



上海外国语大学
SHANGHAI INTERNATIONAL STUDIES UNIVERSITY

Computational Journalism

Lecture 2: Software for Programming

Ting Wang

Outlines

- The Process of Software Development
- Python Programming Environment





a management approach to software engineering

The Process of Software Development

The Process of Software Development

Software Crisis

- The First NATO Software Engineering Conference in 1968, Germany.
- How to cope with the difficulty of writing useful and efficient computer programs in the required time.



The Process of Software Development

Difficulties in Software Development

1. Projects running over-budget
2. Projects running over-time
3. Software was very inefficient
4. Software was of low quality
5. Software often did not meet requirements
6. Projects were unmanageable and code difficult to maintain
7. Software was never delivered



The Process of Software Development

What is Software Engineering

- Software engineering is the application of engineering to the **design, development, implementation, testing and maintenance** of software in a systematic method.

From Wikipedia

The Process of Software Development

Basic Elements in Software Engineering

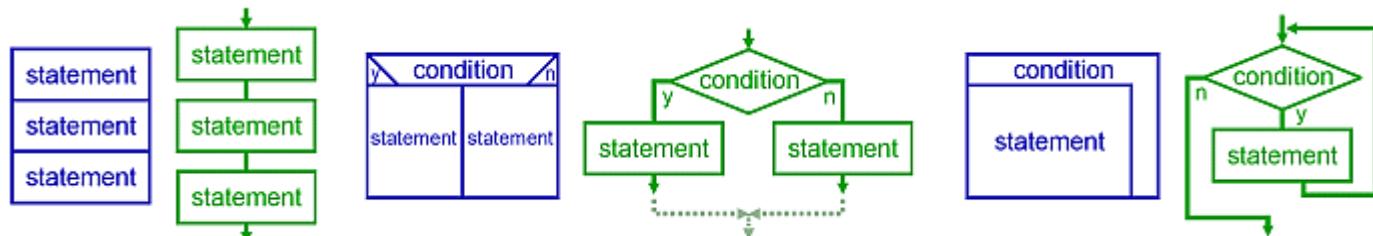
- Development Stages
- Management Pipeline
- Demands Changing
- Cooperative Team Work
- Professional Expert Participation



The Process of Software Development

Programming Paradigm 编程范式

- Structured Programming (1) 结构化编程
 - Control Structure
 - 1. Sequence
 - 2. Selection: *if..then..else..endif, switch*
 - 3. Iteration: *while, repeat, for, do...until*

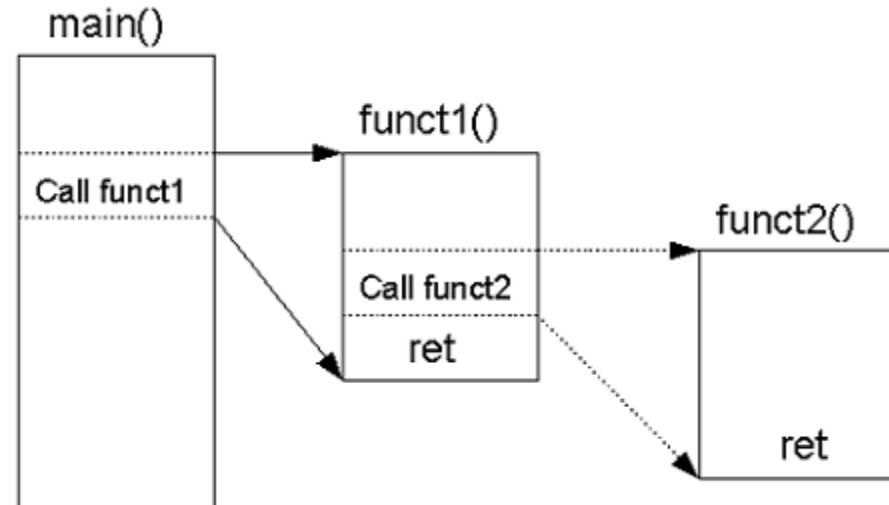


The Process of Software Development

Programming Paradigm 编程范式

- Structured Programming (2) 结构化编程
 - Subroutines

子程序



The Process of Software Development

Programming Paradigm 编程范式

- Object Oriented Programming(1)

面向对象编程 针对物件的编程

- Object 对象
- Class 类
- Attribute 属性
- Method 方法



The Process of Software Development

Programming Paradigm 编程范式

- Object Oriented Programming(2)

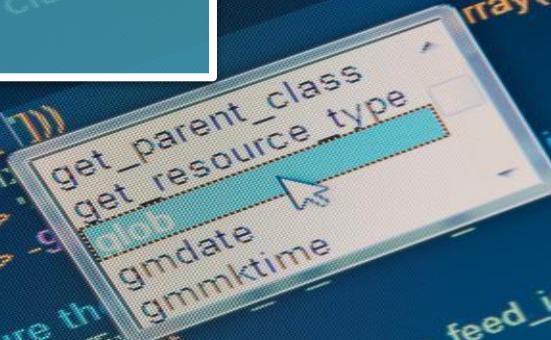
面向对象编程

- Encapsulation 封装
- Inheritance 继承
- Polymorphism 多态



The Process of Software Development

EXAMPLE 1: Class



The Process of Software Development

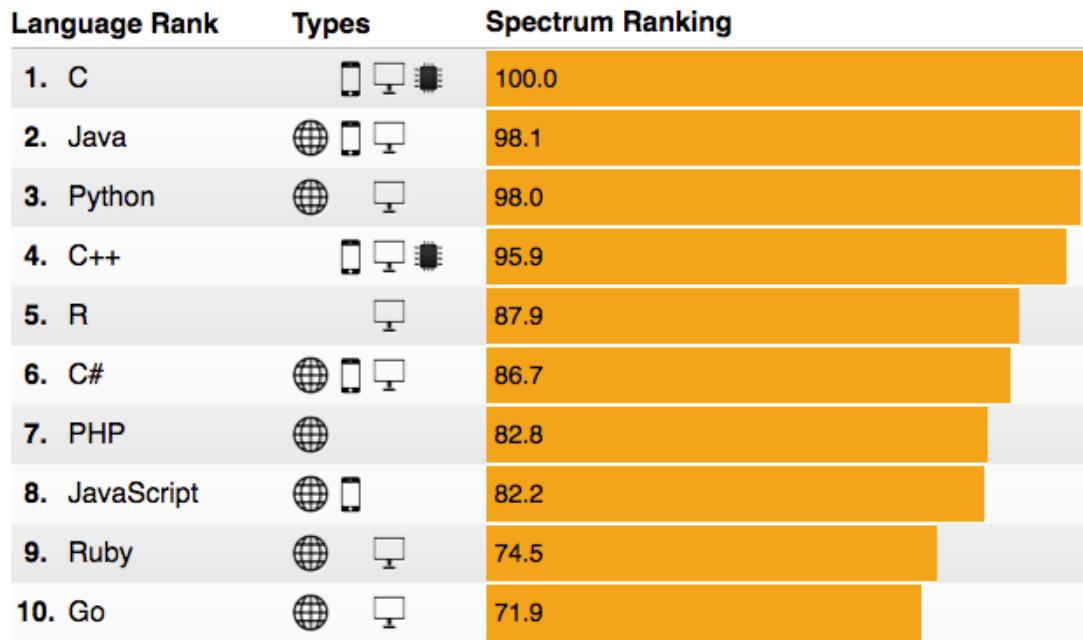
```
1  class Customer(object):
2      ...
3      name =''
4      password =''
5
6      def __init__(self, name, password):
7          self.__name = name
8          self.__password = password
9          print ('Name: %s' %self.__name)
10
11     def get_validation(self, password):
12         if password=='sisu':
13             return 1
14         else:
15             return 0
16
17     class Student(Customer):
18
19         name =''
20         password =''
21         studentID =''
22
23         def __init__(self, name, password, studentID):
24             Customer.__init__(self, name, password)
25             self.studentID = studentID
26
27         def print_studentID(self):
28             return self.studentID
29
30         def get_validation(self, password):
31             if password=='shisu':
32                 return 'Passed'
33             else:
34                 return 'Failed'
35
36 Thomas = Customer('Thomas Edison', 'sisu')
37 print('Thomas.get_validation() =', Thomas.get_validation('sisu'))
38 Albert = Student('Albert Einstein', 'sisu', '20160001')
39 print('Albert.print_studentID() =', Albert.print_studentID())
40 print('Albert.get_validation() =', Albert.get_validation('sisu'))
41
```



