大学计算机入门课

Chenyu Guo

Class #2

［教学目的］

- Know most python instructions.
- Be able to take human problems and write Python programs that solve them.
- Have a sense of what computer scientists do.

［课程大纲］

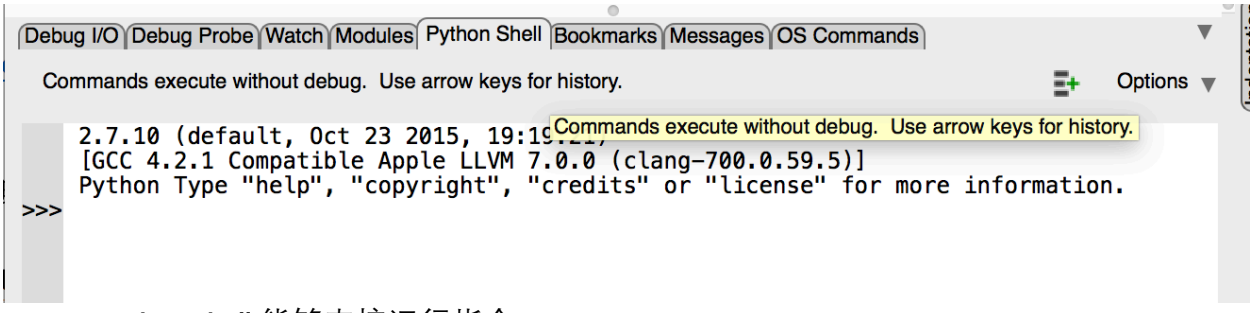
- Part1- installation
- Part2 – python shell
  - As a calculator
  - Data type
  - variables
- part 3 – function
  - python build-in function
  - design function

part1 – installation

download python
https://www.python.org/downloads/
download Wing101
http://www.wingware.com/downloads/wingide-101/5.1.7-1/

part 2 — Python shell

as a calculator



- **Python shell** 能够直接运行指令
- 他具有计算功能
  basic Syntax（语法）在这里我们讨论如何运用计算机语言。

Syntax	Math	Operation Name
`a+b`	$a + b$	addition
`a-b`	$a - b$	subtraction
`a*b`	$a \times b$	multiplication
`a/b`	$a \div b$	division (see note below)
`a//b`	$\lfloor a \div b \rfloor$	floor division (e.g. 5//2=2) - Available in Python 2.2 and later
`a%b`	$a \mod b$	modulo
`-a`	$-a$	negation
`abs(a)`	$\lvert a \rvert$	absolute value
`a**b`	a^b	exponent
`math.sqrt(a)`	$\sqrt{a}$	square root

”>>>” 这一部分是用来写自己的指令的
“return”(回车) 可以等到自己的运行结果
运行结果前并没有 “>>>”

```
Python Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 2
3
>>> 5 + 7
12
>>> 7 - 6
1
>>> 5 * 3
15
>>> 2**3
8
>>> 8/4
2
>>> 8/3
2
>>> 8/3.0
2.6666666666666665
>>> 8//3
2
```

当然，并不是所有的命令都会执行。比如当我们想计算 4/0 时， 计算机会自动出现 Error(ZeroDivisionError).

```
>>> 4/0
Traceback (most recent call last):
  Python Shell, prompt 1, line 1
ZeroDivisionError: integer division or modulo by zero
>>>
```

当格式不对时，我们会出现 invalid syntax

```
>>> 4 ** 2
16
>>> 4**2
16
>>> 4 * * 2
Traceback (most recent call last):
  Python Shell, prompt 13, line 1
invalid syntax: <string>, line 1, pos 5
>>>
```

- 更多的数学计算，我们可以通过 import math 来实现

```
>>> import math
>>> math.sqrt(4)
2.0
>>> math.pi
3.141592653589793
>>> math.sin(math.pi)
1.2246467991473532e-16
>>> math.sin(math.pi/3)
0.8660254037844386
>>> math.sin(math.pi/6)
0.49999999999999994
```

- 计算顺序

Name	Syntax	Description	PEMDAS Mnemonic
Parentheses	`( ... )`	Before operating on anything else, Python must evaluate all parentheticals starting at the innermost level. (This includes functions.)	Please
Exponents	`**`	As an exponent is simply short multiplication or division, it should be evaluated before them.	Excuse
Multiplication and Division	`* / // %`	Again, multiplication is rapid addition and must, therefore, happen first.	My Dear
Addition and Subtraction	`+ -`		Aunt Sally

括号 -> 指数 -> 乘除 -> 加减

- 我们还可以比较大小关系，return value： True or False

```
>>> 2 > 3
False
>>> 3 >= 2
True
>>> 3 == 3
True
>>> 3 == 5
False
```

data type

在 Python 里，所有的 data 都有 type

- 1 – int （integer）
- 0.55 – float
- True, False - boolean
- “23438” ， “hello” – string

