NLP 南京大学
自然语言处理研究组
Natural Language Processing Group

南京大学
NANJING UNIVERSITY

# 初识Python

黄书剑


南京大學
NANJING UNIVERSITY

# PYTHON简介

# Python语言

- **Started around Christmas 1989 by Guido van Rossum**
- **Python 0.9.0 1991.02**
- **Python 2.0 2000.10-2020**
- **Python 3.0 2008.12**
  - 3.6.x and later
  - Latest Python 3.10.2

Guido van Rossum
吉多·范罗苏姆

Monty Python‘s Flying Circus (蒙提·派森的飞行马戏团) BBC
Run by Python Software Foundation (2001.03)
“Benevolent Dictator for Life” (BDFL) until 2018.07
Now by a Python Steering Council

# Python的特点

- **脚本语言（script language）**
  - 解释执行(官方解释器: CPython)
  - 可以避免编译过程，方便处理各种简单任务
- **胶水语言（glue language）**
  - 可以在Python中调用C/C++等语言代码
  - 支持多语言混合编程

- **动态类型语言、强类型语言**
- **支持函数式、命令式、结构化、面向对象等范式**
- **跨平台、资源丰富**

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

**显式强于隐式！**
**语法的简单性！**
**强调代码的可读性！**
**语法的规范性（少打补丁）！**
**严格的错误处理！**
**……**

# Hello World!

- **第一个python程序！**
- **交互方式 v.s. 文件方式**

在python交互窗口中输入：

```
>>> print ("hello world")
hello world
>>>
```

文件hello_world.py:

```
print ("hello world")
```

```
huangsj$ python3 hello_world.py
hello world
huangsj$
```

# 如何学习一门语言?

- **运行环境**
  - 编译/解释运行
- **程序基本风格**
- **基本符号**
  - 变量、类型、表达式、语句
- **流程控制**
- **函数**
- **类和对象**
- **其他（文件处理、异常处理等）**


南京大學
NANJING UNIVERSITY

# 基本程序环境

# Python运行环境

- **编译执行**
  - 编译-目标代码
  - 链接-可执行文件
  - 运行

```
Ltmp0:
        movq    %rsp, %rbp
Ltmp1:
        subq    $16, %rsp
Ltmp2:
        movq    __ZSt4cout@GOTPCREL(%rip), %rax
        leaq    (%rax), %rax
        leaq    L_.str(%rip), %rcx
        movq    %rax, %rdi
        movq    %rcx, %rsi
```

  - 运行时可以脱离编译器
  - 编译运行流程复杂

- **v.s. 解释执行**
  - 由解释器逐行执行
  - 文件方式/交互方式相同

```
>>> x = 3
>>> y = 3 * x
>>> print(y)
9
>>> z = x + y
>>> print(z)
12
```

  - 依赖于运行环境
  - 易于修改更新

# 解释执行遇到问题

- **解释器同样会返回解释错误：**

```
>>> print x
  File "<stdin>", line 1
    print x
          ^
SyntaxError: Missing parentheses in call to 'print'. Did
you mean print(x)?
```

```
>>> print(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
```

– 耐心阅读解释器提示，解决可能的问题。

# 交互式运行的优势

- **随时查看解释器的状态**
  - 变量的值、声明的函数等
- **dir() help() 等**

保持解释器状态和预期一致是程序调试的核心

```
>>> help(print)

Help on built-in function print in module builtins:
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

# 程序调试

- **在控制台下可以导入pdb包，通过pdb.run进入调试环境：**

```
>>> import pdb
>>> pdb.run('print("hello world")')
> <string>(1)<module>()
(Pdb) n
hello world
--Return--
```

- **执行文件时，可使用 –m pdb 参数，加载pdb模块。**

```
huangsj$ python -m pdb hello_world.py
>
huansj/hello_world.py(1)<module>()
-> print ("hello world")
(Pdb)
```

https://docs.python.org/3/library/pdb.html

# 常用环境和工具

- **文本编辑器 + Python解释器**





# 程序基本风格

- # 表示 注释语句
- 换行 表示 语句结束（若一行有多个语句，分号分隔）
- 缩进 表示 语句块

```
x = eval(input())
if x > 5 :
    x -= 5
    print(x)
else:
    print(5)
```

c/c++ code

```
int x;
scanf("%d", &x);
if(x > 5)
{
    x -= 5;
    printf("%d", x);
}
else
{
    printf("5");
}
```