The Process of Software Development

IEEE Spectrum

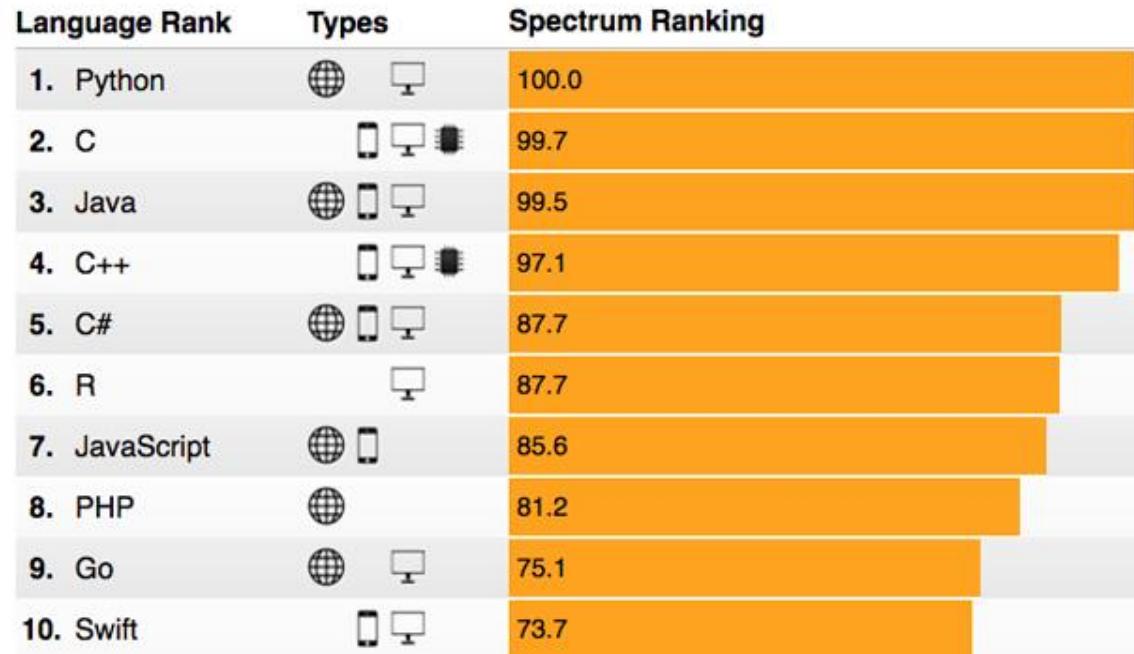
<http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages>



The Process of Software Development

IEEE Spectrum

<https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>



The Process of Software Development

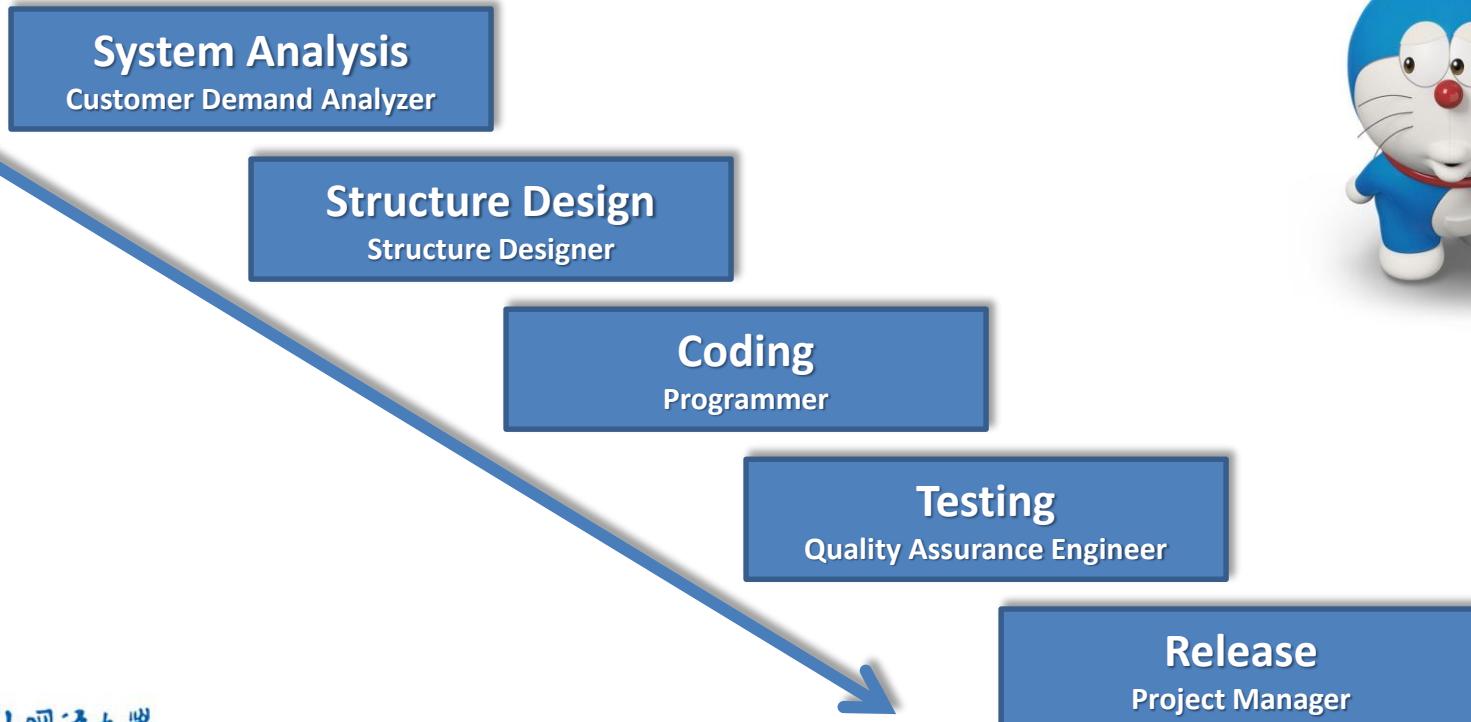
IEEE Spectrum

<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>

Language Rank	Types	Spectrum Ranking
1. Python	🌐💻胞	100.0
2. C++	📱💻胞	99.7
3. Java	🌐📱💻	97.5
4. C	📱💻胞	96.7
5. C#	🌐📱💻	89.4
6. PHP	🌐	84.9
7. R	💻	82.9
8. JavaScript	🌐📱	82.6
9. Go	🌐💻	76.4
10. Assembly	胞	74.1

