

Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

Mục lục

Mục lục	3
1 Lời nói đầu	4
I Lập trình	6
2 Giới thiệu	7
2.1 Các vấn đề lập trình	7
2.1.1 Introduction	7
2.1.2 Data Structure	8
2.1.3 OOP	8
2.1.4 Networking	8
2.1.5 Sample Project Ideas	8
2.2 How to ask a question	9
2.3 Các vấn đề lập trình	9
2.4 Các mô hình lập trình	9
2.5 Testing	11
2.6 Logging	13
2.7 Lập trình hàm	14
2.8 Lập trình song song	15
2.9 IDE	16
3 Python	17
3.1 Giới thiệu	17
3.2 Cài đặt	18
3.3 Cơ bản	19
3.4 Cú pháp cơ bản	19
3.5 Yield and Generators	21
3.6 Cấu trúc dữ liệu	25
3.6.1 Number	25
3.6.2 Collection	26
3.6.3 String	28
3.6.4 Datetime	29
3.6.5 Object	30
3.7 Object Oriented Programming	30
3.7.1 Metaclasses	32
3.7.2 Design Patterns	35

3.8	File System IO	36
3.9	Operating System	37
3.10	Networking	37
3.11	Concurrency and Parallelism	37
3.12	Event Based Programming	39
3.13	Web Development	40
3.14	Logging	42
3.15	Configuration	43
3.16	Command Line	43
3.17	Testing	43
3.18	IDE Debugging	44
3.19	Package Manager	46
3.20	Environment	47
3.21	Module	49
3.22	Production	51
3.23	Quản lý gói với Anaconda	51
3.24	Test với python	52
3.25	Xây dựng docs với readthedocs và sphinx	52
3.26	Pycharm Pycharm	55
3.27	Vì sao lại code python?	55
4	C++	56
4.1	Get Started	56
4.2	Basic Syntax	57
4.3	Cấu trúc dữ liệu	57
4.4	Lập trình hướng đối tượng	59
4.5	Cơ sở dữ liệu	59
4.6	Testing	60
4.7	IDE Debugging	61
5	Java	62
5.1	Get Started	62
5.2	Basic Syntax	62
5.3	Data Structure	67
5.4	OOP	67
5.4.1	Classes	67
5.4.2	Encapsulation	71
5.4.3	Inheritance	71
5.4.4	Polymorphism	73
5.4.5	Abstraction	75
5.5	File System IO	77
5.6	Error Handling	81
5.7	Logging	86
5.8	IDE	87
5.9	Package Manager	88
5.10	Build Tool	88
5.11	Production	88
6	PHP	89

<i>MỤC LỤC</i>	3
II Xác suất	91
7 Các hàm phân phối thông dụng	92
7.0.1 Biến rời rạc	92
III Khoa học máy tính	94
8 Hệ điều hành	95
9 Ubuntu	96
IV Khoa học dữ liệu	97
10 Học máy	98
11 Học sâu	101
11.1 Tài liệu Deep Learning	101
11.2 Các layer trong deep learning	101
11.2.1 Sparse Layers	101
11.2.2 Convolution Layers	101
12 Xử lý ngôn ngữ tự nhiên	103
13 Nhận dạng tiếng nói	104
14 Phân loại văn bản	108
15 Pytorch	109
V Linh tinh	111
16 Nghiên cứu	112
17 Nghề lập trình	114
18 Latex	115
19 Chào hàng	117
20 Phát triển phần mềm	118
21 Phương pháp làm việc	119
Tài liệu	121
Chỉ mục	122

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, 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, 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 Viết mark-down và latex hỗ trợ bởi wordpress Server cho phép lưu ảnh động giphy Vấn đề lưu trữ ảnh: sử dụng tính năng live upload của github.com

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, 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:

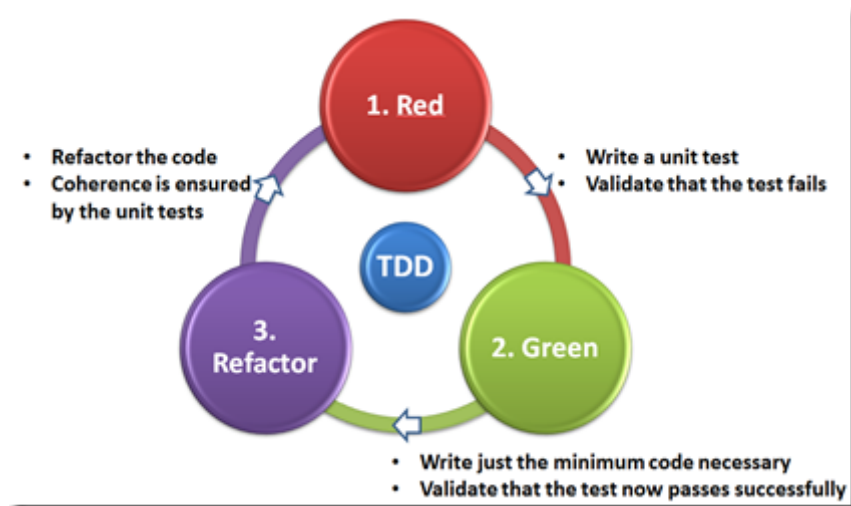
Treats programs as a model of a finite state machine or any other formal automata

Main Characteristics:

