Python 介绍

易读、动态、优美、灵活、快速、强大的语言

目录

背景

什么是Python?

应用广泛(后台程序, WEB, script, GUI...)

面向对象

解释性语言

动态语言同时也是强类型

注重可读性和开发效率

功能

自备全套工具, 开箱即用

万物皆对象

交互式SHELL

自省

跨平台

CPython \ jython \ IconPython \ PyPy

谁在用Python?

Google

NASA

Douban

Dropbox

Reddit

创业公司大多的首选语言...

版本迭代历史

Created in 1989 by Guido Van Rossum

Python 1.0 released in 1994.

Python 2.0 released in 2000.

Python 3.0 released in 2008.

Python 2.7 推荐版本, 同时兼容2x、3x

Python 3.5 最新版本 2015-09-13

语法

Hello world

```
# 打印Hello world!
print("Hello world!")

# Python 2.7 之前，不包括Python2.7
print "Hello world!"
```

缩进

大多数语言不关心缩进

但大多数码农都去关注缩进

Python 强制缩进(4个空格)

逻辑块分明

缩进

```
if a > b:
    print("a 大于 b")
elif a < b:
    print("a 小于 b")
else:
    print("a 等于 b")
```

```
if a > b:
print("a 大于 b")
elif a < b:
print("a 小于 b")
else:
print("a 等于 b")
```

第一种明显比较清晰更加已读

事实上， 我们选择不了第二种

Python 强 制 缩 进

注释

```
"""
这是一个多行文档注释， 一般生成文档的时候使用.
Python 可以支持代码即文档就是通过这里产生.
"""

"这是单行文档注释"

# 此为代码注释, 一般给阅读代码的人看
```

```
>>> help(comments)
Help on module comments:

NAME
    comments

FILE
    /Users/JackeyGao/Coding/python-intro/comments.py

DESCRIPTION
    这是一个多行文档注释， 一般生成文档的时候使用.
    Python 可以支持代码即文档就是通过这里产生.
```

数据类型

Strings

```
# 这是个字符串
name = "Jackey Gao (that \"s me)"

# 这也是个字符串
home = 'Henan, Shangqiu'

# 这是个多行字符串, 使用'三单引'
address = '''河南省
商丘市 '''

# 下面也是多行字符串, 使用"三双引"
address = """河南省
商丘市"""

# 看自己心情使用, 最好使用单引号, 容易输入
```

Numbers

```
# 整型数字
year = 2016
year = int("2016")

# 浮点小数
pi = 3.1415
pi = float("3.14159265")

# 固定小数点
from decimal import Decimal
price = Decimal("0.02")
```

Null

```
# 使用None, 第一个必须大写
option = None
```

Lists

```
# 一个空的列表
favorites = []

# 追加元素
favorites.append(26)

# 延伸列表
favorites.extend(["Python", True])
# output: [26, 'Python', True]

# 直接赋值声明
favorites = [26, 'Python', True]
```

Dicts

```
# 一个空字典
person = {}

# 设置字典的一个key和value
person["name"] = "Jackey Gao"

# 更新字典
person.update({
    "favorites": [1, 2, 3],
    "gender": "male"
    })

# 任何对象都可以成为key
person[26] = "数字"
person[(3.14, 1.23)] = "元组"
```

Dict Methods

```
person = { "name": "JackeyGao", "gender": "Male" }

# 通过以下方法可以访问value
person['name']

# 当一个key 不存在, 会抛出KeyError
# 可以使用以下方法, 设置默认值
person.get('name', 'nobody')

# 获取字典的keys(list)
person.keys()
# ['name', 'gender']

# 获取字典的values(list)
person.values()
# ['JackeyGao', 'Male']

# 获取键值对应
person.items()
# [['name', 'JackeyGao'], ['gender', 'Male']]
```

Booleans

```
# 布尔值 True 和 False
is_python = True

is_perl = False

# 所有的对象都可以当作布尔值
is_python = bool("any object")


# 任何为空都是False
these_are_false = False or 0 or "" or [] or {} or None

# 相反任何不为空则为True
these_are_true = True and "null" and 1 and "Str" and [1,2] and {"n": 1
```