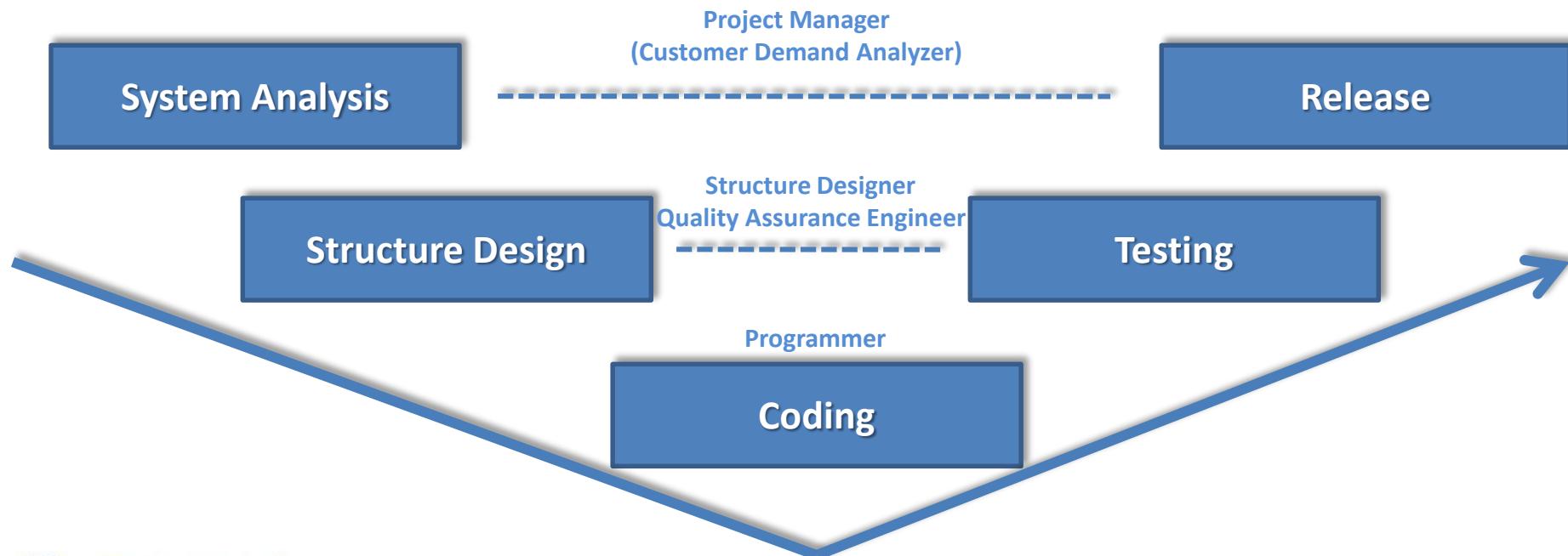
The Process of Software Development

- Water Fall Model



The Process of Software Development

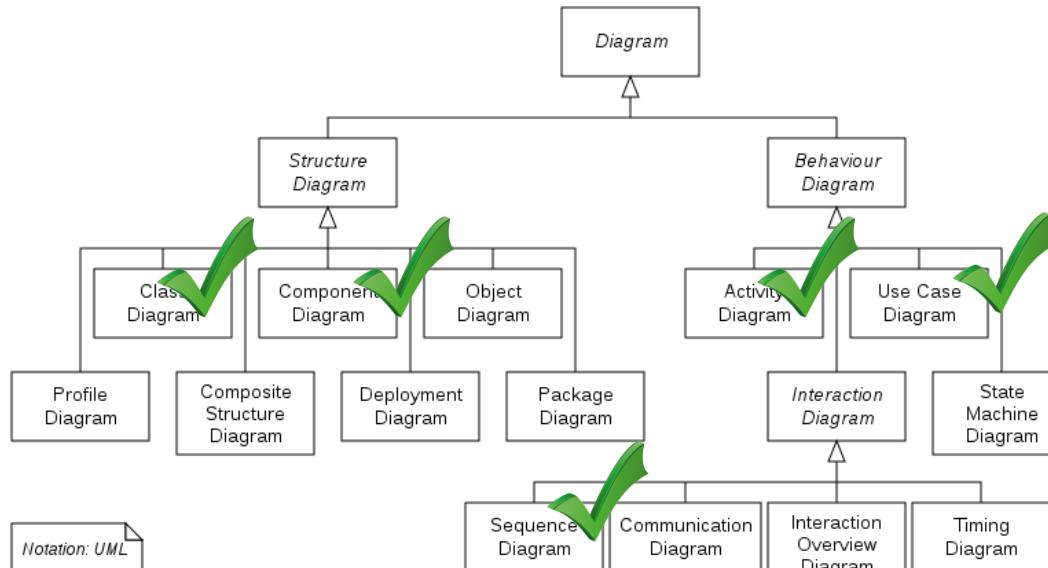
- V-Model



The Process of Software Development

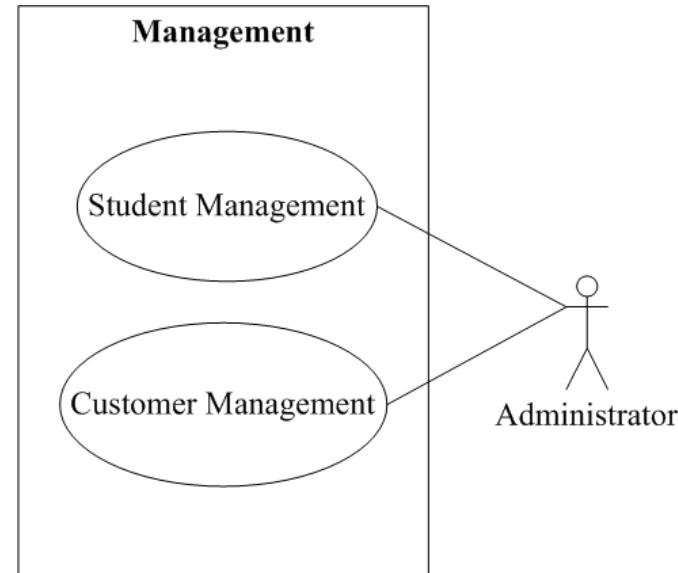
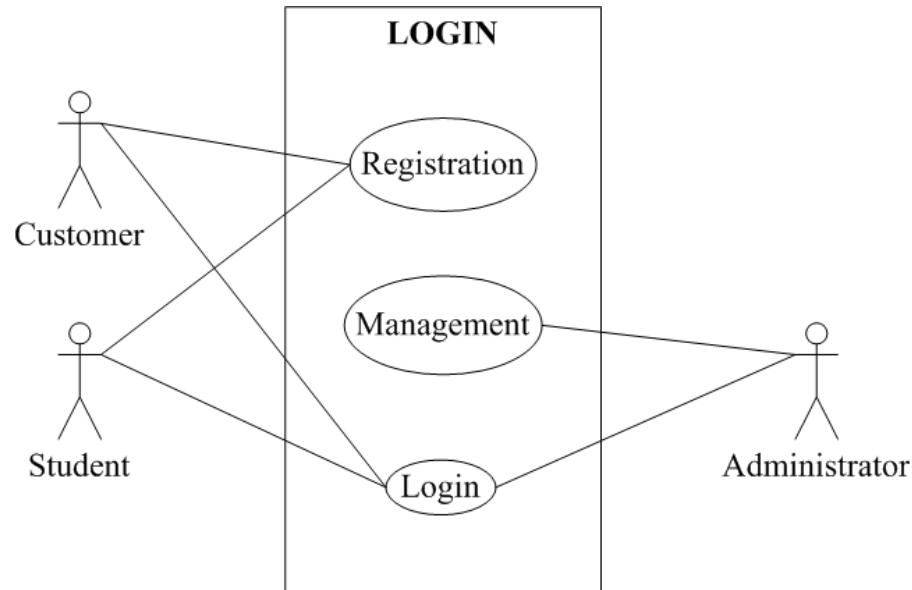
Unified Modeling Language (UML)