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

Examples: AsmL

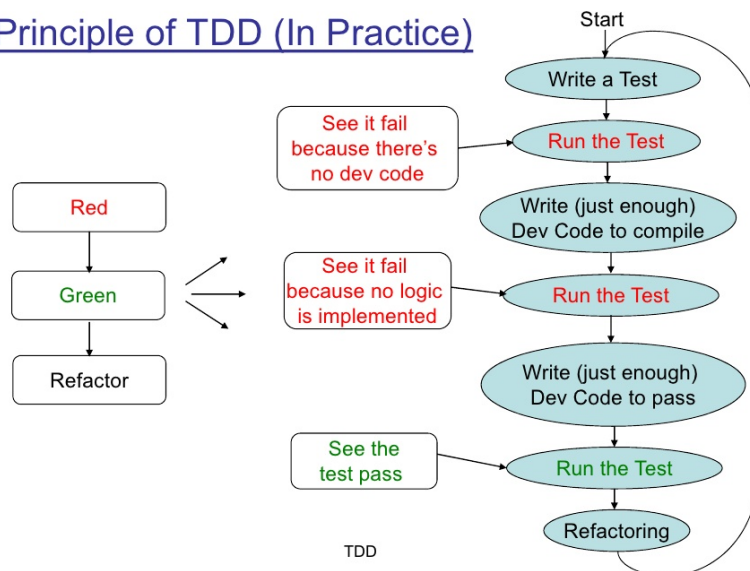
2.5 Testing



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.

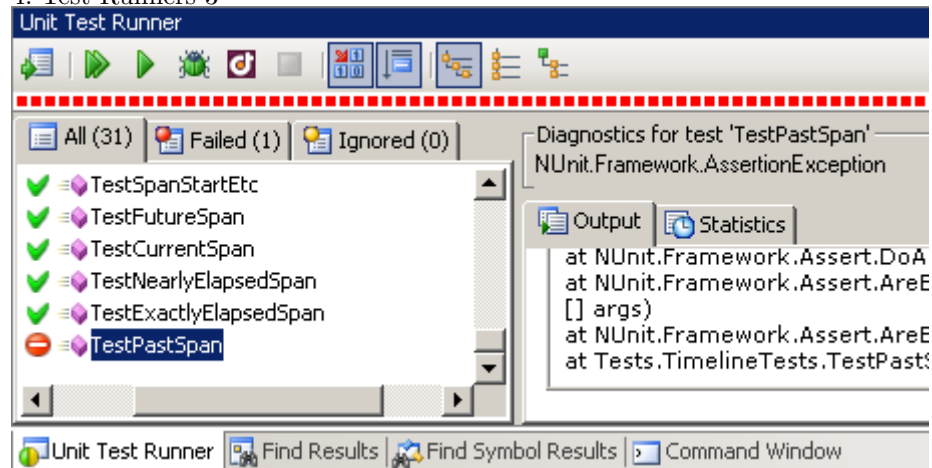
Eliminate duplication Red: (Automated test fail) Green (Automated test pass because dev code has been written) Refactor (Eliminate duplication, Clean the code)

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.

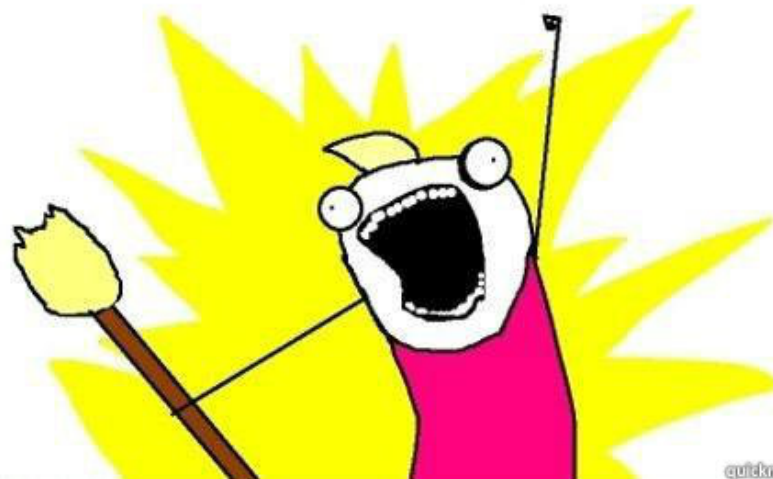
5. Test Frameworks

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

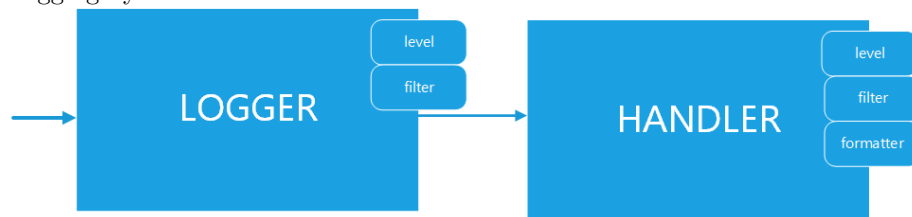
2.6 Logging

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_var*. 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 "Python Loops". www.tutorialspoint.com
 "What does the "yield" keyword do?". stackoverflow.com "Improve Your Python:
 'yield' and Generators Explained". jeffknupp.com

Vấn đề với mảng

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

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

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

to remember is, in Python, everything described here as a coroutine is still a generator. Python formally defines the term generator; coroutine is used in discussion but has no formal definition in the language.

Example: Fun With Prime Numbers Suppose our boss asks us to write a function that takes a list of ints and returns some Iterable containing the elements which are 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_pprimes* 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_pprimes* 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_pprimes*. 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?"