```
int x;
cin >> x;
if(x > 5)
{
    x -= 5;
    cout << x;
}
else
{
    cout << 5;
}
```


南京大学
NANJING UNIVERSITY

值、类型、变量、表达式、语句

# 语言的基本符号

# Python中的值和类型

- **Python是强类型语言！**
- **整数和浮点数**
  - 数值类型
  - 可以用类似科学计算法的形式
  - 可以用 _ 分隔很长的数字

```
>>> type(5)
<class 'int'>
>>> type(3.1)
<class 'float'>
```

```
>>> print(10.8e5)
1080000.0
>>> type(100_000_000)
<class 'int'>
```

- **数值运算**
  - 可以进行 + - * / 运算
    - 除法运算的结果总是浮点数
  - 用 ** 表示 乘方运算
  - 用 // 表示 除后向下取整

```
>>> 1//5
0
>>> -1 // 5
-1
```

# Python中的值和类型

- **逻辑值 bool 类型**
  - 逻辑运算 not and or
  - 本质是整型的0、1
  - 其他值参与逻辑运算时：
    0或空为False，非空为True

```
>>> type( 3 > 5 )
<class 'bool'>
```

```
>>> type(3 + 5j)
<class 'complex'>
>>> 1j **2
(-1+0j)
```

- **复数类型**
  - 实部和虚部组合构成
  - 虚部用j表示

```
>>> j**2
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
NameError: name 'j' is not defined
```

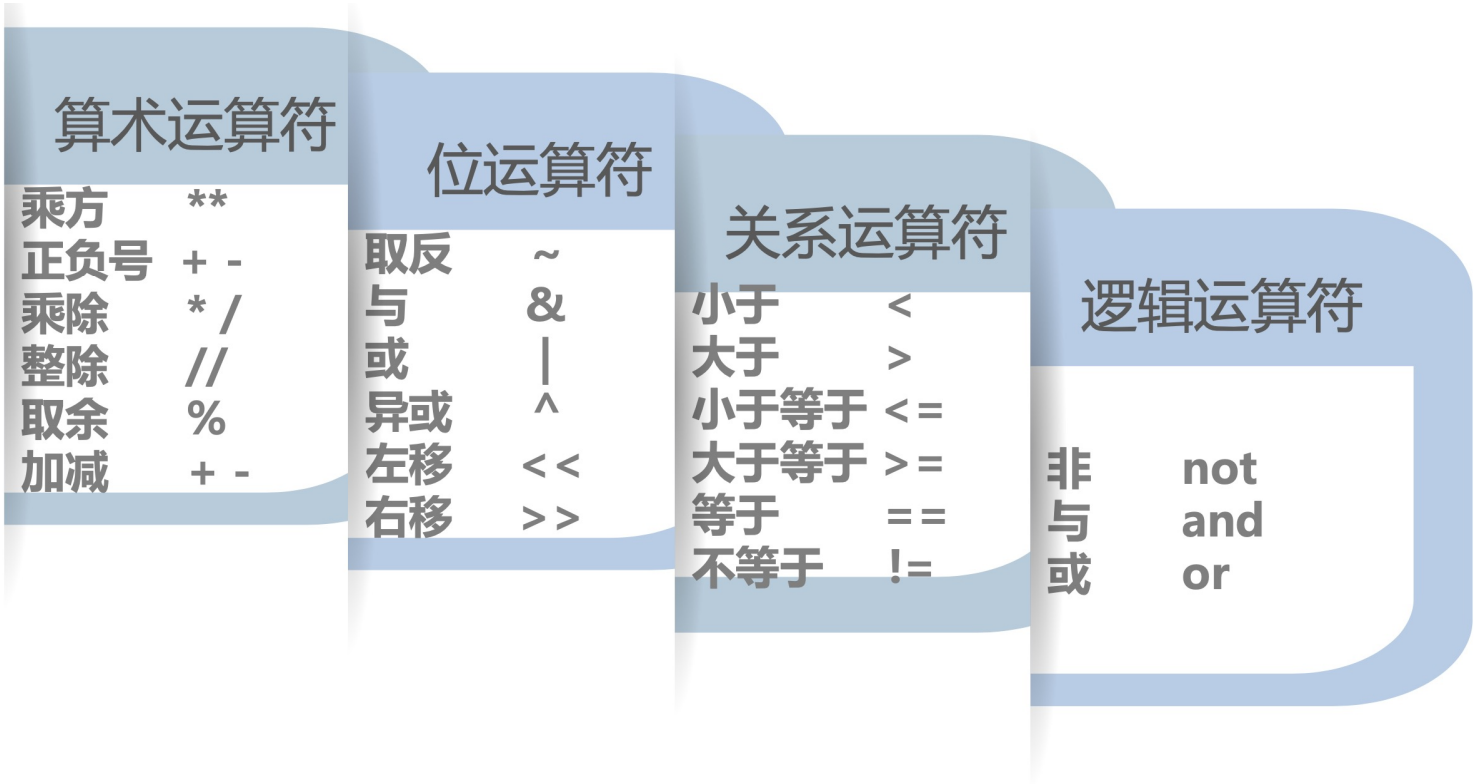
# Python中的值和类型

- **字符串**
  - 单引号、双引号标识
  - 三引号可以跨行引用

