大学计算机入门课

Class #3

［教学目的］

- Know the function design recipe and can write the formal function

［课程大纲］

- Part1- function design recipe
- Part2 – more functions
- Part 3 – print() vs. return
- Part 4 – nested function
- Part 5 – Boolean function
- Part 6 – if statement.

part1 – function design recipe

上节课里，我们介绍了几个简单的 function。但是仅仅有 function body 与 header，很多阅读 code 的人并不容易理解这个 function 的作用以及用法。并且有时还不确定该如何确定我们 parameter 的 type。所以在这里， 我们要学习一下什么是 formal function structure。

Step 1: Examples

第一步，我们先要考虑我们的 function 想要达到什么样的效果。

如果我们想要只知道点（x，y）距离原点的距离。

注意：我们尽可能考虑到所有可能出现的情况，这样全面测试 便于提高我们 code 的正确性

```
>>> distance(0,0)
0.0
>>> distance(1,1)
1.4142135623730951
```

Step 2: type contract

我们要知道我们输入的 parameter 是什么 type，return 的是什么 type。

两种写法

```
@type x: number
@type y: number
@rtype: float
```

or

```
(number, number) -> float
```

step 3: header

```
def distance(x, y):
```

step 4: description

```
Return the distance from the point (x, y) to the origin.
```

step 5: Code the Body

```
    return (x ** 2 + y ** 2) ** 0.5
```

step 6: Test

在我们文件最下面写一个关于 doctest 的文件，能够帮助我们 test 我们写的所有 example。

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

now, combining all the steps, we will get:

```
def distance(x, y):
    """
    (number, number) -> float

    Return the distance from the point (x, y) to the origin.

    >>> distance(0,0)
    0.0
    >>> distance(1,1)
    1.4142135623730951
    """
    return (x ** 2 + y ** 2) ** 0.5

if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

当我们 evaluate 我们的文件时，出现以下形式证明我们所写的所有 examples 都通过。

```
>>> [evaluate functions_1.py]
```

但是如果出现：

```
**********************************************************************
File "/Applications/WingIDE.app/Contents/Resources/src/debug/tserver/_sandbox.py",
Failed example:
    distance(1,7)
Expected:
    1.4142135623730951
Got:
    7.0710678118654755
**********************************************************************
1 items had failures:
   1 of   2 in __main__.distance
***Test Failed*** 1 failures.
```

说明我们有出现问题的 example，要回去检查修改，直到所有都通过的时候，这就是为什么我们要全面考虑我们的 example。

其中 step1，2，4 是 **documentation strings**。

Part2 – more functions

- Function about string.
  If we want to return a particular word by n times.

```
def repeat_word(word, num_times):
    """ (str, int) -> str

    Return word repeated num_times.

    >>> repeat_word('Marcia ', 3)
    'Marcia Marcia Marcia '
    >>> repeat_word('Monday', 0)
    ''
    """
    return word * num_times
```

- Number_ of_cents.
  Let's follow the documentation string to write the function body.

```
def number_of_cents(change):
    """ (number) -> float

    Return the number of cents left when you take away the dollar
    amount from change.

    >>> number_of_cents(1.25)
    25.0
    >>> number_of_cents(20.00)
    0.0
    """

    dollar_remainder = change % 1
    cents = dollar_remainder * 100
    return cents
```

Body:

```
    dollar_remainder = change % 1
    cents = dollar_remainder * 100
    return round(cents)
```

note: don't forget to test every single function in your python shell.

- Calculate total value after tax.
  Note: we know 0.0 <= tax_rate <= 1.0
  So, we need to add precondition in our documentation strings.
  (在这里，我们需要限制我们给的 parameter 的值。)

```
def calculate_total_value(bill, tax_rate):
    """(number, number) -> number

    precondition: 0.0 <= tax_rate <= 1.0

    Return the amount of bill and the tax to be paid on bill for tax rate tax_rate

    >>> calculate_total_value(5.00, 0.1)
    5.5
    >>> calculate_total_value(100.00, 0.05)
    105.0
    """
    return bill + bill * tax_rate
```

Part 3 – print() vs. return

Let define two functions:

```
def return_word(word):
    return word

def print_word(word):
    print(word)
```

注意 print 后一定要有括号，return 没有。

and run them:

```
>>> return_word("hi")
'hi'
>>> print_word("hi")
hi
```

let's try something more:

```
>>> a = return_word("hi")
>>> b = print_word("hi")
hi
>>> a
'hi'
>>> b
```

现在，我们来讨论一下 print 和 return 的区别

return 可以建立一个 memory address。当我们运行完一个 return statement 时，我们将会终止运行这个 function。

Print 不能建立一个 memory address， 我们就把 print 看做 code 的普通代码的一部分，它的功能就是将 word printing 出来。当我们运行完一个 print statement 时，我们还会继续运行这个。

所以当我们运行我们的 code 时，他会一步一步走，如果见到 print statement，我们就 print， 然后继续我们的 code。

所以，

```
def return_word(word):
    """(str) -> str
    """
    return word

def print_word(word):
    """(str) -> None
    """
    print(word)
```

考虑

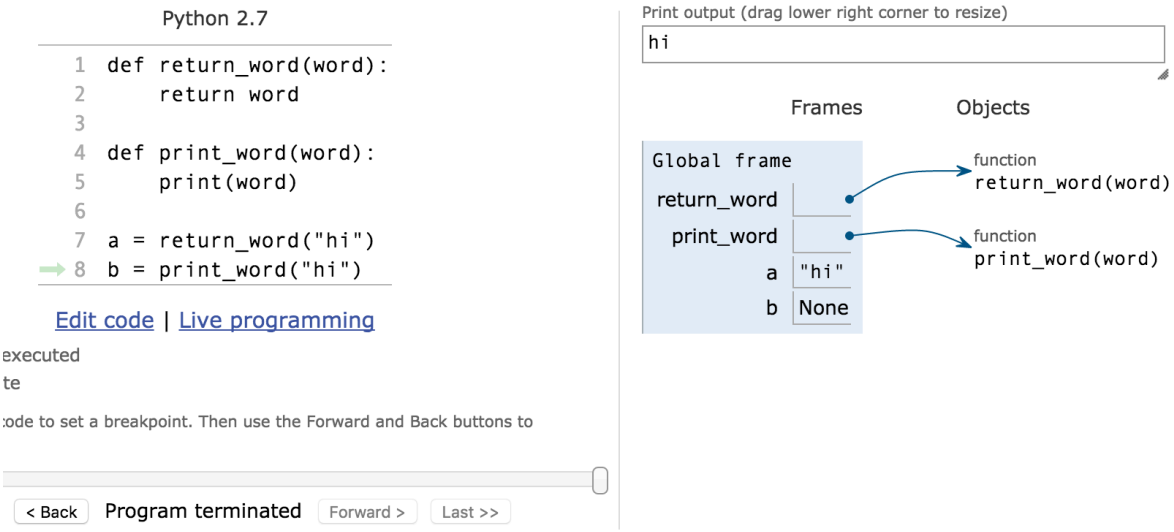
```
def r():
    print("start to run r()")
    return 5
```

当我们 run r() in the python shell, we will get:

```
>>> a = r()
start to run r()
>>> a
5
```

we can see more clearly in the python visualize
http://pythontutor.com/visualize.html#mode=edit

visualizing our first example:

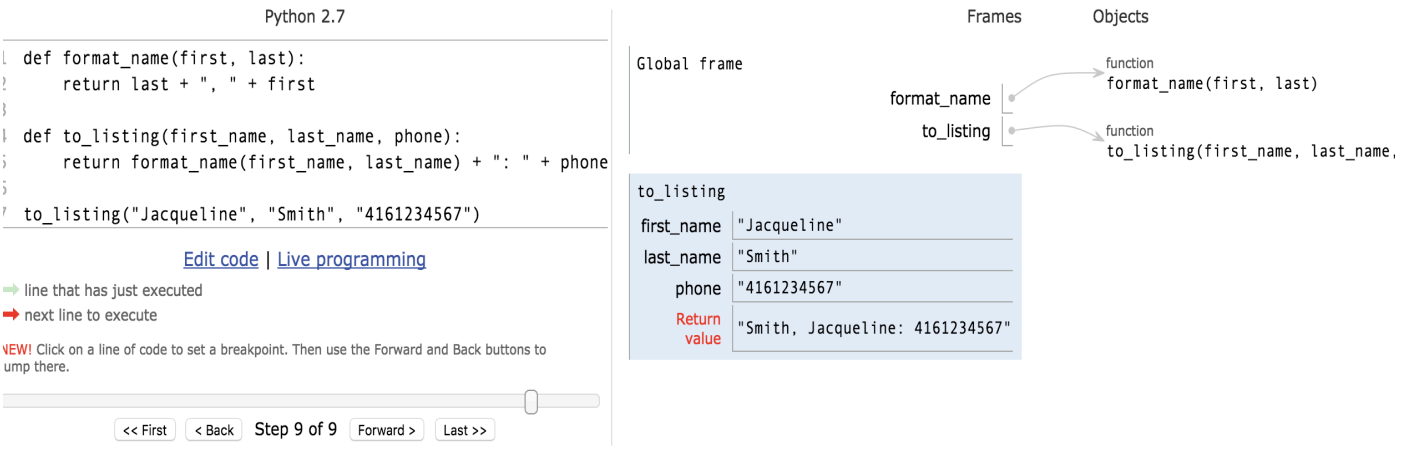


Part 4 – nested function

一个 function A 的 body 里可以含有另外一个 function B.
这样，A 就是一个 nested function。
列子：