Imagine what we could do if `getpprimes` could simply return the next value instead of all the values at once. It would be great!

Unfortunately, this doesn't seem possible. Even if we had a magical function that allowed us to iterate from `n` to infinity, we'd get stuck after returning the first value:

```
def getpprimes(start) : for element in magical_infinite_range(start) : if ispprime(element) :
    return element
Imagine getpprimes is called like so :
def solvenumber10() : She * is * working on Project Euler 10, I knew it! total =
2 for nextpprime in getpprimes(3) : if nextpprime < 2000000 : total += nextpprime else :
print(total) return
Clearly, in getpprimes, we would immediately hit the case where number =
3 and return at line 4. Instead of return, we need a way to generate a value and, when asked for the next one, pick up where we left off.
```

Functions, though, can't do this. When they return, they're done for good. Even if we could guarantee a function would be called again, we have no way of saying, "OK, now, instead of starting at the first line like we normally do, start up where we left off at line 4." Functions have a single entry point: the first line.

Enter the Generator This sort of problem is so common that a new construct was added to Python to solve it: the generator. A generator "generates" values. Creating generators was made as straightforward as possible through the concept of generator functions, introduced simultaneously.

A generator function is defined like a normal function, but whenever it needs to generate a value, it does so with the `yield` keyword rather than `return`. If the body of a `def` contains `yield`, the function automatically becomes a generator function (even if it also contains a `return` statement). There's nothing else we need to do to create one.

generator functions create generator iterators. That's the last time you'll see the term generator iterator, though, since they're almost always referred to as "generators". Just remember that a generator is a special type of iterator. To be considered an iterator, generators must define a few methods, one of which is `next()`. To get the next value from a generator, we use the same built-in function as for iterators: `next()`.

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

(`next()` takes care of calling the generator's `next()` method). Since a generator is a type of iterator, it can be used in a `for` loop.

So whenever `next()` is called on a generator, the generator is responsible for passing back a value to whomever called `next()`. It does so by calling `yield` along with the value to be passed back (e.g. `yield 7`). The easiest way to remember what `yield` does is to think of it as `return` (plus a little magic) for generator functions.**

Again, this bears repeating: `yield` is just `return` (plus a little magic) for generator functions.

Here's a simple generator function:

```
>>> def simple_generator_function() : >>> yield 1 >>> yield 2 >>> yield 3 And here are two simple ways to use it
>>> for value in simple_generator_function() : >>> print(value) 1 2 3 >>>
our_generator = simple_generator_function() >>> next(our_generator) 1 >>>
next(our_generator) 2 >>> next(our_generator) 3 Magic? What's the magic part? Glad you asked! When a generator function is called, it returns a generator object. This object has a next() method that returns the next value. If there are no more values, a StopIteration exception is raised. This is how the for loop knows when to stop. If it weren't, the first time next() was called we would check if the number is prime and return it. If it is, we would return it. If it is not, we would return the next value. This is how the for loop knows when to stop.
```

Let's rewrite `getpprimes` as a generator function. Notice that we no longer need the magical `infinite_range` function. We can use `while True` to loop forever. If we reach the end of the definition, a `StopIteration` exception is raised. This is how the `for` loop knows when to stop. If it weren't, the first time `next()` was called we would check if the number is prime and return it. If it is, we would return it. If it is not, we would return the next value. This is how the `for` loop knows when to stop.


```

>>> 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 the value of the expression following it, and then suspends the function's execution and returns the next value when the function is next called.

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 find the next prime after a given number.

```

```

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() : dogssubfunctionThismethodreplacesalldoccurrencesoftheREpatternins
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*bc]nota, 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 RegExr: Learn, Build, Test RegEx, 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", "`

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/11110400/how-to-get-current-time-in-python) Does Python's `time.time()` return the local or UTC timestamp?, [stackoverflow](http://stackoverflow.com/questions/11110400/does-pythons-time-time-return-the-local-or-utc-timestamp) Measure time elapsed in Python?, [stackoverflow](http://stackoverflow.com/questions/11110400/measure-time-elapsed-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.a > s.b 2 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 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 initialization 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 = 'foo.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 need to modify the __new__ method of InterfaceMeta. 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)
userinterface < open file 'foo.txt', mode 'w' at 0x21b8c00 >
```

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, function 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 fewerediver' 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__':
    a=A.instance()
    b=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:
    f.write(json.dumps(data, indent=4, sort_keys=True, ensure_ascii=False))
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
{
    'menu': {
        'header': 'Sample Menu',
        'items': [
            {'id': 'Open', 'label': 'Open New', 'value': None},
            {'id': 'Help', 'label': 'About', 'value': '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 ID : ", 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.getallneweventsbytime
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'headerispresentontherequestedresource.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]', OranIPAddressthatyourDBishostedon'PORT' : '3306',
Install this module to your virtual environment
conda install mysql-python if you are using virtual environment
pip install mysql-python if you using are root environment
Custom View 8 from rest_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.HTTP201_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.yamlbuild : -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.pyfile
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/-nUnit/CppUnit series of tools.

The Basics Creating test cases is accomplished by subclassing `unittest.TestCase`.

```
import unittest
```

```
def fun(x): return x + 1
```

```
class MyTest(unittest.TestCase): def test(self): self.assertEqual(fun(3), 4)
```

Skipping tests Unittest supports skipping individual test methods and even whole classes of tests. In addition, it supports marking a test as an “expected failure,” a test that is broken and will fail, but shouldn’t be counted as a failure on a `.code` `TestResult`.