```
>>> type("string2")
<class 'str'>
>>> type('string1')
<class 'str'>
>>> type('str"i"ng1')
<class 'str’>
>>> print("str'i'ng1")
str'i’ng1

>>> print('''this is a very long
... sentence, so i have to break
... into several lines''')
this is a very long
sentence, so i have to break
into several lines
```

- **各类数值可以进行数值运算**



- **Python提供了常用字符串功能**
  - upper(), lower()
  - split()
  - strip(), lstrip(), rstrip()
  - 字符串加、乘表示其拼接操作
  - 但字符串的值不能修改

```
>>> "Guido van Rossum".upper()
'GUIDO VAN ROSSUM'

>>> "Guido van Rossum".split()
['Guido', 'van', 'Rossum']

>>> "  Guido ".lstrip()
'Guido '
```

```
>>> (("happy" + " ") * 3).strip()
'happy happy happy'
```

```
>>> "hello"[0]
'h'
>>> "hello"[0]='H'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support
item assignment
```

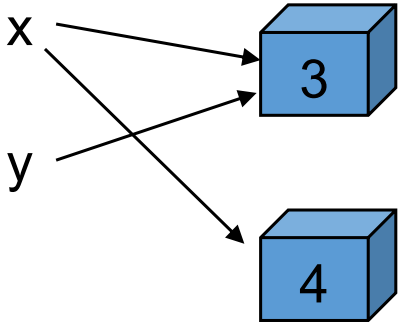
# 复杂（复合）数据类型

- **列表**
  - [1, 2, 3]  [‘Monday’,’Tuesday’,’Thursday’]
- **元组**
  - (‘Tim’,’Peters’) (‘Nanjing’, ‘Qixia’, ‘Xianlin Avenue’, ‘163’)
- **range对象**
  - range(1,5)  ：1到4组成的序列
- **字典**
  - {'Mayue': 3000, 'Lilin': 4500, 'Wuyun': 8000}
- **集合**
  - {1, 2, 3}

后续再详细介绍

# Python中的变量

- **Python是动态类型语言！**
  - 值的类型无须指定
- **每个变量都是一个标识符/变量名**
  - 用于表示一个值（binding）
  - 变量首次出现时即与对应值相关联
  - 随后也可以关联其他的值

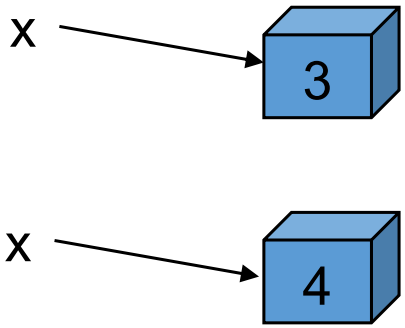



```
>>> x = 3
>>> y = x
>>> x = 4
>>> y
3
```

```
>>> x = 3
>>> y = x
>>> y = 4
>>> x
3
```

```
>>> id(x)
4314154224
>>> id(y)
4314154192
>>> id(3)
4314154192
>>> id(4)
4314154224
```

变量是可以赋给值的标签，而不是这些值本身




```
>>> x = 3
>>> id(x)
4314154192
>>> x = 4
>>> id(x)
4314154224
>>> x = x + 1
>>> id(x)
4314154256
```

- **值的生命周期管理由Python解释器完成（自动垃圾回收）**

http://pythontutor.com/

# 变量命名

- **只能包含字母、数字和下划线**
  - 不能以数字开头、不能包含空格
- **不能使用Python的关键字**
- **不建议使用内建函数名**

| False | None | True | and | as | assert | break | class | continue |
|---|---|---|---|---|---|---|---|---|
| def | del | elif | else | except | finally | for | from | global |
| if | import | in | is | lambda | nonlocal | not | or | pass |
| raise | return | try | while | with | yield | | | |

| | | Built-in Functions | | |
|---|---|---|---|---|
| abs() | dict() | help() | min() | setattr() |
| all() | dir() | hex() | next() | slice() |
| any() | divmod() | id() | object() | sorted() |
| ascii() | enumerate() | input() | oct() | staticmethod() |
| bin() | eval() | int() | open() | str() |
| bool() | exec() | isinstance() | ord() | sum() |
| bytearray() | filter() | issubclass() | pow() | super() |
| bytes() | float() | iter() | print() | tuple() |
| callable() | format() | len() | property() | type() |
| chr() | frozenset() | list() | range() | vars() |
| classmethod() | getattr() | locals() | repr() | zip() |
| compile() | globals() | map() | reversed() | __import__() |
| complex() | hasattr() | max() | round() | |
| delattr() | hash() | memoryview() | set() | |

# 变量和赋值

- **没有赋值的变量不能使用（有时可能是命名错误）**

