trận fMLLR), sau đó loại bỏ nó khỏi mô hình \*bằng cách nhân ma trận nghịch đảo với đặc trưng vector). Mô hình âm học quasi-speaker-independent có thể sử dụng trong quá trình đóng hàng.

**Dóng hàng (Forced Alignment)**

Hệ thống nhận dạng tiếng nói sử dụng một máy tìm kiếm bên cạnh mô hình âm học và ngôn ngữ trong đó chứa tập các từ, âm và tập dữ liệu để đối chiếu với dữ liệu âm thanh cho câu nói. Máy tìm kiếm này sử dụng các đặc trưng được trích xuất bởi dữ liệu âm thanh để xác định sự xuất hiện của từ, âm và đưa ra kết quả.



Quá trình dòng hàng cũng tương tự như vậy, nhưng khác ở một điểm quan trọng. Thay vì đưa vào tập các từ có thể để tìm kiếm, máy tìm kiếm đưa vào đoạn phụ đề tương ứng với câu nói. Hệ thống sau đó đóng hàng dữ liệu văn bản với dữ liệu âm thanh, xác định đoạn nào trong âm thanh tương ứng với từ cụ thể nào trong dữ liệu văn bản.



Dóng hàng có thể sử dụng để đóng âm trong dữ liệu với bản với dữ liệu âm thanh, giống như hình dưới đây, các âm được xác định trong từng đoạn của âm thanh.



**Hidden Markov Model**

Hidden Markov Model (HMM) là mô hình trọng số với các trọng số ở cung, chỉ khả năng xuất hiện của cung.

\*Một trong những ứng dụng của HMM, là phán đoán chuỗi các trạng thái thay đổi, dựa vào chuỗi các quan sát\*

Các trọng số trong trạng thái gọi là observation likelihood, các trọng số ở cung gọi là transition likelihood.

Sau đây là một ví dụ:

\* Thời tiết trong một ngày có thể là NÓNG hoặc LẠNH \* Khi trời NÓNG,  
20\* Khi trời NÓNG, 30\* (  
qimg-a6744f9e17e59f3729d6fef02d54391b.webp)

Giờ, giả sử chúng ta quan sát trong 3 ngày, bạn dùng 1,2,3 viên đá. Thời tiết có khả năng diễn ra như thế nào?

Đến đây chúng ta dùng thuật toán Viterbi. Về cơ bản, nó là dynamic programming với hai chiều [state, position]<sub>in sequence</sub>

Gọi S là trạng thái hiện tại HOT, COLD trong quan sát i, S' là trạng thái trước đó, và A là lượng đá tiêu thụ 1, 2, 3 trong quan sát i

$$\text{latex} Viterbi[S, i] = Viterbi[S', i - 1] * p(S|S') * p(A|S)$$

$$\text{latex} V[S, i] = V[S', i - 1] * \text{transition}_{ikelihood} * \text{observation}_{ikelihood}$$

HMM được sử dụng trong các hệ thống thoại miễn

1. Có hữu hạn các trạng thái nội tại (internal state), là nguyên nhân của các sự kiện (external events) (các quan sát) 2. Trạng thái nội tại không quan sát được (hidden) 3. Trạng thái hiện tại chỉ phụ thuộc vào trạng thái trước đó (quá trình Markov)

Wow! George nhanh chóng liên hệ vụ của anh đây với mô hình HMM. George nhận ra rằng CCTV footage từ các cặp có thể coi như là chuỗi quan sát được, anh đây có thể dùng mô hình và sử dụng nó để phát hiện hành vi ẩn mà Bob và William hoạt động.

**\*\*3 vấn đề cơ bản\*\*** được Jack Ferguson giới thiệu trong những năm 1960

Vấn đề 1 (Likelihood): Cho một HMM  $\lambda = (A, B)$  và một chuỗi quan sát  $O$ , xác định likelihood  $P(O|\lambda)$

Vấn đề 2 (Decoding): Cho một chuỗi quan sát  $O$ , và một HMM  $\lambda = (A, B)$ , xác định chuỗi ẩn  $Q$  tốt nhất

Vấn đề 3 (Learning): Cho một chuỗi quan sát  $O$ , một tập các trạng thái trong HMM, học các tham số  $A$  và  $B$

**\*\*Likelihood Computation\*\***

Vấn đề đầu tiên là tính xác suất xảy ra của một chuỗi quan sát. Ví dụ, trong bài toán ăn đá ở hình 9.3, xác suất xảy ra chuỗi \*3 1 3\* là bao nhiêu?