Skipping a test is simply a matter of using the `skip()` decorator or one of its conditional variants.

```
import sys import unittest
```

```
class MyTestCase(unittest.TestCase):
```

```
@unittest.skip("demonstrating skipping") def test_nothing(self) : self.fail("shouldn'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: ALT + SHIFT + UP/DOWN (move line up and down)

Graphical Debugger PyCharm provides extensive options for debugging your Python/Django and JavaScript code:

Set breakpoints right inside the editor and define hit conditions Inspect context-relevant local variables and user-defined watches, including arrays and complex objects, and edit values on the fly Set up remote debugging using remote interpreters Evaluate an expression in runtime and collect run-time type statistics for better autocompletion and code inspections Attach to a running process Debug Django templates

Inline Debugger

With an inline debugger, all live debugging data are shown directly in the editor, with variable values integrated into the editor's look-and-feel. Variable values can be viewed in the source code, right next to their usages.

Step into My Code

Use Step into My Code to stay focused on your code: the debugger will only step through your code bypassing any library sources.

Multi-process debugging

PyCharm can debug applications that spawn multiple Python processes, such as Django applications that don't run in `--no-reload` mode, or applications using many other Web frameworks that use a similar approach to code auto-reloading.

Run/Debug configurations

Every script/test or debugger execution creates a special 'Run/Debug Configuration' that can be edited and used later. Run/Debug Configurations can be shared with project settings for use by the whole team.

Workspace Custom Scheme Go to File - Settings... then Editor - Colors Fonts

Now you can change your scheme, I like Darcula

https://confluence.jetbrains.com/download/attachments/51945983/appearance3.png?version=1modificationDate=1481234567890&_af=1

IPython Support PyCharm supports usage of IPython magic commands.

<http://i.stack.imgur.com/aTEW2.png>

Vim Support You can configure PyCharm to work as a Vim editor

https://confluence.jetbrains.com/download/attachments/51946537/vim4.png?version=1modificationDate=1481234567890&_af=1

Keyboard Shortcuts: Ctrl+Shift+V (paste)

3.19 Package Manager

py2exe py2exe is a Python Distutils extension which converts Python scripts into executable Windows programs, able to run without requiring a Python installation.

Installation `py2exe conda install -c https://conda.anaconda.org/clinicalgraphics cg-py2exe Build 1 python setup.py py2exe build PyQt python setup.py py2exe --includes sip Known Issues Error: Microsoft Visual C++ 10.0 is required (Unable to find vcvarsall.bat) (link)`

How to fix

Step 1: Install Visual Studio 2015

Step 2:

set VS100COMNTOOLS=

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 didn't 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 developer's environment from their

→ `environment.yml` file:

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

```
# remove environment
```

```
conda remove -n <your_environment> --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_to_see_your_package

Step 1/4: 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[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 Step1: 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[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] :
Step2: 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
Step3: Usage your module in the same machine
import your_project_name
In other 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[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
Copy your package to a new channel
Add this line to MANIFEST.in
recursive-include project_folder *
Step2: 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
Step3: 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 đâu mà build model.

Thử build lại chục lần, thay đổi file conf.py và package_reference.rst chán chê không được. Giả thiết đầu tiên là do hai nguyên nhân (1) docstring ghi sai, (2) nội dung trong package_reference.rst bị sai. Sửa chán chê cũng vẫn thế, thử checkout các commit của git. Không hoạt động!

Mất khoảng vài tiếng mới để ý 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).

[¹] : <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](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_name like this class_data_member_name like this
//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 *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;
    sqlite3_close(db);
    cout << "Closed MyDb.db" << 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 TESTMETHODfunction
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 Config-
uration 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

Java

01/11/2017: Java đơn giản là gay nhé. Không chơi. Viết java chỉ viết thế này thôi. Không viết hơn. Thề!

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

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

5.1 Get Started

Installation Ubuntu Step 1. Download sdk

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html> Step 2. Create folder jvm

sudo mkdir /usr/lib/jvm/ Step 3. cd to folder downloads jdk and run command

```
sudo mv jdk1.7.0_x /usr/lib/jvm/jdk1.7.0_x 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
```

5.2 Basic Syntax

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

Here is a list of all primitives in Java:

byte (number, 1 byte) short (number, 2 bytes) int (number, 4 bytes) long (number, 8 bytes) float (float number, 4 bytes) double (float number, 8 bytes) char (a character, 2 bytes) boolean (true or false, 1 byte) Java is a strong typed language, which means variables need to be defined before we use them. Numbers To declare and assign a number use the following syntax:

int myNumber; myNumber = 5; Or you can combine them:

int myNumber = 5; To define a double floating point number, use the following syntax:

double d = 4.5; d = 3.0; If you want to use float, you will have to cast:

float f = (float) 4.5; Or, You can use this:

float f = 4.5f (f is a shorter way of casting float) Characters and Strings

In Java, a character is 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 = 20

a- -> 19

The Relational Operators There are following relational operators supported by Java language

== (equal to) Checks if the values of two operands are equal or not, if yes then condition becomes true.

Example: (A == B) is not true. 2 != (not equal to) Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

Example: (A != B) is true.

3 > (greater than) Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.

Example: (A > B) is not true. 4 < (less than) Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.

Example: (A < B) is true. 5 >= (greater than or equal to) Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.

Example (A >= B) is not true. 6 <= (less than or equal to) Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

example(A <= B) is true.

The Bitwise Operators Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.

Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13; now in binary format they will be as follows:

a = 0011 1100

b = 0000 1101

ab = 0000 1100

a|b = 0011 1101

a^b = 00110001

a = 1100 0011

The following table lists the bitwise operators:

Assume integer variable A holds 60 and variable B holds 13 then:

(bitwise and) Binary AND Operator copies a bit to the result if it exists in both operands.

Example: (A & B) will give 12 which is 0000 1100 2 | (bitwise or) Binary OR Operator copies a bit if it exists in either operand.

Example: (A | B) will give 61 which is 0011 1101 3 (bitwise XOR) Binary XOR Operator copies the bit if it exists in only one operand.

Example: (A ^ B) will give 49 which is 0011 0001 4 (bitwise complement) Binary Ones Complement Operator is used to convert 1 to 0 and 0 to 1.

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 straight 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 statement.
```

```
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 another way to write a one line if - else statement by using the operator `?:`

```
int a = 4; int result = a == 4 ? 1 : 8;
// result will be 1 // This is equivalent to int result;
if (a == 4) result = 1; else result = 8;
```

Again, we strongly recommend for beginners not to use this version of if.

`==` and equals The operator `==` works a bit different on objects than on primitives. When we are using objects and want to check if they are equal, the operator `==` will say if they are the same, if you want to check if they are logically equal, you should use the equals method on the object. For example:

```
String a = new String("Wow"); String b = new String("Wow"); String
sameA = a;
```

```
boolean r1 = a == b; // This is false, since a and b are not the same object
boolean r2 = a.equals(b); // This is true, since a and b are logically equals
boolean r3 = a == sameA; // This is true, since a and sameA are really the
same object
```

5.3 Data Structure

Data Structure Number, String Convert number to string

String.valueOf(1000) Make a random

// create a random number from 0 to 99 (new Random()).nextInt(100) Collection Arrays Arrays in Java are also objects. They need to be declared and then created. In order to declare a variable that will hold an array of integers, we use the following syntax:

int[] arr; Notice there is no size, since we didn't create the array yet.

arr = new int[10]; This will create a new array with the size of 10. We can check the size by printing the array's length:

System.out.println(arr.length); We can access the array and set values:

arr[0] = 4; arr[1] = arr[0] + 5; Java arrays are 0 based, which means the first element in an array is accessed at index 0 (e.g: arr[0], which accesses the first element). Also, as an example, an array of size 5 will only go up to index 4 due to it being 0 based.

int[] arr = new int[5] //accesses and sets the first element arr[0] = 4; We can also create an array with values in the same line:

int[] arr = {1, 2, 3, 4, 5}; Don't try to print the array without a loop, it will print something nasty like [I@f7e6a96.

Set

import java.util.HashSet; import java.util.Set;

public class HelloWorld

public static void main(String []args) Set<Dog> dogs = new HashSet<Dog>();

Dog dog1 = new Dog("a", 1); Dog dog2 = new Dog("a", 2); Dog dog3 = new Dog("a", 1); Dog dog4 = new Dog("b", 1); dogs.add(dog1); dogs.add(dog2); dogs.add(dog3); dogs.add(dog4); System.out.println(dogs.size());

// 3 public class Dog public String name; public int age; public int value;

public Dog(String name, int age) this.name = name; this.age = age; value = (this.name + String.valueOf(this.age)).hashCode();

@Override public int hashCode() return value;

@Override public boolean equals(Object obj) return (obj instanceof Dog ((Dog) obj).value == this.value); List<String> places = Arrays.asList("Buenos Aires", "Córdoba", "La Plata"); Datetime Calendar c = Calendar.getInstance(); Suggest Readings Initialization of an ArrayList in one line How to convert from int to String?

5.4 OOP

5.4.1 Classes

Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts

Classes and Objects Encapsulation Inheritance Polymorphism Abstraction Instance Method Message Parsing In this chapter, we will look into the concepts - Classes and Objects.

Object Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging the tail, barking, eating. An object is an instance of a class. Class A class can be defined as a template/blueprint

that describes the behavior/state that the object of its type support. Objects Let us now look deep into what are objects. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.

If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running.

If you compare the software object with a real-world object, they have very similar characteristics.

Software objects also have a state and a behavior. A software object's state is stored in fields and behavior is shown via methods.

So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

Classes A class is a blueprint from which individual objects are created.

Following is a sample of a class.

Example

```
public class Dog {
    String breed;
    int 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
```

5.4.2 Encapsulation

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

Implementation To achieve encapsulation in Java

Declare the variables of a class as private. Provide public setter and getter methods to modify and view the variables values. Example Following is an example that demonstrates how to achieve Encapsulation in Java

```
/* File name : EncapTest.java */ public class EncapTest private String
name; private String idNum; private int age;
public int getAge() return age;
public String getName() return name;
public String getIdNum() return idNum;
public void setAge( int newAge) age = newAge;
public void setName(String newName) name = newName;
public void setIdNum( String newId) idNum = newId; The public setXXX()
and getXXX() methods are the access points of the instance variables of the 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. N.p., 2016. Web. 10 Dec. 2016.

5.4.3 Inheritance

In the preceding lessons, you have seen inheritance mentioned several times. In the Java language, classes can be derived from other classes, thereby inheriting

fields and methods from those classes.

The idea of inheritance is simple but powerful: When you want to create a new class and there is already a class that includes some of the code that you want, you can derive your new class from the existing class. In doing this, you can reuse the fields and methods of the existing class without having to write (and debug!) them yourself.

A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

Class Hierarchy The `Object` class, defined in the `java.lang` package, defines and implements behavior common to all classes—including the ones that you write. In the Java platform, many classes derive directly from `Object`, other classes derive from some of those classes, and so on, forming a hierarchy of classes.

At the top of the hierarchy, `Object` is the most general of all classes. Classes near the bottom of the hierarchy provide more specialized behavior.

An Example Here is the sample code for a possible implementation of a `Bicycle` class that was presented in the *Classes and Objects* lesson:

```
public class Bicycle
// the Bicycle class has three fields public int cadence; public int gear; public
int speed;
// the Bicycle class has one constructor public Bicycle(int startCadence, int
startSpeed, int startGear) gear = startGear; cadence = startCadence; speed =
startSpeed;
// the Bicycle class has four methods public void setCadence(int newValue)
cadence = newValue;
public void setGear(int newValue) gear = newValue;
public void applyBrake(int decrement) speed -= decrement;
public void speedUp(int increment) speed += increment;
```

A class declaration for a `MountainBike` class that is a subclass of `Bicycle` might look like this:

```
public class MountainBike extends Bicycle
// the MountainBike subclass adds one field public int seatHeight;
// the MountainBike subclass has one constructor public MountainBike(int
startHeight, int startCadence, int startSpeed, int startGear) super(startCadence,
startSpeed, startGear); seatHeight = startHeight;
// the MountainBike subclass adds one method public void setHeight(int
newValue) seatHeight = newValue; MountainBike inherits all the fields and
methods of Bicycle and adds the field seatHeight and a method to set it. Except
for the constructor, it is as if you had written a new MountainBike class entirely
from scratch, with four fields and five methods. However, you didn't have to do
all the work. This would be especially valuable if the methods in the Bicycle
class were complex and had taken substantial time to debug.
```

What You Can Do in a Subclass A subclass inherits all of the public and protected members of its parent, no matter what package the subclass is in. If the subclass is in the same package as its parent, it also inherits the package-private members of the parent. You can use the inherited members as is, replace them, hide them, or supplement them with new members:

The inherited fields can be used directly, just like any other fields. You can declare a field in the subclass with the same name as the one in the superclass, thus hiding it (not * recommended). You can declare new fields in the subclass that are not in the superclass. The inherited methods can be used directly as they are. You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it. You can write a new static method in the subclass that has the same signature as the one in the superclass, thus hiding it. You can declare new methods in the subclass that are not in the superclass. You can write a subclass constructor that invokes the constructor of the superclass, either implicitly or by using the keyword `super`. The following sections in this lesson will expand on these topics.

Private Members in a Superclass A subclass does not inherit the private members of its parent class. However, if the superclass has public or protected methods for accessing its private fields, these can also be used by the subclass.

A nested class has access to all the private members of its enclosing class—both fields and methods. Therefore, a public or protected nested class inherited by a subclass has indirect access to all of the private members of the superclass.

Casting Objects We have seen that an object is of the data type of the class from which it was instantiated. For example, if we write

```
public MountainBike myBike = new MountainBike();
```

then `myBike` is of type `MountainBike`.

`MountainBike` is descended from `Bicycle` and `Object`. Therefore, a `MountainBike` is a `Bicycle` and is also an `Object`, and it can be used wherever `Bicycle` or `Object` objects are called for.

The reverse is not necessarily true: a `Bicycle` may be a `MountainBike`, but it isn't necessarily. Similarly, an `Object` may be a `Bicycle` or a `MountainBike`, but it isn't necessarily.

Casting shows the use of an object of one type in place of another type, among the objects permitted by inheritance and implementations. For example, if we write

```
Object obj = new MountainBike();
```

then `obj` is both an `Object` and a `MountainBike` (until such time as `obj` is assigned another object that is not a `MountainBike`). This is called implicit casting.

If, on the other hand, we write

```
MountainBike myBike = obj;
```

we would get a compile-time error because `obj` is not known to the compiler to be a `MountainBike`. However, we can tell the compiler that we promise to assign a `MountainBike` to `obj` by explicit casting:

```
MountainBike myBike = (MountainBike)obj;
```

This cast inserts a runtime check that `obj` is assigned a `MountainBike` so that the compiler can safely assume that `obj` is a `MountainBike`. If `obj` is not a `MountainBike` at runtime, an exception will be thrown.

Related Readings "Inheritance". docs.oracle.com. N.p., 2016. Web. 8 Dec. 2016. "Java Inheritance". www.tutorialspoint.com. N.p., 2016. Web. 8 Dec. 2016. Friesen, Jeff. "Java 101: Inheritance In Java, Part 1". JavaWorld. N.p., 2016. Web. 8 Dec. 2016.

5.4.4 Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is

used to refer to a child class object.

Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object.

It is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared, the type of a reference variable cannot be changed.

The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.

A reference variable can refer to any object of its declared type or any 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. N.p., 2016.
Web. 10 Dec. 2016.

5.4.5 Abstraction

As per dictionary, abstraction is the quality of dealing with ideas rather than
events. For example, when you consider the case of e-mail, complex details such
as what happens as soon as you send an e-mail, the protocol your e-mail server
uses are hidden from the user. Therefore, to send an e-mail you just need to
type the content, mention the address of the receiver, and click send.

Likewise in Object-oriented programming, abstraction is a process of hiding
the implementation details from the user, only the functionality will be provided
to the user. In other words, the user will have the information on what the object
does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

Abstract Class A class which contains the abstract keyword in its declaration
is known as abstract class.

Abstract classes may or may not contain abstract methods, i.e., methods without body (`public void get();`) But, if a class has at least one abstract method, then the class must be declared abstract. If a class is declared abstract, it cannot be instantiated. To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it. If you inherit an abstract class, you have to provide implementations to all the abstract methods in it. Example

This section provides you an example of the abstract class. To create an abstract class, just use the abstract keyword before the class keyword, in the class declaration.