A general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system



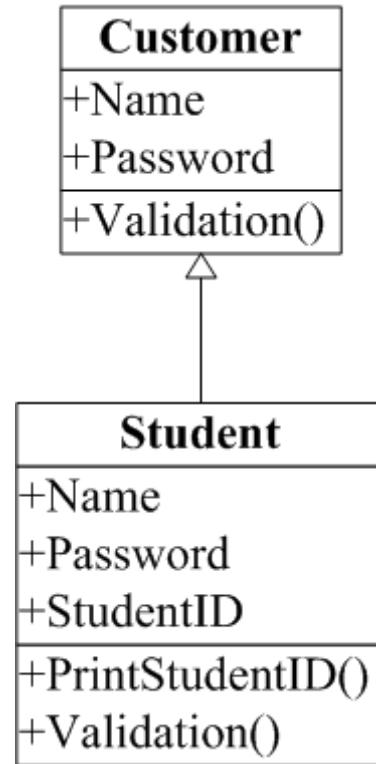
The Process of Software Development

- Use Case Diagram



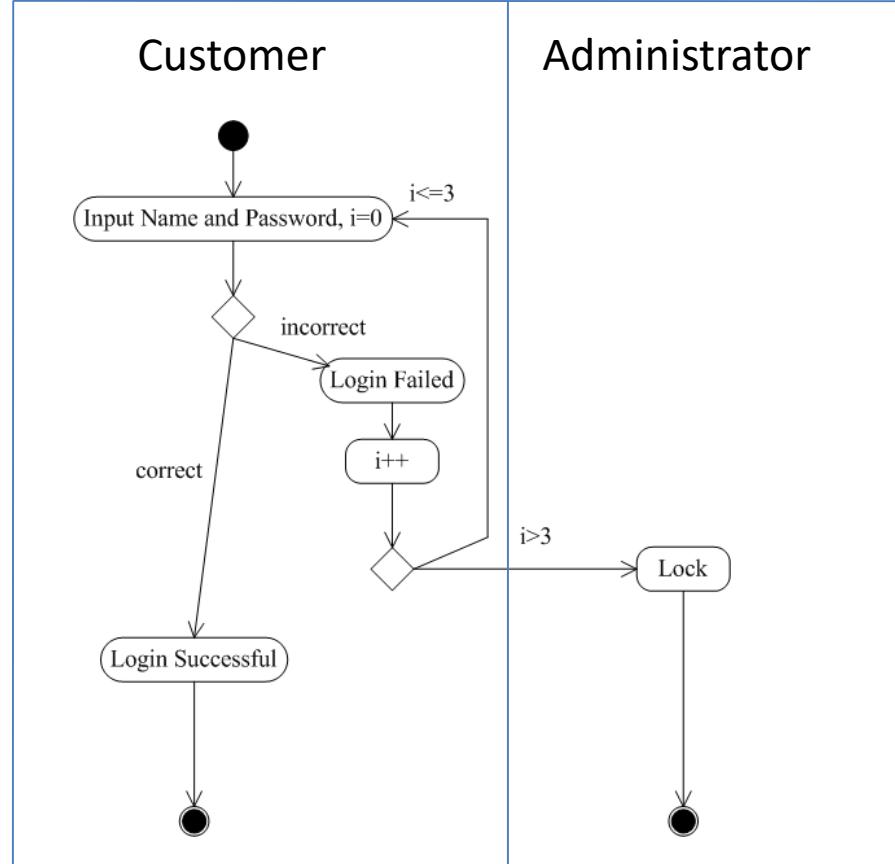
The Process of Software Development

- Class Diagram



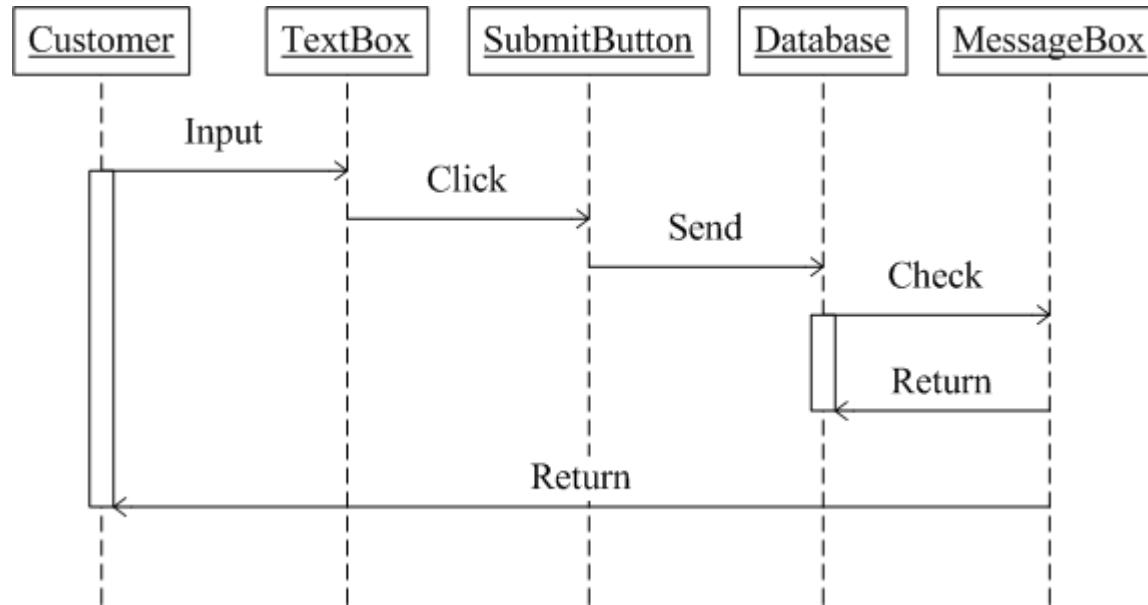
The Process of Software Development

- Activity Diagram



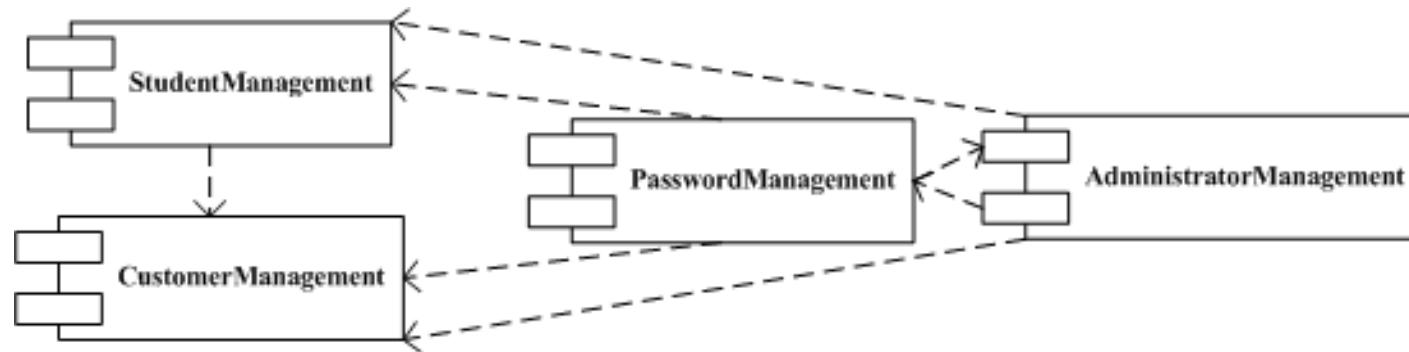
The Process of Software Development

- Sequence Diagram



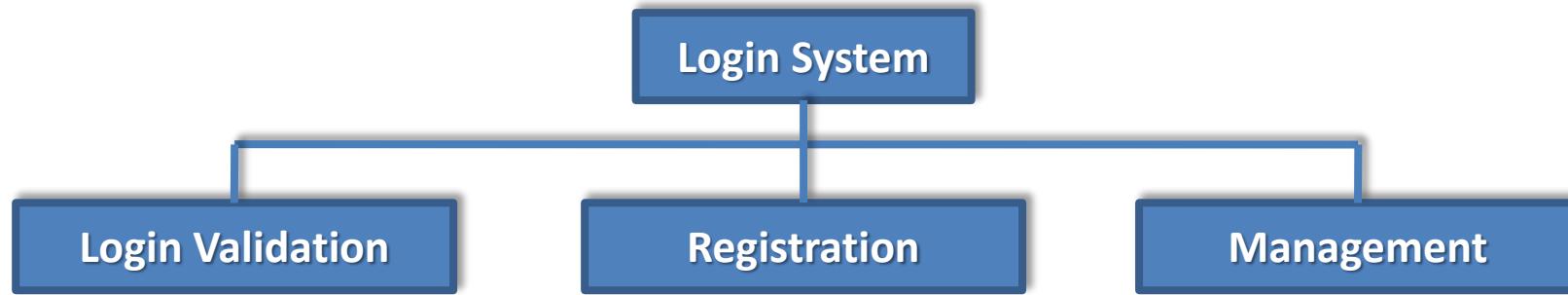
The Process of Software Development

- Component Diagram



The Process of Software Development

- Function Structure Diagram



The Process of Software Development

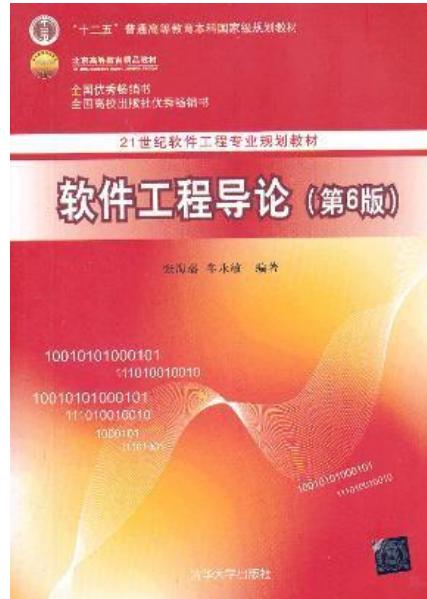
Time Estimation for Software Projects

- Man-Month
- Man-Day
- Basic Function: Insert, Delete, Update, Select
 - Slow: *1 Basic Function per day*
 - Common: *2 Basic Functions per day*
 - Fast: *4 Basic Functions per day*

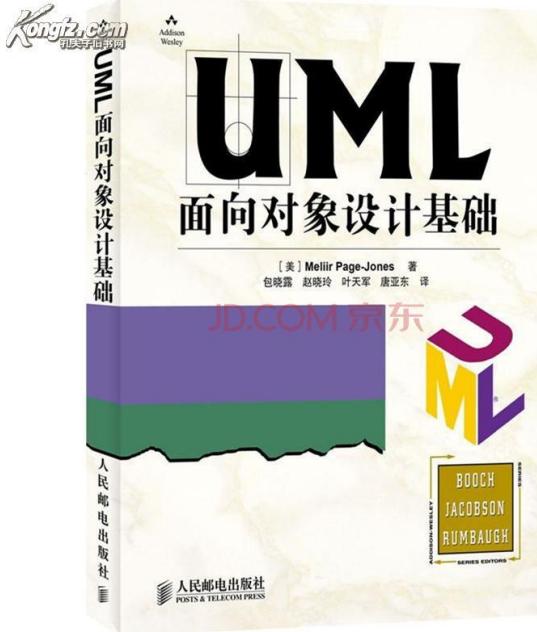
The Process of Software Development

References

软件工程导论（第6版）



UML面向对象设计基础





上海外国语大学
SHANGHAI INTERNATIONAL STUDIES UNIVERSITY

start my programming

Python Programming Environment

Outlines

- Installation
- Grammar
- Functions
- Debugging





上海外国语大学
SHANGHAI INTERNATIONAL STUDIES UNIVERSITY

to build an environment for running python

Installation

Installation



Guido van Rossum

The Birth of Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language designed by Guido van Rossum in 1991.



Installation

Official Website of Python (<https://www.python.org/>)

The screenshot shows the official Python website homepage. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. The main content area features the Python logo and a search bar. Below the search bar is a horizontal menu with links for About, Downloads, Documentation, Community, Success Stories, News, and Events. A central feature is a code editor window displaying a Python script for generating a Fibonacci series up to n=1000. To the right of the code is a section titled "Functions Defined" with text about Python's function definition capabilities and a link to "More about defining functions in Python 3". At the bottom of the page is a footer with a university logo and the text: "Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)".