**\*\*Tính toán Likelihood\*\***: Chuỗi một HMM  $\lambda = (A, B)$ , và mỗi chuỗi quan sát  $O$ , xác định likelihood  $P(O|\lambda)$

Thuật toán Forward, nếu sử dụng Bayes rule, để tính likelihood, cần khối lượng tính toán  $N^T$  với N là số trạng thái có thể có và T là chiều dài chuỗi quan sát. Ví dụ trong bài toán gán nhãn có N=10 nhãn, chiều dài của chuỗi trung bình là 28, thì cần  $10^{28}$  bước tính toán. Một giải thuật với hiệu quả  $O(N^2T)$  được đề xuất với tên gọi **\*\*forward algorithm\*\***

Tài liệu tham khảo

\* <http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html> \* <https://www.isip.piconepress.com/http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html> \* <https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/section> \* <https://www.quora.com/What-is-a-simple-explanation-of-the-Hidden-Markov-Model-algorithm>

## Chương 28

# Phân loại văn bản

**Naive Bayes Classifier** Tham khảo thư viện [http://scikit-learn.org/stable/modules/naive\\_bayes.html](http://scikit-learn.org/stable/modules/naive_bayes.html) *Scikit – learn*

Xét bài toán classification với C classes  $1, 2, \dots, C$ . Tính xác suất để 1 điểm dữ liệu rơi vào class C ta có công thức:  $P(\frac{c}{x})$ . Tức tính xác suất để đầu ra là class C biết rằng đầu vào là vector x. Việc xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất:  $c = \operatorname{argmax}(P(\frac{c}{x}))$  với  $c = 1, \dots, C$  Sử dụng quy tắc Bayes:  $c = \operatorname{argmax}(P(\frac{c}{x})) = \operatorname{argmax}(P(\frac{P(\frac{c}{x})P(x)}{P(x)}) = \operatorname{argmax}(P(\frac{P(\frac{c}{x})}{P(c)}))$

**Các phân phối thường dùng** **Gaussian Naive Bayes** Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục. **Multinomial Naive Bayes** Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó. Khi đó,  $P(\frac{x_i}{c})$  tỉ lệ với tần suất từ thứ i xuất hiện trong các văn bản của class c:  $P(\frac{x_i}{c}) = \frac{N_{x_i}}{N_c}$  Trong đó:  $N_{x_i}$  là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c.  $N_c$  là tổng số từ (kể cả lặp) xuất hiện trong class c. Hay bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c. Nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức trên sẽ bằng 0, điều này dẫn đến vế phải của c bằng 0. **Bernoulli Naive Bayes** Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary. Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không. Khi đó:  $P(\frac{x_i}{c}) = P(\frac{i}{c})x_i + (1 - P(\frac{i}{c}))(1 - x_i)$  Với  $P(\frac{i}{c})$  là xác suất từ thứ i xuất hiện trong các văn bản của class c.

## Chương 29

# Pytorch

**\*\*Bí kíp luyện công\*\***

(cập nhật 08/12/2017): cảm giác [talk](http://videlectures.net/deeplearning2017\_chintala\_torch/) của anh Soumith Chintala

Sau khi nghe bài này thì hôm 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=DBVLcgq2Eg0&t=2s), Strange Loop Sep 2017 \* [Keynote: PyTorch: Framework for fast, dynamic deep learning and scientific computing](https://www.youtube.com/watch?v=LAMwEJZqesU&t=66s), EuroSciPy Aug 2017

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.whl pip install torchvision pip install torchtext”

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|grep VGA01 : 00.0VGA compatible controller : NVIDIA Corporation GM204 [GeForce GTX 980] (c) 2016 NVIDIA Corporation. Built on Sep 4 2016 at 14:01 CDT 016Cuda compilation tools, release 8.0, V8.0.44”

Kiểm tra CUDA đã cài đặt trong Ubuntu [1]

“nvcc --versionnvcc : NVIDIA(R) Cuda compiler driver Copyright (c) 2005–

2016 NVIDIA Corporation. Built on Sep 4 2016 at 14:01 CDT 016Cuda compilation tools, release 8.0, V8.0.44”

Kiểm tra pytorch chạy với cuda ‘test\_cuda.py’

“python import torch print("Cuda:", torch.cuda.is\_available())”

“`pythontest_cuda.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_available() = False`’. *Sau khi dùng `torch.Tensor().cuda()` thì gặp lỗi*

“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/to-install-latest-nvidia-drivers-in-linux>) để update NVIDIA, chưa biết kết quả ra sao?