  在这里注意所有的字母如果想要表示成 data 的形式，都需要用 “” 。

```
>>> hello
Traceback (most recent call last):
  Python Shell, prompt 15, line 1
NameError: name 'hello' is not defined
>>> "hello"
'hello'
>>>
```

- [1, 2, 3, 2] , [“hello”, ”bye”] – list （we will cover this in the future）
- {1,2,3,4} – set
- {1: “Monday”, 2: “Tuesday”, 3: “Wednesday”, 4: “Thursday”} – dictionary

当我们想要知道一个 data 的 type 时

```
>>> type(1)
<type 'int'>
>>> type(4.5)
<type 'float'>
>>> type({1,2})
<type 'set'>
>>>
```

data type 的相互转化

为什么我们需要这个？

因为很多命令执行时会要求响应的 data type

```
>>> 1 + 1
2
>>> "1" + 1
Traceback (most recent call last):
  Python Shell, prompt 21, line 1
TypeError: cannot concatenate 'str' and 'int' objects
>>>
```

所以我们想要把 string 转化成 int。（通过 python 自带的 function）

```
>>> int("1")
1
>>> int("1") + 1
2
```

但是只有引号里的全是数字时我们才可以做这样的转化

```
>>> int("234")
234
>>> int("abc")
Traceback (most recent call last):
  Python Shell, prompt 25, line 1
ValueError: invalid literal for int() with base 10: 'abc'
```

其他的转化

```
>>> float(1)
1.0
>>> int(1.0)
1
>>> int(1.5)
1
>>> str(2)
'2'
```

note：string 与 sting 之间也可以相加，变成一个 string。

```
>>> "1" + "2"
'12'
>>> "hello " + "tommorow"
'hello tommorow'
>>>
```

也可以比较大小关系 （大小通过字母的先后顺序）

```
>>> "a" < "b"
True
>>> "abxsdhj" < "bsjdhks"
True
>>> "apple" > "append"
True
>>> "aa">"ab"
False
>>> "apple" == "apple"
True
```

学会用 help()

当我们想运用 pow 来算指数时，我们可以通过 help(pow)来知道它的具体 syntax。

```
>>> help(pow)
Help on built-in function pow in module __builtin__:

pow(...)
    pow(x, y[, z]) -> number

    With two arguments, equivalent to x**y.  With three arguments,
    equivalent to (x**y) % z, but may be more efficient (e.g. for longs).
```

当我们想要了解所有可以运用在 integer 上的 function

（以下是部分截图）

```
>>> help(5)
Help on int object:

class int(object)
 |  int(x=0) -> int or long
 |  int(x, base=10) -> int or long
 |
 |  Convert a number or string to an integer, or return 0 if no arguments
 |  are given.  If x is floating point, the conversion truncates towards zero.
 |  If x is outside the integer range, the function returns a long instead.
 |
 |  If x is not a number or if base is given, then x must be a string or
 |  Unicode object representing an integer literal in the given base.  The
 |  literal can be preceded by '+' or '-' and be surrounded by whitespace.
 |  The base defaults to 10.  Valid bases are 0 and 2-36.  Base 0 means to
 |  interpret the base from the string as an integer literal.
 |  >>> int('0b100', base=0)
 |  4
 |
 |  Methods defined here:
 |
 |  __abs__(...)
 |      x.__abs__() <==> abs(x)
 |
 |  __add__(...)
 |      x.__add__(y) <==> x+y
 |
```

variables

以上的计算我们只是让计算机执行指定命令，但是我们的计算机并不能记忆。
所以我们可以运用 variable 去建立一个 memory。

Form

<< variable >> = <<expression>>

how its executed?

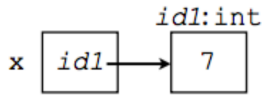
Evaluate the expression on the RHS to produce a value.
(相当于指定 variable 一个值)
Store that memory address in the variable on the LHS. (Create a new variable if it doesn't exist; otherwise just reuse the existing variable)

注意我们要把 variable 写在左边，如果写在右边的话也会发生 error

```
>>> a = 4
>>> 4 = a
    Traceback (most recent call last):
      Python Shell, prompt 35, line 1
    can't assign to literal: <string>, line 1
```

For the statements

x = 7



相当于 ： "x refers to the value 7"
" memory address id1 is stored in variable x"

```
>>> x = 7
>>> y = 8
>>> x
    7
>>> y
    8
>>> x + y
    15
```

id of data

当每一个 variable 被 assign 一个 value 时，这个 value 就会被存起来，那么我们要如何找到这个 value 呢，就会通过他的 id, 我们可以用 id(value) 这个 function 来找到每个 value 对应的 id. （画图）

（这个 value 被硬件 create 出来，存到了 memory 里面随时使用时，就会有 id）

但是并不是所有相等的 data 都 share 同一个 id, 比如 1 和 1.0

```
>>> id(1)
4297537888
>>> id(1.0)
4302351048
```

所以科学家们发明了两个等式来 compare 两个 data 或者 variable， 'is' and '=='

在这里呢，所有的计算机语言都是有一个小小的分别， 比如 id of 1.0 会变，这是为什么呢？

```
>>> id(1)
4297537888
>>> id(1.0)
4302351048
>>> id(1.0)
4302351096
```

因为每个计算机语言会分 primary data(stored in memory) 和 other data (not created yet, only stored when assigned to a variable).