```
>>> z
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'z' is not defined
```

- **可以同时给多个变量赋值**

```
>>> x, y = 1, 2
>>> x
1
>>> y
2
```

# 从控制台获取值

- **input()函数的执行结果是字符串类型**
  - eval()可以对字符串求值
  - int() , float()转换成对应类型

```
>>> type (input())
3.5
<class 'str'>
>>> type (eval(input()))
3
<class 'int'>
>>> type (eval(input()))
3.5
<class 'float'>
```

- **Python是强类型语言，每个值的类型都是确定的，不进行隐式的类型转换**

“Explicit is better than implicit.”

- **Python是动态类型语言，值和变量的类型是运行时确定的（相对于编译时通过定义、申明确定）**

# 变量和赋值

- **可以将函数赋值给变量**
**(函数也是一种值)**
**(函数式编程的重要特点)**

```
>>> myfunc = print
>>> myfunc("hello world")
hello world
```

- **可以从其他模块中引入变量**

```
>>> from math import pi
>>> pi
3.141592653589793
```

```
>>> from math import floor
>>> floor(pi)
3
```

# Python中的表达式

- **值、变量是表达式**
  - 3　　“Hello World”
- **运算符表达式**
  - 3 + y　　x + 5　　x ** 2　　“Hello World” * 3
- **函数调用表达式**
  - print(3+y)　　input()　　help(print)
    dir(__name__)
  - 内建函数、第三方函数、自定义函数

# 表达式求值

- **解释器确定表达式值的过程**
  - 值　　的 求值结果 是 值本身
  - 变量　　的 求值结果 是 变量引用的值
  - 表达式 的 求值结果 是 操作符与操作数作用的结果
    - 分别求操作符、操作数的值
    - 再按照操作符/函数的定义进行运算
    - （函数的 求值结果 是其返回值表达式的运算结果）

- **当多个操作符同时出现时，考虑操作符的优先级和结合性**
  - 算术运算符 > 位运算符 > 关系运算符 > 逻辑运算符
  - 运用括号提升程序可读性

# 表达式求值

- **函数调用可以看做是一种特殊的操作符应用**
  - print(4)  -> print 4
  - max(3, 4, 1) ->  max 3 4 1
- **也可以把所有操作符都看做是函数调用**
  - 4+5  -> add(4, 5)
  - 4**2 + 3 – 2 -> minor( add(power(4, 2), 3), 2)

```
>>> def myadd (x, y):
...     return x + y
...
>>> myadd(2, 4)
6
```

自定义函数 后续再详细介绍

# 构成更大规模的代码

- **通过复合语句控制程序流程**
  - 条件分支语句 if-else  if-elif-else 等
  - 循环语句 while for 等

- **更大规模的模块、包、库等**
  - 模块：一个.py文件构成一个模块（module）
  - 包：一个包有多个模块或者子包构成（一般包含一个__init__.py文件，相当于包的头文件）

```
packageA/
    __init__.py
    moduleA.py
    subPackageB/
        __init__.py
        moduleB1.py
        moduleB2.py
    subPackageC/
        __init__.py
        moduleC1.py
        moduleC2.py
        ...
```

```
import packageA.subPackageB
from packageA.subPackageC import moduleC1
```

# 回顾

- **Python的运行环境**
- **Python的程序基本风格**
- **Python的基本元素**

- **一些常用函数**
  - input()
  - print()
  - id() type()

# 本周任务


- 搭建自己的Python代码编辑、执行环境
- 完成OJ在线编程作业
- 尝试自行设计完成一些简单的功能

# 参考资料

- **快速了解一下python的要点：**
  **https://www.stavros.io/tutorials/python/**

- **更多的Python入门资源：**

**https://wiki.python.org/moin/BeginnersGuide/Programmers**