May quá, sau khi update lên nvidia-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\\_learning\\_60min\\_blitz.h](http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html)

Hướng dẫn implement các mạng neural với pytorch rất hay tại [PyTorch-Tutorial] (<https://github.com/MorvanZhou/PyTorch-Tutorial>)

(lướt lướt) 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.

(lướt lướt) Loạt video hướng dẫn pytorch [PyTorchZeroToAll] (<https://www.youtube.com/watch?v=SKq-pmkekTkamp;list=PLlMkM4tgfjnJ3I-dbhO9JT7gNty6o2m>) *catcgiSungKimtrnyoutube*.

Bước tiếp theo là visualize loss và graph trong tensorboard, sử dụng [tensorboard\_logger] ([https://github.com/TeamHG-Memex/tensorboard\\_logger](https://github.com/TeamHG-Memex/tensorboard_logger)) *khay*.

“`pip install tensorboard_loggerpipinstalltensorboard`”

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/text-classification-cnn/>)

[<sup>1</sup>] : <https://askubuntu.com/questions/799184/how-can-i-install-cuda-on-ubuntu-16-04>

**Phần V**

**Linh tinh**

## Chương 30

# Nghiên cứu

Các công cụ

[Google Scholar](https://scholar.google.com.vn/) vẫn là lựa chọn tốt

\* Tìm kiếm tác giả theo lĩnh vực nghiên cứu và quốc gia: sử dụng filter label: + đuôi \* ví dụ: [danh sách các nhà nghiên cứu Việt Nam thuộc lĩnh vực xử lý ngôn ngữ tự nhiên (label:natural<sub>l</sub>anguage<sub>p</sub>rocessing + .vn)](https://scholar.google.com.vn/citations?hl=en&view\_op=search\_authors&mauthors=label\*danh+sach+nhua+ngheien+cuu+Viet+Nam+thuoc+linh+vuc+xu+ly+ngon+ngu+tuy+nhiên)

Bên cạnh đó còn có [semanticscholar](https://www.semanticscholar.org/) (một project của [allenai](http://allenai.org/)) với các killer features

\* [Tìm kiếm các bài báo khoa học với từ khóa và filter theo năm, tên hội nghị](https://www.semanticscholar.org/search?venue\*) [Xem những người ảnh hưởng, ảnh hưởng bởi một nhà nghiên cứu, cũng như xem co-author, journals và conferences mà một nhà nghiên cứu hay gửi bài](https://www.semanticscholar.org/author/Christopher-D-Manning/1812612)

Mendeley rất tốt cho việc quản lý và lưu trữ. Tuy nhiên điểm hạn chế lại là không lưu thông tin về citation

Các hội nghị tốt về xử lý ngôn ngữ tự nhiên

\* Rank A: ACL, EACL, NAACL, EMNLP, CoNLL \* Rank B: SemEval

Các tạp chí

\* [Computational Linguistics (CL)](http://www.mitpressjournals.org/loi/coli)

Câu chuyện của Scihub

Sci-Hub được tạo ra vào ngày 5 tháng 9 năm 2011, do nhà nghiên cứu đến từ Kazakhstan, [Alexandra Elbakyan](https://en.wikipedia.org/wiki/Alexandra\_Elbakyan)