```
/* File name : Employee.java */ public abstract class Employee private
String name; private String address; private int number;
public Employee(String name, String address, int number) System.out.println("Constructing
an Employee"); this.name = name; this.address = address; this.number = 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); ¹*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. N.p., 2016. Web. 10 Dec. 2016.

5.5 File System IO

The `java.io` package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the `java.io` package supports many data such as primitives, object, localized characters, etc.

Stream A stream can be defined as a sequence of data. There are two kinds of Streams

InputStream The `InputStream` is used to read data from a source. **OutputStream** The `OutputStream` is used for writing data to a destination.

Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionality related to streams and I/O. We will see the most commonly used examples one by one

Byte Streams Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, `FileInputStream` and `FileOutputStream`. Following is an example which makes use of these two classes to copy an input file into an output file

Example

```
import java.io.*; public class CopyFile
public static void main(String args[]) throws IOException {
    FileInputStream in = null; FileOutputStream out = null;
    try { in = new FileInputStream("input.txt"); out = new FileOutputStream("output.txt");
        int c; while ((c = in.read()) != -1) out.write(c); finally { if (in != null)
            in.close(); if (out != null) out.close(); }
    } Now let's have a file input.txt with
    the following content
```

This is test for copy file. As a next step, compile the above program and execute it, which will result in creating `output.txt` file with the same content as we have in `input.txt`. So let's put the above code in `CopyFile.java` file and do the following

```
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.
```

5.6 Error Handling

An exception (or exceptional event) is a problem that arises during the execution of a program. When an `Exception` occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

A user has entered an invalid data. A file that needs to be opened cannot be found. A network connection has been lost in the middle of communications or the JVM has run out of memory. Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Based on these, we have three categories of Exceptions. You need to understand them to know how exception handling works in Java.

Type of exceptions Checked Exception

A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be

ignored at the time of compilation, the programmer should take care of (handle) these exceptions.

For example, if you use `FileReader` class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a `FileNotFoundException` occurs, and the compiler prompts the programmer to handle the exception.

```
import java.io.File; import java.io.FileReader;
public class 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 :
unreportedexceptionFileNotFoundException; 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.

5.7 Logging

Log4j log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License. log4j is a popular logging package written in Java. log4j has been ported to the C, C++, C, Perl, Python, Ruby, and Eiffel languages.

log4j is highly configurable through external configuration files at runtime. It views the logging process in terms of levels of priorities and offers mechanisms to direct logging information to a great variety of destinations, such as a database, file, console, UNIX Syslog, etc.

log4j has three main components:

loggers: Responsible for capturing logging information. appenders: Responsible for publishing logging information to various preferred destinations. layouts: Responsible for formatting logging information in different styles. log4j features

It is thread-safe. It is optimized for speed. It is based on a named logger hierarchy. It supports multiple output appenders per logger. It supports internationalization. It is not restricted to a predefined set of facilities. Logging behavior can be set at runtime using a configuration file. It is designed to handle Java Exceptions from the start. It uses multiple levels, namely ALL, TRACE, DEBUG, INFO, WARN, ERROR and FATAL. The format of the log output can be easily changed by extending the Layout class. The target of the log output as well as the writing strategy can be altered by implementations of the Appender interface. It is fail-stop. However, although it certainly strives to ensure delivery, log4j does not guarantee that each log statement will be delivered to its destination. Example Step 1: Add log4j dependency to your build.gradle file
 compile group: 'log4j', name: 'log4j', version: '1.2.17' Step 2: Add log configuration in main/resources/log4j.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=Ste
```

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>.

5.8 IDE

Java: IDE IntelliJ 1. Project Manager 2. Search Replace 3. Navigation 4. Formatting 5. Debugging 6. Build Release 7. Git Integration 1. Project Manager 1.1 Create New Project

1.2 Import Maven Project

<https://www.jetbrains.com/help/idea/2016.1/importing-project-from-maven-model.html>

2. Search Replace Global Search Shift Shift 3. Navigation Next/Previous Error F2 / Shift + F2 4. Formatting Auto Format Ctrl + Alt + L

5.9 Package Manager

Java: Package Manager Gradle

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

`mkdir gradle_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"`

5.10 Build Tool

Java: Build Tool Apache Ant

Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks. 1

Install Ant Download and extract Apache Ant 1.9.6

`wget http://mirrors.viethosting.vn/apache//ant/binaries/apache-ant-1.9.6-bin.tar.gz` `tar -xzf apache-ant-1.9.6-bin.tar.gz` Set path to ant folder

Build Ant through proxy Requirement: 1.9.5+

Add the following lines into build.xml

```
<target name="ivy-init" depends="ivy-proxy, ivy-probe-antlib, ivy-init-antlib"
description="-> initialise Ivy settings"> <ivy:settings file="ivy.dir/ivysettings.xml" / ><
/target >< targetname = "ivy-proxy" description = "--> ProxyIvysettings" ><
propertyname = "proxy.host" value = "proxy.com" / >< propertyname =
"proxy.port" value = "8080" / >< propertyname = "proxy.user" value = "user" / ><
propertyname = "proxy.password" value = "password" / >< setproxyproxyhost =
"proxy.host" proxyport="proxy.port" proxyuser = "proxy.user" proxypassword="proxy.password" / ><
/target > ApacheAnt™
```