操作符

运算符

```
a = 10 # 10
a += 1 # 11
a -= 1 # 10

b = a + 1  # 11
c = a - 1  # 9

# 乘除余幂
d = a * 2  # 20
e = a / 2  # 5
f = a % 3  # 1
g = a ** 2 # 100
```

字符串操作符

```
animals = "Cats " + "Dogs "
animals += "Rabbits "
# 'Cats Dogs Rabbits '

# 以,分割把list拼成字符
fruit = ', '.join(['Apple', 'Banana', 'Orange'])
# 'Apple, Banana, Orange'

# 字符串格式化
date = "%s %d %d" % ('Apr', 25, 2016)
# 'Apr 25 2016'

# 字符串格式化 - 字典
description = '我叫%(name)s, 我来自%(country)s' % {
        "name": "王铁柱",
        "country": "澳大利亚"}

# 我叫王铁柱, 我来自澳大利亚
```

逻辑比较

```
a = '1'
b = 2
c = None
# and
a and b

# or
a or c

# 逻辑否
not a

# 复合逻辑
(a and not (b or c))
```

等式比较

```
# 等式比较
1 is 1 == True
1 is not '1' = True

# 案例
bool(1) == True
bool(True) == True

1 and True = True
1 is True = False
```

算术比较

```
# 比大小
a > b
a >= b
a < b
a <= b

# 等于 不等
a == b
a != b
```

控制流

条件语句

```
grade = 82

if grade >= 90:
    if grade == 100:
        print('A+')
    else:
        print('A')
elif grade >= 80:
    print('B')
elif grade >= 70:
    print('C')
else:
    print('F')

# B
```

循环语句

```
for i in xrange(10):
    print i

for i in ['Apple', 'Banana']:
    print i
```

扩展循环语句

```
states = {
        "ZZ": "诸葛铁柱",
        "SN": "上官铁牛"
        }

for code, name in states.items():
    print code, name
```

while循环语句

```
while True:
    print("死循环")


x = 0
while x < 100:
    x += 1
```

列表推导式

```
# 列表推导, 求奇数
odds = [ x for x in range(50) if x % 2  ]

# 同等于下面
odds = []
for x in range(50):
    if x % 2:
        odds.append(x)


# 字典推导
list1 = ["Apple", 'Banana', 'Orange']
list2 = [1, 2, 3]
d = { a: b for a, b in zip(list2, list1) }
# {1: 'Apple', 2: 'Banana', 3: 'Orange'}
```

函数

基础函数

```
def my_function():
    """这是一个简单的函数, 这个函数没有返回任何东西"""
    print("Hello world")

function_return = my_function()
# function_return is None
```

带参数函数

```
# 位置参数
def add(x, y):
    return x + y

added = add(1, 1) # 2

# 键值参数， 可以设置默认值
def shout(phrase="Yipee!"):
    return phrase

# 位置参数 + 键值参数
def echo(text, prefix='!'):
    print("%s%s" % (text, prefix))

echo("Hello")

# stdout: Hello!
```

任意参数函数

```python
def some_method(*args, **kwargs):
    print args
    print kwargs

some_method(1.47, '40kg', name=u"小明")

# (1.47, '40kg')
# {'name': u'小明'}

# 案例
def add_all(*numbers):
    n = 0
    for i in numbers:
        n += i
    return n

add_all(1, 2, 3, 4, 5) # 15
add_all(1, 2, 3, 4, 5, 6, 7, 8) # 36
```

Fibonacci

```
def fib(n):
    """这个数列从第2项开始，每一项都等于前两项之和。"""
    results = []
    a, b = 0, 1
    while a < n:
        results.append(a)
        a, b = b, a + b

    return results

print fib(50)
# [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Fibonacci Generator

```
def fib():
    """Yield Fibonacci"""
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

f = fib()
fibs = [ f.next() for i in range(10) ]
# [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