Hãy nghe chia sẻ của cô về sự ra đời của Sci-Hub

> Khi tôi còn là một sinh viên tại Đại học Kazakhstan, tôi không có quyền truy cập vào bất kỳ tài liệu nghiên cứu. Những bài báo tôi cần cho dự án nghiên cứu của tôi. Thanh toán 32 USD thì thật là điên rồ khi bạn cần phải đọc lướt hoặc đọc hàng chục hoặc hàng trăm tờ để làm nghiên cứu. Tôi có được những bài báo như vậy vào trộm chúng. Sau đó tôi thấy có rất nhiều và rất nhiều nhà nghiên cứu (thậm chí không phải sinh viên, nhưng các nhà nghiên cứu trường đại học) giống như tôi, đặc biệt là ở các nước đang phát triển. Họ đã tạo ra các cộng đồng trực tuyến (diễn đàn) để giải quyết vấn đề này. Tôi là một thành viên tích cực trong một cộng đồng như vậy ở Nga. Ở đây ai cần có một bài nghiên cứu, nhưng không thể trả tiền cho nó, có thể đặt một yêu cầu và các thành viên

Về phần mình, là một nhà nghiên cứu trẻ, đương nhiên phải đọc liên tục. Các báo cáo ở Việt Nam về xử lý ngôn ngữ tự nhiên thì thường không tải lên các trang mở như arxiv.org, các kỷ yếu hội nghị cũng không public các proceedings. Thật sự scihub đã giúp mình rất nhiều.

Đành phải cài tor để truy cập vào scihub ở địa chỉ <http://scihub222660qcxt.onion/> <https://dl.acm.org/citation.cfm?id=3242814>  
 Và moi chuyên lai ốn.

\* Làm việc mỗi ngày \* Viết nhật ký nghiên cứu mỗi tuần (tổng kết công việc tuần trước, các ý tưởng mới, kế hoạch tuần này) \* Cập nhật các kết quả từ các hội nghị, tạp chí

\* [Machine Learning Yearning, by Andrew Ng] (<https://gallery.mailchimp.com/dc3a7ef4d750c0abfc19202a3>)

\* Review các khóa học Deep Learning: <https://www.kdnuggets.com/2017/10/3-popular-courses-deep-learning.html>

(01/11/2017) Không biết mình có phải làm nghiên cứu không nữa? Vừa  
kiêm phát triển, vừa đọc paper mỗi ngày. Thôi, cứ (miễn cưỡng) cho là nghiên  
cứu viện đi.



## Chương 31

# Nghề lập trình

Chân kinh con đường lập trình: [Teach Yourself Programming in Ten Years. Peter Norvig](<http://norvig.com/21-days.html>)

Trang web hữu ích

\* Chia sẻ thú vị: [15 năm lập trình ở Việt Nam](<https://vozforums.com/showthread.php?t=3431312>) của Blanic (vozfourm) \* Trang web chứa cheatsheet so sánh các ngôn ngữ lập trình và công nghệ [<http://hyperpolyglot.org/>](<http://hyperpolyglot.org/>)

01/11/2017

Vậy là đã vào nghề (đi làm full time trả lương) được 3 năm rưỡi rồi. Thời gian trôi qua nhanh như \*ó chạy ngoài đồng thật. Tâm đắc nhất với câu trong một quyển gì đó của anh lead HR google. Có 4 level của nghề nghiệp. 1 là thỏa mãn được yêu cầu cả bản. 2 là dự đoán được tương lai. 3 là cá nhân hóa (ý nói là tận tình với các khách hàng). 4 là phiêu diêu tự tại. Hay thật! Bao giờ mới được vào mức 4 đây.

## Chương 32

# Latex

15/12/2017:

Hôm nay tự nhiên nổi hứng vẽ hình trên latex. Thấy blog này là một guide line khá tốt về viết blog phần mềm. Quyết định cài latex

Theo [hướng dẫn này](<http://milq.github.io/install-latex-ubuntu-debian/>)

“ sudo apt-get install texlive-full sudo apt-get install texmaker “

Tìm được ngay bên này <https://www.overleaf.com/> có vẻ rất hay luôn

Hướng dẫn cực kì cơ bản <http://www.math.uni-leipzig.de/hellmund/LaTeX/pgf-tut.pdf>