Python 3: Fibonacci series up to n

```
>>> def fib(n):  
>>>     a, b = 0, 1  
>>>     while a < n:  
>>>         print(a, end=' ')  
>>>         a, b = b, a+b  
>>>     print()  
>>> fib(1000)  
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

Python 2.X or 3.X?

Python 2.x is legacy, Python 3.x is the present and future of the language

Python 2.x is old, but mature;

Python 3.x is new, but slow.

Download the latest version for Windows

[Download Python 3.5.2](#)

[Download Python 2.7.12](#)

<https://wiki.python.org/moin/Python2orPython3>

Installation

Download the Installation Package

<https://www.python.org/downloads/>

Python Releases for Windows

- [Latest Python 2 Release - Python 2.7.12](#)
- [Latest Python 3 Release - Python 3.5.2](#)

Python Releases for Mac OS X

- [Latest Python 2 Release - Python 2.7.12](#)
- [Latest Python 3 Release - Python 3.5.2](#)

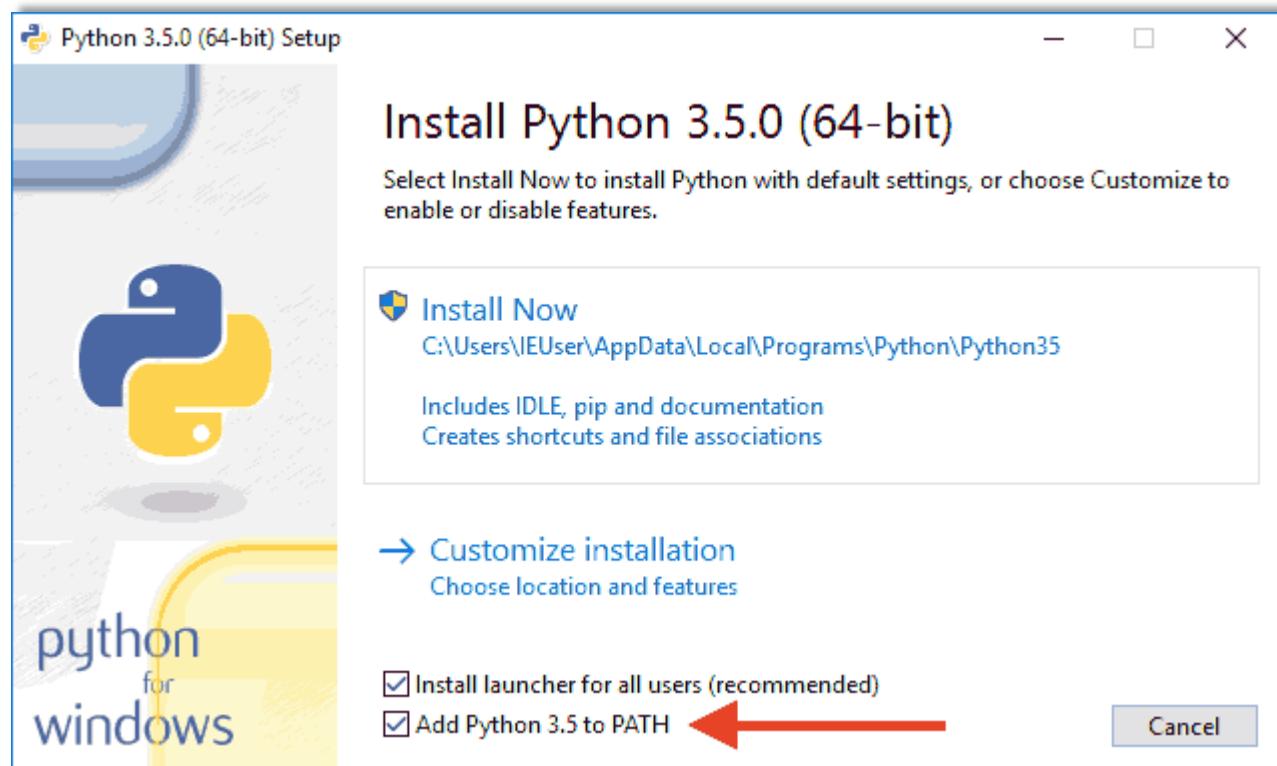
Python Source Releases

- [Latest Python 2 Release - Python 2.7.12](#)
- [Latest Python 3 Release - Python 3.5.2](#)



Installation

Next Step



Installation

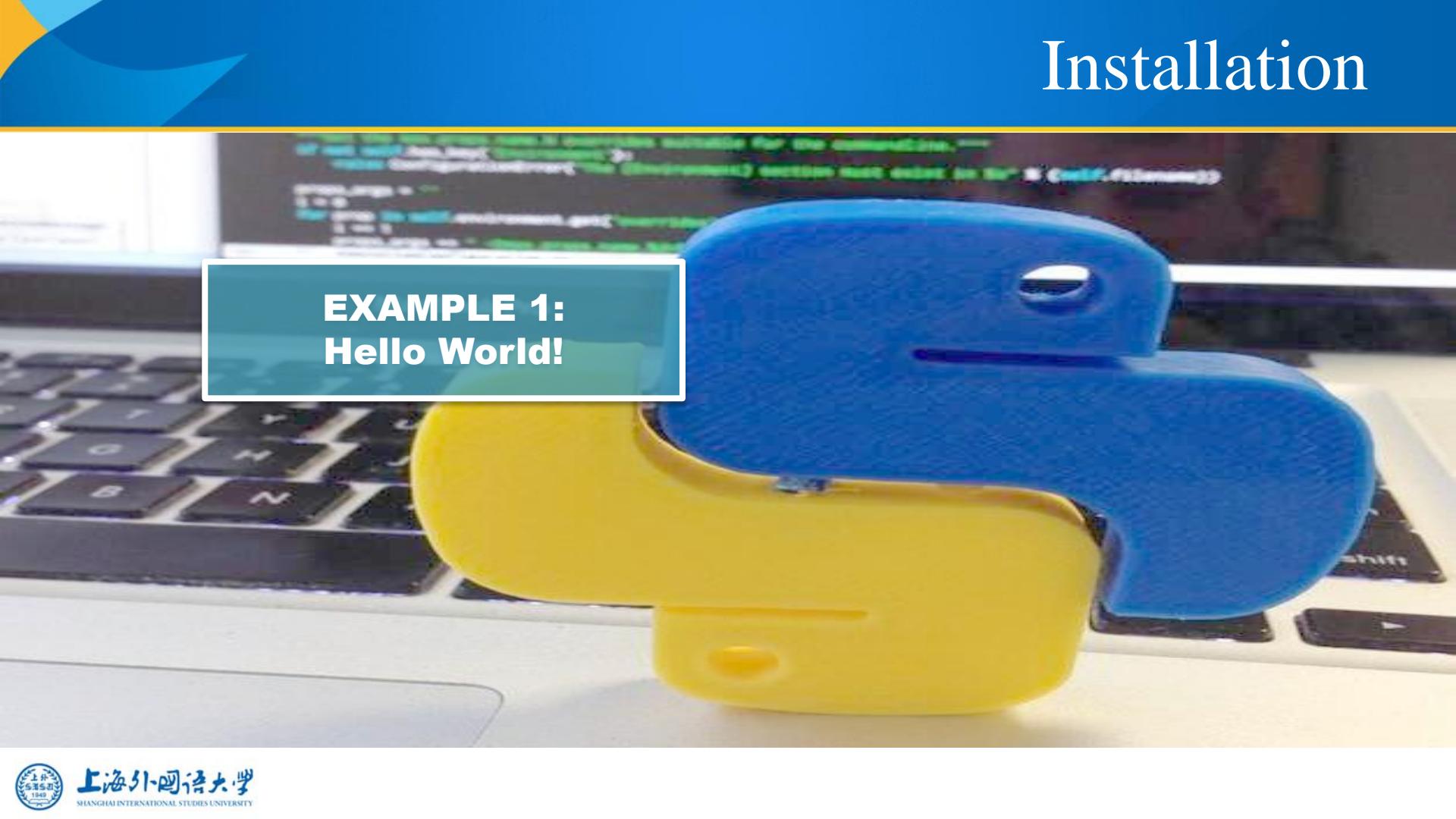
Success !

Start menu:



A screenshot of a Windows Command Prompt window titled "管理员: C:\Windows\system32\cmd.exe - python". The window displays the following text:
Microsoft Windows [版本 6.1.7601]
版权所有 © 2009 Microsoft Corporation。保留所有权利。
C:\Users\Ting>python
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

Installation

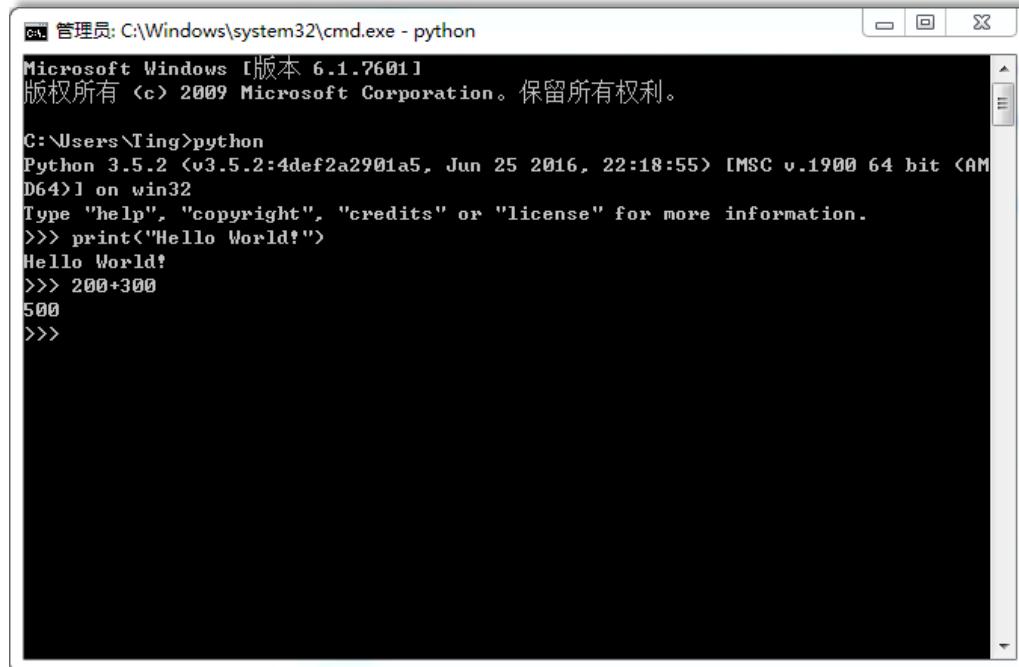


EXAMPLE 1:
Hello World!



Installation

Example 1: Hello World, Python with CMD