Changing in variables

>>> x = 7
>>> x = 2
what will be the value of x now?

```
>>> x = 7
>>> x = 2
>>> x
2
>>>
```

note: the variable refers to the latest value assigned.

>>> x = 7
>>> y = 6
>>> x = y

```
>>> x = 7
>>> y = 6
>>> x = y
>>> x
6
>>> y
6
```

在上述情况下，如果我们又再一次变化 y 值会发生什么情况呢？
>>> x = 7

```
>>> y = 6
>>> x = y
>>> y = 2
```

```
>>> x = 7
>>> y = 6
>>> x = y
>>> y = 2
>>> x
    6
```

因为在这个时候，x 只是记录了 y 的 value。
有些情况下， x 会随着 y 的变化而变化（我们会在以后讨论 Class 时提到）

Variable Names

- must star with a letter (or underscore)
- can include letters, digits, and underscores, but nothing else.
- Case matters

```
>>> age = 15
>>> age
    15
>>> Age
    Traceback (most recent call last):
      Python Shell, prompt 3, line 1
    NameError: name 'Age' is not defined
```

example:
valid name: age, *age,* apple_1, Total_number_of_attendence.
invalid name: 1apple, apple 1, @name

通常情况下,
我们起名字时通常用一些更 make sense 的名字
列如，当我们想分别记录 Annie 的年龄 20 岁，Alex 的年龄 21 岁时
age_annie = 20
age_alex = 21
instead of
x = 20
y = 21
这样的好处在于当我们看到这个 variable 时，我们就知道它代表的是什么。
也方便别人去阅读你的 code。
Note: we can assign any type of data to a variable.

Part 3 – functions

我们知道计算机可以帮助人们做一些很繁琐的计算，但是去运行这些计算之前，我们必须要告诉它运算规则，从而来实现我们的目的。

Python build-in function

如以上我们讨论的 pow（），ord（），min（），max（）都是 python 自带的一些 function。

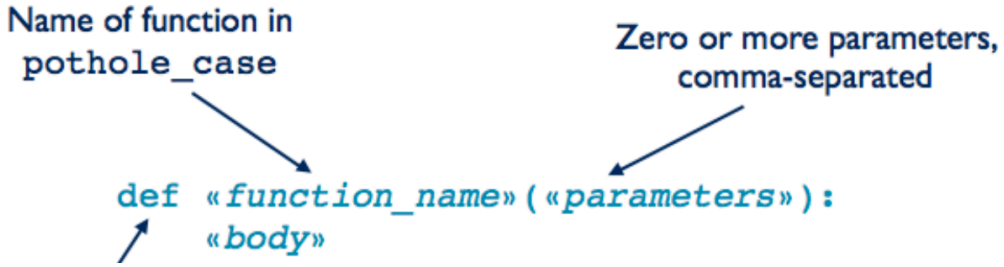
首先，我们来想一想它们背后的机制：
对于 pow（x, y, z）
我们相当于运算 (x ** y) % z

对于 min（x，y）
我们相当于比较 x 和 y，然后 return 出最小值。

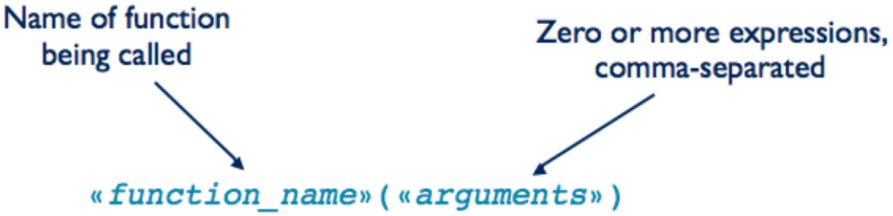
Design Function

首先我们需要在 Wing 里建一个新文件

- 对于一个新的 function



- 如果我们想要这个 function return the value.
  我们需要有一个 return statement（inside 《body》）
  Form：
  return <<expression>>
  how it's executed:
  Evaluate the expression. This produces a value (which has a memory address)
  Exit the function and produce that value to the caller.
- Function call (当我们想要运用这个 function 的时候)



注意：arguments 的数量一定要对于，否则会出现 error

举例：
rectangle_area(a,b):
a, b are side-lengths.

```
def rectangle_area(a,b):
    return a * b
```

when we want to call this function
firstly, we need to evaluate the file
then

```
>>> rectangle_area(5, 3)
15
```

注意：

1. ‘def’是 function identifier, 它与 function_name 之间有一个空行
2. Parameters 之前用逗号隔开
3. header 后面一定要有冒号
4. Body 一定要空四行表示 body 在 header 里面

homework
写一个 function 来计算圆柱体的体积。
Given parameter radius and height。
完成 exercise 1.