Chương trình đầu tiên, vẽ diagram cho LanguageFlow

```
\documentclass[border=10pt]{standalone}
\usepackage{verbatim}
\begin{comment}
\end{comment}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
  \node[draw] (model) at (0, 0) {Model Folder};
  \node[draw] (analyze) at (6, 0) {Analyze Folder};
  \node[draw] (board) at (3,2) {Board};
  \node[draw] (logger) at (3, -2) {Logger};

  \path[->, densely dotted] (board.east)
    edge [out=0, in=90]
    node[fill=white, pos=.5] {\tiny (1) init}
    (analyze.north) ;
  \path[->, densely dotted] (board.south)
    edge [out=-90, in=180]
    node[fill=white, pos=.3] {\tiny (2) serve}
    (analyze.west) ;
  \path[->, densely dotted] (logger.west)
    edge [out=180, in=-90]
    node[fill=white, pos=.7] {\tiny (1) read}
    (model.south) ;
  \path[->, densely dotted] (logger.east)
```

```
edge [out=0, in=-90]
node[fill=white, pos=.7] {\tiny (2) write}
(analyze.south) ;
\end{tikzpicture}
\end{document}
```

Doc! Doc! Doc! <https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ>

## Chương 33

# Chào hàng

**\*\*16/01/2018\*\*** Bối cảnh. Hôm nay gửi lời mời kết bạn đến một thằng làm research về speech mà nó "chửi" mình không biết pitch. Tổ sư. Tuy nhiên, nó cũng dạy mình một bài học hay về pitch.

Chửi nó là vậy nhưng lần sau sẽ phải đầu tư nhiều hơn cho các lời pitch.

Vẫn không ưa Huyền Chíp như ngày nào, nhưng [bài này](<https://www.facebook.com/notes/huyen-chip/k>)

Tóm lại skill này có 4 phần

1. Ngôn ngữ không trau chuốt 2. Giới thiệu bản thân không tốt 3. Không chỉ ra cho người nhận rằng họ sẽ được gì 4. Không có phương án hành động

Đối với email, thì cần triển khai thể này

\* [Chào hỏi] \* [Giới thiệu bản thân một cách nào đó để người đọc quan tâm đến bạn] \* [Giải thích lý do bạn biết đến người này và bạn ấn tượng thế nào với họ – ai cũng thích được nghe khen] \* [Bạn muốn gì từ người đó và họ sẽ được gì từ việc này] \* [Kết thúc]

## Chương 34

# Phát triển phần mềm

\* Phát triển phần mềm là một việc đau khổ. Từ việc quản lý code và version, packing, documentation. Dưới đây là lược lặt những nguyên tắc cơ bản của mình.

Quản lý phiên bản

Việc đánh số phiên bản các thay đổi của phần mềm khi có hàm được thêm, lỗi được sửa, hay các phiên bản tiền phát hành cần thống nhất theo chuẩn của [semversion]. Điều này giúp nhóm có thể tương tác dễ hơn với người dùng cuối.

)

**\*\*Đánh số phiên bản\*\***

Phiên bản được đánh theo chuẩn của [semversion](<https://semver.org/>).

\* Mỗi khi một bug được sửa, phiên bản sẽ tăng lên một patch. \* Mỗi khi có một hàm mới được thêm, phiên bản sẽ tăng lên một patch. \* Khi một phiên bản mới được phát hành, phiên bản sẽ tăng lên một minor. \* Trước khi phát hành, bắt đầu với x.y.z-rc, x.y.z-rc.1, x.y.z-rc.2. Cuối cùng mới là x.y.z \* Mỗi khi phiên bản rc lỗi, khi public lại, đặt phiên bản alpha x.y.z-alpha.t (một phương án tốt hơn là cài đặt thông qua github)

**\*\*Đánh số phiên bản trên git\*\***

Ở nhánh develop, mỗi lần merge sẽ được đánh version theo PATCH, thể hiện một bug được sửa hoặc một thay đổi của hàm

Ở nhánh master, mỗi lần release sẽ được thêm các chỉ như x.y1.0-rc, x.y1.0-rc.1, x.y1.0-rc, x.y1.0

\*Vẫn còn lẩn tẩn\*:

\* Hiện tại theo workflow này thì chưa cần sử dụng alpha, beta (chắc là khi đó đã có lượt người sử dụng mới cần đến những phiên bản như thế này)

**\*\*Tải phần mềm lên pypi\*\***

Làm theo hướng dẫn [tại đây](<http://peterdowns.com/posts/first-time-with-pypi.html>)

1. Cấu hình file ‘.pypirc’ 2. Upload lên pypi

“ python setup.py sdist upload -r pypi “

## Chương 35

# Phương pháp làm việc

Xây dựng phương pháp làm việc là một điều không đơn giản. Với kinh nghiệm 3 năm làm việc, trải qua 2 project. Mà vẫn chưa produce được sản phẩm cho khách hàng. Thiết nghĩ mình nên viết phương pháp làm việc ra để xem xét lại. Có lẽ sẽ có ích cho mọi người.