```
管理员: C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 <c> 2009 Microsoft Corporation。保留所有权利。

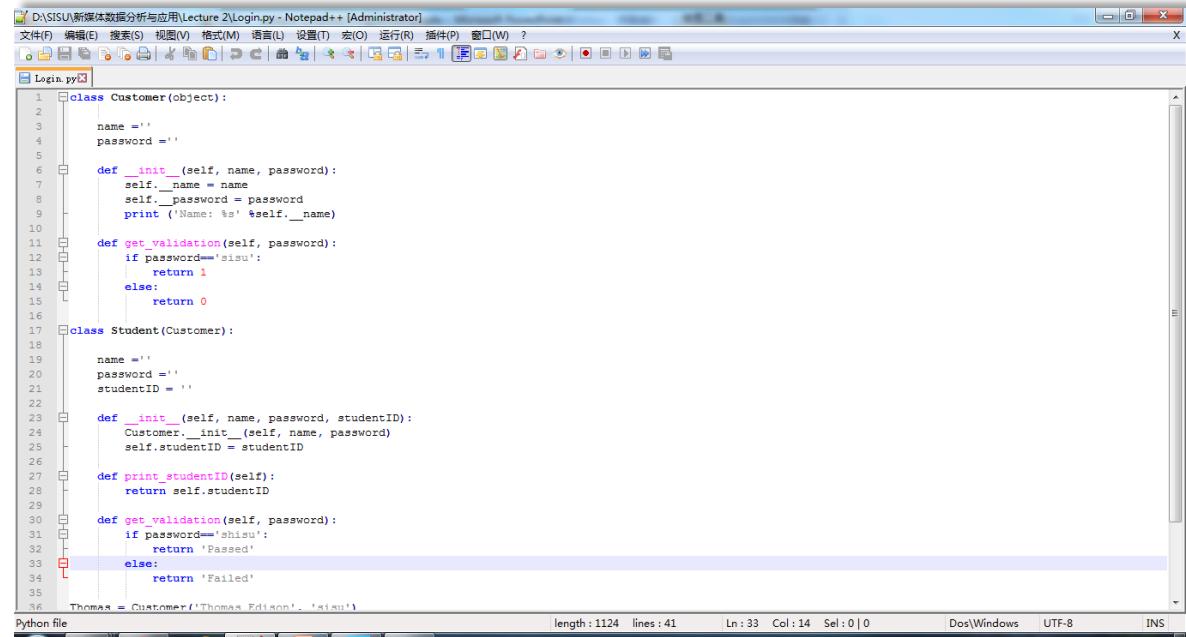
C:\Users\Ting>python
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> 200+300
500
>>>
```

Installation

IDE, Integrated Development Environment

集成开发环境

1. Notepad++



The screenshot shows the Notepad++ interface with a Python script named `Login.py`. The code defines two classes: `Customer` and `Student`, which inherit from `Customer`. The `Customer` class has attributes `name` and `password`, and a constructor `__init__` that prints the name. It also has a method `get_validation` that returns 1 if the password is 'sisu' and 0 otherwise. The `Student` class inherits from `Customer` and adds attribute `studentID`. It has a constructor `__init__` that initializes `Customer.__init__` and sets `studentID`. It also has methods `print_studentID` and `get_validation` that return the student ID and a validation message ('Passed' or 'Failed') respectively. An instance of `Customer` is created at the bottom.

```
1  class Customer(object):
2
3      name = ''
4      password = ''
5
6      def __init__(self, name, password):
7          self._name = name
8          self._password = password
9          print ('Name: %s' %self._name)
10
11     def get_validation(self, password):
12         if password=='sisu':
13             return 1
14         else:
15             return 0
16
17 class Student(Customer):
18
19     name = ''
20     password = ''
21     studentID = ''
22
23     def __init__(self, name, password, studentID):
24         Customer.__init__(self, name, password)
25         self.studentID = studentID
26
27     def print_studentID(self):
28         return self.studentID
29
30     def get_validation(self, password):
31         if password=='shisu':
32             return 'Passed'
33         else:
34             return 'Failed'
35
36 Thomas = Customer('Thomas Edison', 'sisu')
```

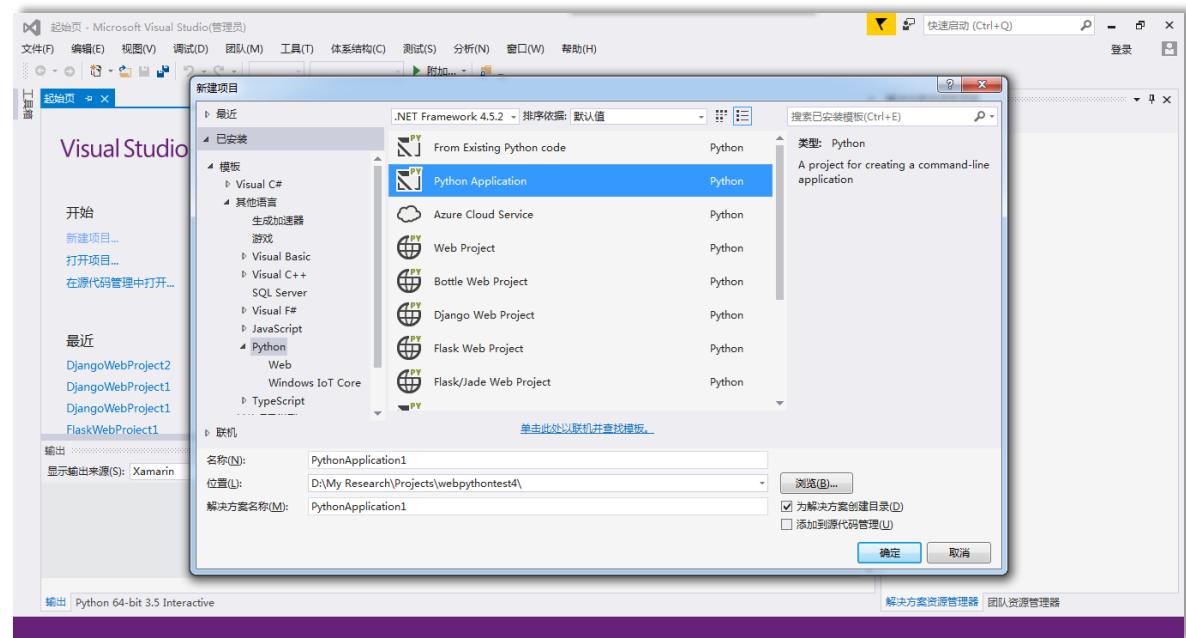
Python file length : 1124 lines : 41 Ln : 33 Col : 14 Sel : 0 | 0 Dos\Windows UTF-8 INS



Installation

IDE

2. Visual Studio 2015

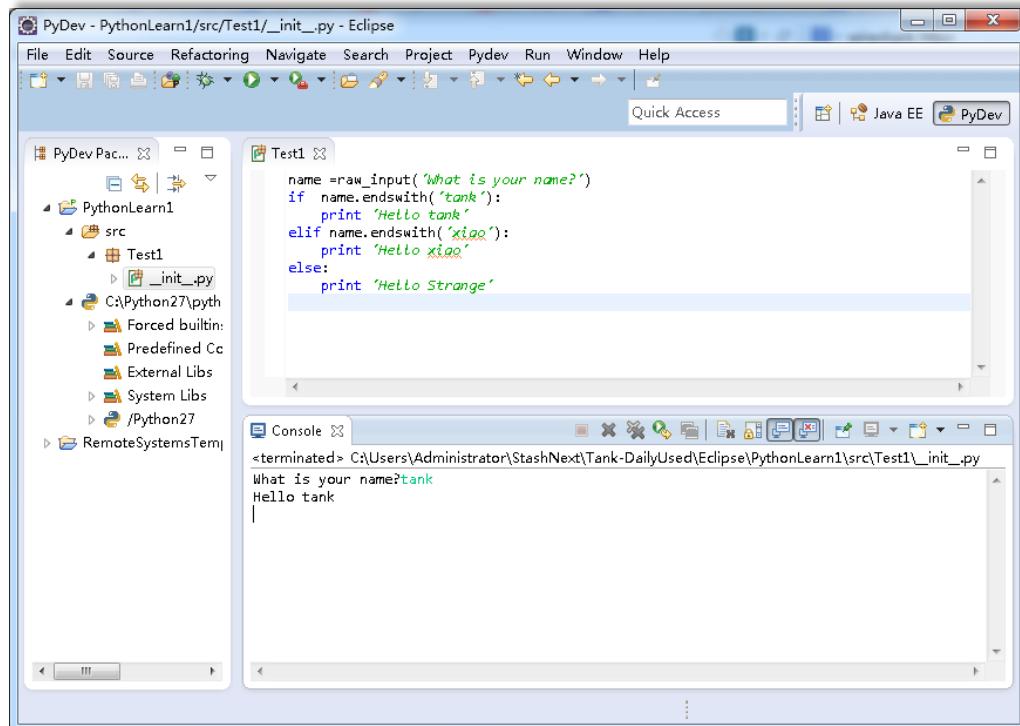


Installation

IDE

3. Eclipse + pydev

<http://pydev.org/>



The screenshot shows the Eclipse PyDev interface. On the left is the PyDev Package Explorer, displaying a project structure with a file named `__init__.py` under `src/Test1`. In the center is the `Test1` editor window containing the following Python code:

```
name = raw_input('What is your name?')
if name.endswith('tank'):
    print 'Hello tank'
elif name.endswith('xiao'):
    print 'Hello xiao'
else:
    print 'Hello Strange'
```

Below the editor is the Console window, which shows the output of running the script:

```
<terminated> C:\Users\Administrator\StashNext\Tank-DailyUsed\Eclipse\PythonLearn1\src\Test1\__init__.py
What is your name?tank
Hello tank
```

<http://www.cnblogs.com/Bonker/p/3584707.html>

Installation

IDE

4. PyCharm

Professional Version
Community Version



The screenshot shows the PyCharm Community Edition interface. The top window is titled "PycharmTest1 - [D:\My Research\Projects\PycharmTest1] - ...\\HelloWorld.py - PyCharm Community Edition 2016.2.3". It displays a Python file named "HelloWorld.py" with the following code:

```
print("Hello World!")
```

The bottom window is a terminal titled "Run" showing the output of running the script:

```
C:\Program Files\Python35\python.exe" "D:/My Research/Projects/PycharmTest1/HelloWorld.py"
Hello World!
Process finished with exit code 0
```

<http://www.jetbrains.com/pycharm/>

Installation on Other Operation Systems

<https://www.python.org/download/other/>

- IBM AS/400 (OS/400)
- BeOS
- MS-DOS
- IBM OS/2
- IBM OS/390
- Series 60
- Oracle Solaris
- HP-UX



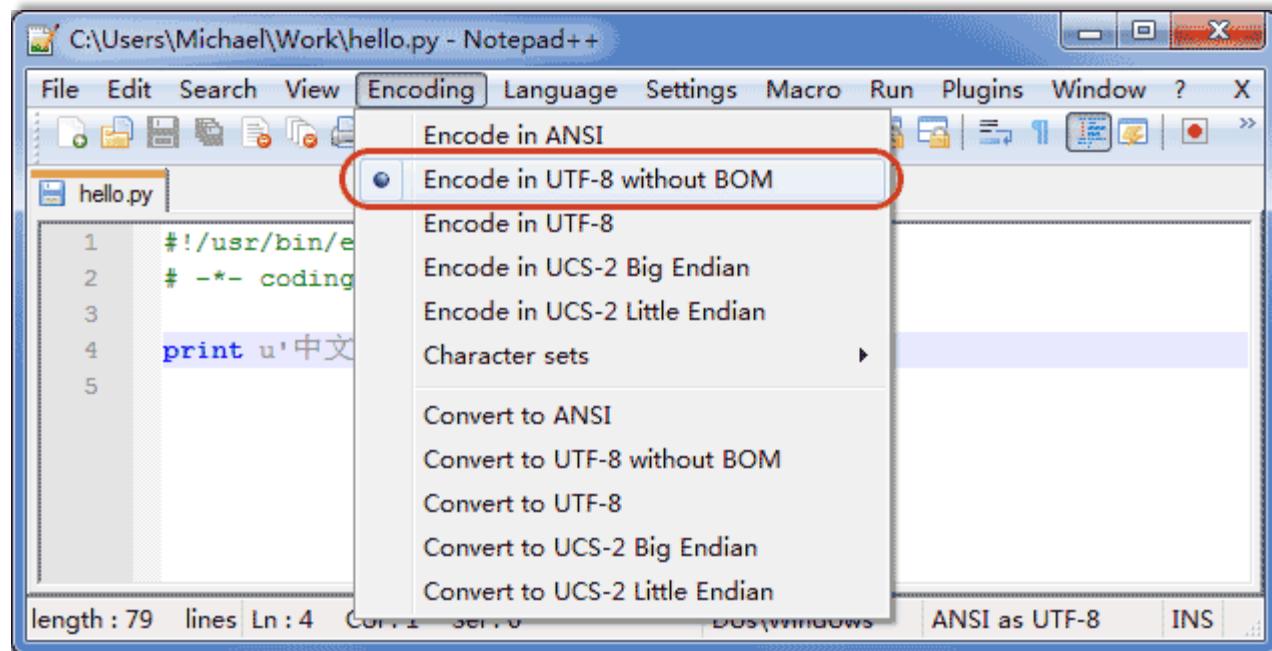


上海外国语大学
SHANGHAI INTERNATIONAL STUDIES UNIVERSITY

how to use python
Grammar

Character Encoding 字符编码

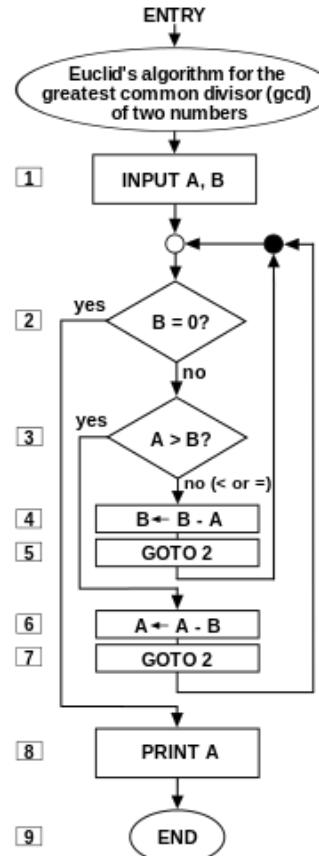
- Make sure that the encode is in UTF-8



Grammar

Algorithm 算法

a self-contained step-by-step set of operations



Variables 变量

a storage location paired with an associated symbolic **name** (an identifier), which contains some known or unknown quantity of information referred to as a **value**.

```
>>>x=2  
>>>name="Thomas"
```

Case sensitive 大小写敏感 in python

```
>>> x=2  
>>> X=3
```

They are different variables!



Input and output 输入和输出

- input()
- print()



Grammar

EXAMPLE 2:
input and output



Example 2: Hello 谁谁谁!

```
>>>name=input("What is your name?")
>>>print("Hello "+name+"!")
```



Data Structure of Variables 变量的数据结构

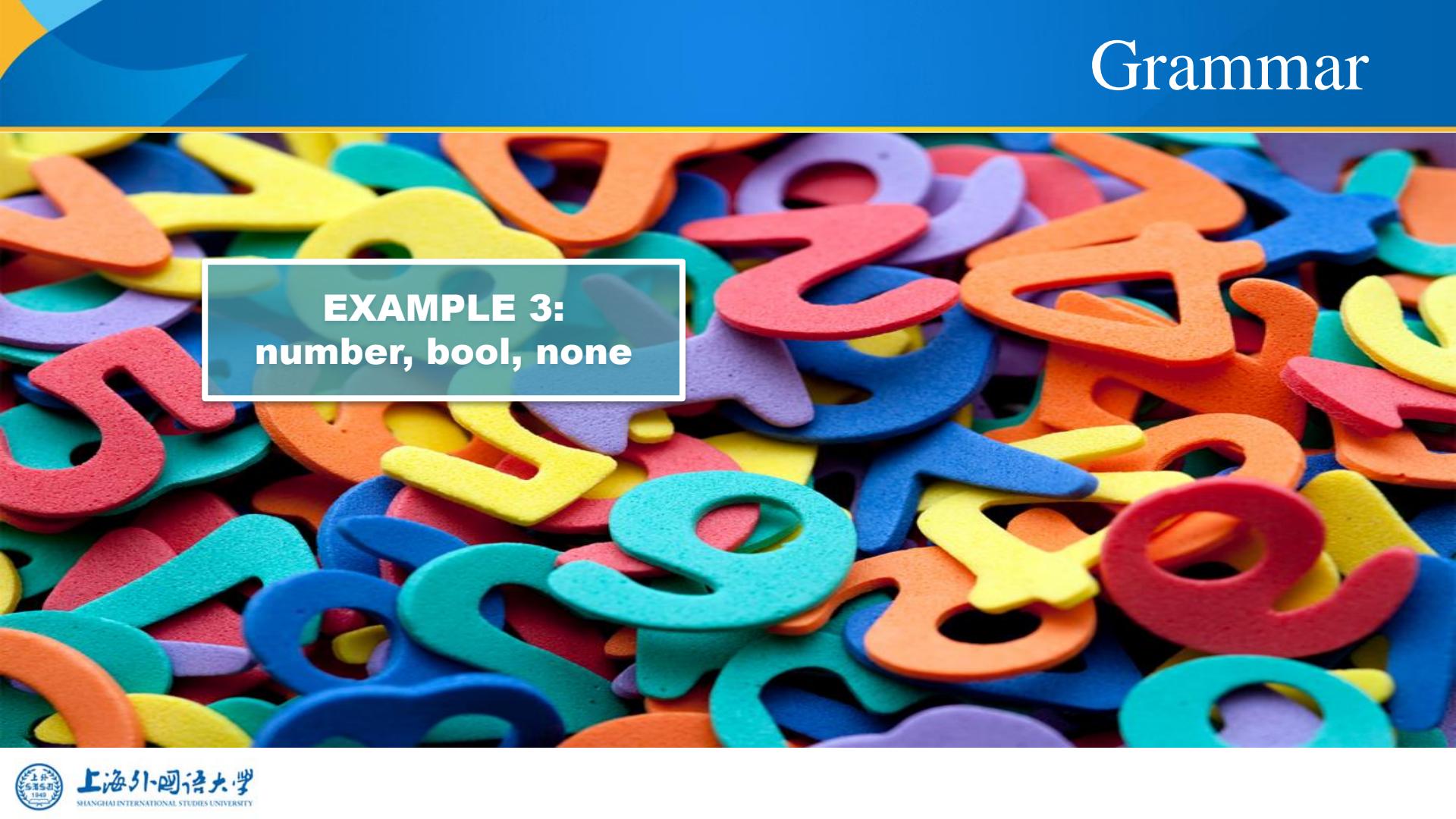
#	Type	#	Type
1	Number 数字	5	List 列表
2	Bool 布尔逻辑	6	Tuple 元组
3	None 空值	7	Dict 字典
4	String 字符串	8	Datetime 日期

Reference: <http://www.cnblogs.com/linjiqin/p/3608541.html>

Number, bool, none 数字, 布尔, 空

Type_EN	Type_CN	Human_words	Example
integer	整型	整数	x=1
float	浮点型	小数	y=1.0
bool	布尔值 True, False, and, or, not	对错, 与或非	print(x>y) print(x==y) print(not(x>y))
none	空值	二胎还没怀上, 先把 名字给取了, 占位	x=None print(x)

Grammar



EXAMPLE 3:
number, bool, none



Grammar

Operator Precedence

运算符	描述
lambda	Lambda表达式
or	布尔“或”
and	布尔“与”
not x	布尔“非”
in, not in	成员测试
is, is not	同一性测试
<, <=, >, >=, !=, ==	比较
	按位或
^	按位异或
&	按位与
<<, >>	移位
+, -	加法与减法

Low

High



运算符	描述
*, /, %	乘法、除法与取余
+x, -x	正负号
~x	按位翻转
**	指数
x.attribute	属性参考
x[index]	下标
x[index:index]	寻址段
f(arguments...)	函数调用
(expression,...)	绑定或元组显示
[expression,...]	列表显示
{key:datum,...}	字典显示
'expression,...'	字符串转换



String 字符串

- Convert

强制转化

- ESC

转义字符

- %d 整数
- %f 浮点数
- %s 字符串