5.11 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 6

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

Phần II

Xác suất

Chương 7

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 ¹

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

7.0.1 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

¹http://www.ctec.tvu.edu.vn/ttkhai/xacsuatthongke_dh.htm

²<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 8

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 : fpuvmedepsetscmsrpaemccecx8apicsepmttrrpgemcacrmoovpatpse36clflushdtsacpimxfxsrssesse

5005.20clflushsize : 64cache_alignment : 64addresssizes : 46bitsphysical, 48bitvirtualpowermanagement :

“

Kết quả cho thấy cpu của 6 core và 12 siblings

Chương 9

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 ^[1] “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.

^[1] : <https://askubuntu.com/questions/389903/ibus-doesnt-seem-to-restart>

Phần IV

Khoa học dữ liệu

Chương 10

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).

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>). *CÓ CP ON*

* [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>) đỡ 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/IT23246_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)*camtanhMicrosoft.TrongO, (Ongnhin)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))*grepcode* : [Shawn1993/cnn-text-classification-pytorch](<https://github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.py#L18>)*(Engvaitrnhmtlookuptable, mapmtwordvidenseve*

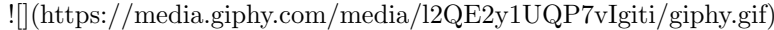
Convolution Layers

****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_size` $thngbngregion_size * word_dim$ (*Oiviconv1d*) *hay* (`region_size, word_dim`) *Oiviconv2d*

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://d3kbpzmbcynnmix.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' đối với 'nn.conv1d'

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

 (<http://d3kbpzmbcynnmix.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
latexlamp; s = 22 norm constraint	3

Đọc thêm:

* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017] (<http://web.stanford.edu/class/cs224n-2017-lecture13-CNNs.pdf>) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks] (<https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb>) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015] (<https://arxiv.org/pdf/1510.03820.pdf>)

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)

[¹] : [UnderstandingConvolutionalNeuralNetworksforNLP] (<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp>) [²] : [<http://pytorch.org/docs/master/nn.html>] (<http://pytorch.org/docs/master/nn.html>)

Chương 11

Học sâu

11.1 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.

11.2 Các layer trong deep learning

11.2.1 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

11.2.2 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_{dim}' đối với 'nn.conv1d'

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

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>l₂</i> 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)

[¹] : [UnderstandingConvolutionalNeuralNetworksforNLP](<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp>)[²] : <http://pytorch.org/docs/master/nn.html>

Chương 12

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 đây 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/), *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 13

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)³ *triphone, nhngchcmttpthcstntitrongdliu.Hnna, ccQnvxyr*

****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.

Dòng 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.

Dòng 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]_{in sequence}

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 14

Phân loại văn bản

Naive Bayes Classifier Tham khảo thư viện 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: $\text{latex}P(\frac{c}{x})$. 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: $c = \text{argmax}(\text{latex}P(\frac{c}{x}))$ với $c = 1, \dots, C$ Sử dụng quy tắc Bayes: $c = \text{argmax}(\text{latex}P(\frac{c}{x})) = \text{argmax}(\text{latex}P(\frac{P(\frac{c}{x})P(x)}{P(x)}) = \text{argmax}(\text{latex}P(\frac{P(\frac{c}{x})}{P(c)}))$

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, $\text{latex}P(\frac{x_i}{c})$ tỉ lệ với tần suất từ thứ i xuất hiện trong các văn bản của class c : $\text{latex}P(\frac{x_i}{c}) = \text{latex}\frac{N_{x_i}}{N_c}$ Trong đó: $\text{latex}N_{x_i}$ 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 . $\text{latex}N_c$ 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: $\text{latex}P(\frac{x_i}{c}) = \text{latex}P(\frac{i}{c})x_i + (1 - \text{latex}P(\frac{i}{c}))(1 - \text{latex}x_i)$ Với $\text{latex}P(\frac{i}{c})$ là xác suất từ thứ i xuất hiện trong các văn bản của class c .

Chương 15

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|grepVGA01 : 00.0VGAcompatiblecontroller : NVIDIA Corporation GM204 [GeForce GTX 980] (a”

Kiểm tra CUDA đã cài đặt trong Ubuntu [1]

“nvcc --versionnvcc : NVIDIA(R)Cuda compiler driver Copyright (c) 2005–

2016 NVIDIA Corporation, Builton Sun Sep 4 2:14:01 CDT 2016 Cuda 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 gpl*

“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) *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/>)

[¹] : <https://askubuntu.com/questions/799184/how-can-i-install-cuda-on-ubuntu-16-04>

Phần V

Linh tinh

Chương 16

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_language_processing + .vn)](https://scholar.google.com.vn/citations?hl=en&view_op=search_authors&mauthors=label*danhschny0spxptheolngtrchdn)

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.

Vào thời điểm này (12/2017), scihub bị chặn quyết liệt. Hóng được trên page facebook của scihub các cách truy cập scihub. Đã thử các domain khác như .tw, .hk. Mọi chuyện vẫn ổn cho đến hôm nay (21/12/2017), không thể truy cập vào nữa.

Làm sao để nghiên cứu tốt

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 17

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/>

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 18

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 19

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 20

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 21

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 trăng 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, 101