Làm sao để làm việc hiệu quả, hay xây dựng phương pháp làm việc hữu ích? Câu trả lời ngắn gọn là "Một công cụ không bao giờ đủ".

<!-more->

Nội dung

1. [Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?](section1)
2. [Làm sao để quản lý project?](section2)
3. [Làm sao để công việc trôi chảy?](section3)
4. [Làm sao để xem xét lại quá trình làm việc?](section4)

<p id="section1">nbsp;</p>

Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?

Câu trả lời OKR (Objectives and Key Results)

 \*OKR Framework\*

Đầu mỗi quý , nên dành vài ngày cho việc xây dựng mục tiêu và những kết quả quan trọng cho quý tới. Cũng như review lại kết quả quý trước.

Bước 1: Xây dựng mục tiêu cá nhân (Objectives)

Bước 2: Xây dựng các Key Results cho mục tiêu này

Bước 3: Lên kế hoạch để hiện thực hóa các Key Results

<p id="section2">nbsp;</p>

Làm sao để quản lý một project

Meistertask

 \* MeisterTask\*

<p id="section3">nbsp;</p>

Làm sao để công việc trôi chảy?

Có vẻ trello là công cụ thích hợp

Bước 1: Tạo một team với một cái tên thật ấn tượng (của mình là Strong Coder)

Trong phần Description của team, nên viết Objectives and Key Results của quý này

Sau đây là một ví dụ

“ Objectives and Key Results

-> Build Vietnamese Sentiment Analysis -> Develop underthesea -> Deep Learning Book “

Bước 2: Đầu mỗi tuần, tạo một board với tên là thời gian ứng với tuần đó (của mình là ‘2017 | Fight 02 (11/12 - 16/12)‘)

Board này sẽ gồm 5 mục: "TODO", "PROGRESSING", "Early Fight", "Late Fight", "HABBIT", được lấy cảm hứng từ Kanban Board

)  
TrelloBoardexample\*

\* Mỗi khi không có việc gì làm, xem xét card trong "TODO" \* [FOCUS] tập trung làm việc trong "PROGRESSING" \* Xem xét lại thói quen làm việc với "HABBIT"

Một Card cho Trello cần có

\* Tên công việc (title) \* Độ quan trọng (thể hiện ở label xanh (chưa quan trọng), vàng (bình thường), đỏ (quan trọng)) \* Hạn chót của công việc (due date)

Sắp xếp TODO theo thứ tự độ quan trọng và Due date

<p id="section4">nbsp;</p>

Làm sao để xem xét lại quá trình làm việc?

Nhật lý làm việc hàng tuần . Việc này lên được thực hiện vào đầu tuần . Có 3 nội dung quan trọng trong nhật ký làm việc (ngoài gió mây trắng cảm xúc, quan hệ với đồng nghiệp...)

\* Kết quả công việc tuần này \* Những công việc chưa làm? Lý do tại sao chưa hoàn thành? \* Dự định cho tuần tới

\*Đang nghiên cứu\*

\*\*Làm sao để lưu lại các ý tưởng, công việc cần làm?\*\*: Dùng chức năng checklist của card trong meister. Khi có ý tưởng mới, sẽ thêm một mục trong checklist

\*\*Làm sao để tập trung vào công việc quan trọng?\*\*: Dùng chức năng tag của meister, mỗi một công việc sẽ được đánh sao (với các mức 5 sao, 3 sao, 1 sao), thể hiện mức độ quan trọng của công việc. Mỗi một sprint nên chỉ tập trung vào 10 star, một product backlog chỉ nên có 30 star.

\*\*Tài liệu của dự án\*\*: Sử dụng Google Drive, tài liệu mô tả dự án sẽ được link vào card tương ứng trong meister.

# Tài liệu tham khảo

Goodfellow, Ian / Bengio, Yoshua / Courville, Aaron (2016): *Deep Learning*. , MIT Press.

Hai, Do (2018): *Một số phân phối phổ biến* .



# Chỉ mục

convolution, 203