类

类 - 声明

```
# 声明一个类, 继承新式类
class Person(object):
    pass

# Notes:  python的新式类是2.2版本引进来的,
# 我们可以将之前的类叫做经典类或者旧类。
# 为什么要在2.2中引进new style class呢?
# 官方给的解释是
# "为了统一类(class)和类型(type)。"
# 新式类的好处
#   - 内置属性将会引入
#   - 描述符的引入
#   - 属性可以来计算等等
```

```
>>> class OldClass:
...     pass
...
>>> class NewClass(object):
...     pass
...
>>> old = OldClass()
>>> new = NewClass()
>>> dir(old)
['__doc__', '__module__']
>>> dir(new)
['__class__', '__delattr__', '__dict__',
'__doc__', '__format__', '__getattribute__',
'__hash__', '__init__', '__module__',
'__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__weakref__']
>>>
```

任何时候都应该使用新式类了

类 - 方法

```
# 声明一个类, 继承新式类
class User(object):
    is_staff = False

    def __init__(self, name="Anonymous"):
        self.name = name
        super(User, self).__init__()

    def is_authorized(self):
        # 类方法的定义和函数声明一样
        # 都是使用 def 语句
        # 需要在类的缩进块里面写
        return self.is_staff
```

实例化&属性访问

```
anonymous = User()
print(anonymous.name)
# Anonymous

xiaoming = User(name="xiaoming")
print(xiaoming.name)
# xiaoming

# 调用实例方法
print(xiaoming.is_authorized())
# False
```

实例化&属性访问

```
anonymous = User()
print(anonymous.name)
# Anonymous

xiaoming = User(name="xiaoming")
print(xiaoming.name)
# xiaoming

# 调用实例方法
print(xiaoming.is_authorized())
# False
```

继承

```
# 继承自 User
class SuperUser(User):
    is_staff = True


root = SuperUser('root')
print(root.name)
# root

# 因为前面继承的User有is_authorized方法
print root.is_authorized()
# True
```

Imports

- Python 允许代码隔离和复用
- 允许 variables / class / function 类型的引用到当前作用域.

Imports

```
# 引用标准库 datetime
import datetime
print(datetime.date.today())
print(datetime.timedelta(days=1))

# 具体引用 from ... import ..., ...
from datetime import date, datetime
print(date.today())
print(datetime.now())
```

更多的import方式

```
# 重命名 import, 一般使用在引入的名字冲突的情况下
from datetime import date
from my_module import date as my_date

# 还有以下引入， 导入所有的模块
from datetime import *
```

from module import * 是个很大的禁忌

异常处理

```
import datetime
import random

day = random.choice(['Eleventh', 11])
try:
    date = 'Apr ' + day
except TypeError:
    date = datetime.date(2016, 4, day)
else:
    date += ' 2016'
finally:
    print date
```

文档

help()

```
>>> import datetime

>>> help(datetime)

Help on module datetime:

NAME
    datetime - Fast implementation of the datetime type.

FILE
    /Users/JackeyGao/.virtualenvs/report_server/lib/python2.7/lib-dynload/datetime.so

CLASSES
    __builtin__.object
        date
            datetime
        time
        timedelta
        tzinfo

    class date(__builtin__.object)
     |  date(year, month, day) --> date object
     |
     |  Methods defined here:
```

Tools

Frameworks

Django(web)

Flask(web)

Tornado(web)

Django_rest_framework(RESTful)

scrapy(Spiders)

Celery(Asynchronous task queue)

IDEs

vim

Emacs

Sublime Text 2

komodo

PyCharm

包管理

pip 是python包管理器, pip 一般随着安装python安装
如果没有安装使用以下方式安装

```
$ easy_install pip

or

$ yum -y install python-pip
```

安装Python包

```
$ pip install django
$ pip install django==1.8.0
```

进阶学习

Decorators(装饰器)

Context Managers(上下文管理器)

Lambda Functions(匿名函数)

Generators(迭代器)

.....

Q&A

The end, Thanks!

@JackeyGao