```
def format_name(first, last):
    """ (str, str) -> str

    Return the name formatted as "last, first".

    >>> format_name("Jacqueline", "Smith")
    'Smith, Jacqueline'
    >>> format_name("Justin", "Bieber")
    'Bieber, Justin'
    """
    return last + ", " + first

def to_listing(first_name, last_name, phone):
    """ (str, str, str) -> str

    Return the listing formatted as
    last_name, first_name: phone

    >>> to_listing("Jacqueline", "Smith", "4161234567")
    'Smith, Jacqueline: 4161234567'
    >>> to_listing("Justin", "Bieber", "4161122334")
    'Bieber, Justin: 4161122334'
    """
    return format_name(first_name, last_name) + ": " + phone
```

visualize the nested function:



Part 5 – Boolean function

Recall:

Syntax	Meaning
a == b	Return True if a equals to b, and False otherwise
a != b	Return False if a equals to b, and True otherwise
a < b	Return True if a less than b, and False otherwise
a > b	Return True if a greater than b, and False otherwise
a <= b	Return True if a less than or equals to b, and False otherwise
a >= b	Return True if a greater than or equals to b, and False otherwise

如果我想要知道一个人是否到法定年龄喝酒，我们想要一个 function return True or False. 这样的 function 就叫做 boolean function。

```
def can_drink(age):
    """(int) -> bool

    precondition: age >= 0

    Return True if age is 19 or more, and False otherwise.

    >>> can_drink(21)
    True
    >>> can_drink(16)
    False
    """

    return age >= 18
```

And more for strings,

```
>>> "i" in "hi"
True
>>> "7" in "hello"
False
```

我们还可以知道每个具体位置是什么 char。
注意：第一位的 index 为 0，依次类推。

```
>>> a = "hello"
>>> a[0]
'h'
>>> a[1]
'e'
>>> a[2]
'l'
>>> a[3]
'l'
>>> a[4]
'o'
>>> a[5]
Traceback (most recent call last):
  Python Shell, prompt 12, line 1
IndexError: string index out of range
>>> a[-1]
'o'
>>> a[-2]
'l'
>>> a[-3]
'l'
```

还可以知道一个 string 的长度：

```
>>> len("hello")
5
```

注意：在这里长度为 5， 但是 index 为 0，1，2，3，4.

我们还可以知道 string 的某一部分是什么：

```
>>> a = "hello"
>>> a[0:2]
'he'
>>> a[0:len(a)]
'hello'
>>> a[2:4]
'll'
```

```
def is_start_with_digit(s):
    """(str) -> bool
    |
    precondition: len(s) > 0

    Return True if s is starting with a digit.

    >>> is_start_with_digit("2he")
    True
    >>> is_start_with_digit("hi2")
    False
    """
    return (s[0] in "1234567890")
```

Part 6 – if statement

现在我们要考虑一个 function 要对于不同的情况实行不同的规则。

```
def description(x):
    if x > 0:
        print("I'm a positive number")
    elif x == 0:
        print("I'm Zero")
    else:
        print("I'm a negative number")
```

run it：

```
>>> description(5)
    I'm a positive number
>>> description(0)
    I'm Zero
>>> description(-5)
    I'm a negative number
```

格式：

if … (is True):
  doA()
elif … (else if … is True):
  doB()
else:
  doC()

对比 if and elif：

```
def description_str(word):
    if is_start_with_digit(word):
        print("I'm starting with digit")
    if len(word) == 5:
        print("I have length of 5!")
```

```
>>> description_str("1hihi")
    I'm starting with digit
    I have length of 5!
```

```
def description_str(word):
    if is_start_with_digit(word):
        print("I'm starting with digit")
    elif len(word) == 5:
        print("I have length of 5!")
```

```
>>> description_str("1hihi")
I'm starting with digit
```

we can also visualize it! Try it!