```
def AddNumber(a, b):  
    return int(a)+int(b)
```

```
def AddString(a, b):  
    return a+b
```

```
InputOne=input("please input the first number: ")  
InputTwo=input("please input the second number: ")  
print("AddNumber: "+str(AddNumber(InputOne, InputTwo)))  
print("AddString: "+AddString(InputOne, InputTwo))
```

(another style of Example 2):

```
>>>name=input("What is your name?")  
>>>print("Hello, %s !" %name)
```



Grammar

Start from 0!

List 列表, 数组

```
>>>classmates = ['Michael', 'Bob', 'Tracy']
>>> classmates[0]
'Michael'
>>> classmates[1]
'Bob'
>>> classmates[2]
'Tracy'
>>> classmates[3]
Traceback (most recent call
last): File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Use []!

Tuple 元组

A list where values CANNOT be changed.

```
>>> classmates = ('Michael', 'Bob', 'Tracy')
```

Use ()!



Grammar

dict 字典

```
d = {'key1':value1, 'key2':value2}
```

```
>>> d = {'Michael': 95, 'Bob': 75, 'Tracy': 85}  
>>> d['Michael']
```

```
95
```

Use {}!

set 集合

```
s = set([key1, key2, key3])
```

- Repetitions will be discarded
- No value, only key
- Actually, set is a function

```
>>> s = set([1, 1, 2, 2, 3, 3])  
>>> s  
{1, 2, 3}
```



Datetime 时间

```
>>>import datetime  
>>>print(datetime.datetime.now())  
>>> dt = datetime.datetime(2018, 3, 5, 15, 30) # 用指定日期时间创建datetime  
>>> print(dt)  
2018-03-05 15:30:00
```

Note:

1. “datetime” is a module. It should be imported before it is employed.
2. Python has many modules for different usages. Moreover, there are also a great number of **third-party modules**, which can be installed by Python command “pip”.

Conditional Statement 条件判断

```
if <condition 1>:  
    <statement1>  
elif < condition 2>:  
    <statement2>  
elif < condition 3>:  
    <statement3>  
else:  
    <statement4>
```



Iteration 循环

```
for <counter> in <range>:  
    <statement>
```

```
while <condition>:  
    <statement>
```

break: stop the whole iteration

continue: stop this round, but continue to start the next round of this iteration





reuse and encapsulation

Functions

Define Functions

```
def FunctionName(parameter1, parameter2,...):
```

<statement>

[return value]

Optional

Call Functions

```
FunctionName(para1, para2,...)
```



Functions

EXAMPLE 4: *Recursion*



Recursion: the function call itself 递归

Example: factorial 阶乘

$$n! = n * \dots * 4 * 3 * 2 * 1$$

$$n! = n * (n-1)!$$

Assume that, $F(n)=n!$

Then $F(n-1)=(n-1)!$

$$\therefore F(n)=n * F(n-1)$$

```
def factorial(n):
    if n==1:
        return 1
    else:
        return n*factorial(n-1)
```

```
number=input("Please input the number:")
print(factorial(int(number)))
```



Object Oriented Programming

- Class
- Object
- Attribute
- Method





testing, exception and modification

Debugging

Test-Driven Development

- STEP:
 1. print() it!
 2. Do NOT forget to delete print().



Debugging

try...except...finally...

If we are not sure whether there are some errors in our code, we can use this statement.

Step 1: “try”

Step 2: Errors occur, stop “try”;

Step 3: go to “except”, and finish this part

Step 4: if there is a “finally” part then execute it;

Step 5: finish

```
try:  
    print('try...')  
    r = 10 / 0  
    print('result:', r)  
except ZeroDivisionError as e:  
    print('except:', e)  
finally:  
    print('finally...')  
print('END')
```





References

References

廖雪峰的官方网站 (*Python* 教程)

<http://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000>



目录

Python教程

Python简介
安装Python

Python解释器
第一个Python程序
使用文本编辑器

Python代码运行助手
输入和输出

Python基础
数据类型和变量
字符串和编码
使用list和tuple

条件判断
循环

使用dict和set

函数

调用函数

Python教程

阅读: 2611952

2.7旧版教程

这是小白的Python新手教程，具有如下特点：

中文，免费，零起点，完整示例，基于最新的Python 3版本。

Python是一种计算机程序设计语言。你可能已经听说过很多种流行的编程语言，比如非常难学的C语言，非常流行的Java语言，适合初学者的Basic语言，适合网页编程的JavaScript语言等等。

那Python是一种什么语言？

首先，我们普及一下编程语言的基础知识。用任何编程语言来开发程序，都是为了让计算机干活，比如下载一个MP3，编写一个文档等等，而计算机干活的CPU只认识机器指令，所以，尽管不同的编程语言差异极大，最后都得“翻译”成CPU可以执行的机器指令。而不同的编程语言，干同一个活，编写的代码量，差距也很大。

比如，完成同一个任务，C语言要写1000行代码，Java只需要写100行，而Python可能只要20行。

所以Python是一种相当高级的语言。

你也许会问，代码少还不好？代码少的代价是运行速度慢，C程序运行1秒钟，Java程序可能需要2秒，而Python程序可能就需要10秒。

那是不是越低级的程序越难学，越高级的程序越简单？表面上来说，是的，但是，在非常高的抽象计算中，高级的Python程序设计也是非常难学的，所以，高级程序语言不等于简单。



References

Microsoft Virtual Academy

https://mva.microsoft.com/zh-cn/training-courses/-python--8360?l=EK9zuOO8_2604984382

The screenshot shows a Microsoft Virtual Academy course page. At the top, there's a navigation bar with 'Microsoft Virtual Academy', a search bar, and a login button. Below the navigation, it says '初级 | 发布日期: 12 June 2015'. The main content area features a video player showing two people, a man and a woman, sitting at a desk with monitors. The video title is '使用 Python 编程简介'. To the right of the video, there's a rating of 11 stars and a progress bar indicating '6% 完成'. Below the video, there's a '学习计划' (Learning Plan) section with icons for '信息' (Information), '目录' (Table of Contents), '字幕文本' (Caption Text), and 'Related'. A '论坛' (Forum) link is also present. The '目录' (Table of Contents) section lists the following items:

时间	全部显示
01 入门	00:50:21 ✓
为什么选择 Python	00:16:42
入门	00:23:37
最佳实践	00:41:24
01 幻灯片演示文稿	
Assessment	



上海外国语大学
SHANGHAI INTERNATIONAL STUDIES UNIVERSITY

Home Work

Home Work

1. Print all the Prime Numbers smaller than 10,000.
2. Print the first 30 numbers of *Successione di Fibonacci*

You should do this homework by yourself
and submit the report and the code to me
individually before Oct. 31 via email
attachment with the title
“2018M + Your Chinese Name+ID”





The End of Lecture 2

Thank You

<http://www.wangting.ac.cn>

