

2.3 Function Files Dictionaries

- Function

- defining functions: A named sequence of statements.
 - 用def
 - def hello():
 - `"""This function says hello and greets you"""`
 - `print("Hello")`
 - `print("Glad to meet you")`
 - `hello()`
 - 输出
 - Hello
 - Glad to meet you
 - 一定要有冒号和缩进! 用三引号因为可以跨行代码
 - `print(type(hello))`
 - 输出 `<class 'function'>`
- positional parameter passing
 - 有另外的variable在括号中 叫formal parameter 或者parameter names input parameters
 - 1 `def hello2(s):`
 - 2 `print("Hello " + s)`
 - 3 `print("Glad to meet you")`
 - 4
 - 5 `hello2("Iman")` "Iman"就是parameter value
 - 6 `hello2("Jackie")`
 - 如果 `hello2("Class " * 3)`
 - 输出
 - Hello Class Class Class
 - Glad to meet you
 - 如果是两个变量的话
 - 1 `def hello3(s, n):`
 - 2 `greeting = "Hello {}".format(s)`
 - 3 `print(greeting*n)`
 - 4

- 5 hello3("Wei", 4)
- 6 hello3("", 1)
- 7 hello3("Kitty", 11)
- 输出是Hello Wei重复4遍 Hello Kitty重复11遍

- 例子

- def cyu(s1, s2):
- if len(s1) > len(s2):
- print(s1)
- cyu("Hello", "Goodbye")
- print(s2)
- else:
- 输出Goodbye

- Returning Values

- 例子

- def square(x):
- y = x * x
- return y
- toSquare = 10
- result = square(toSquare)
- print("The result of {} squared is {}".format(toSquare, result))
- 定义了一个square function 是y=x*x 然后要return y值; 定义toSquare是10, 然后result=... 所以相当于result=square(10)带入square方程 returny值是100 再打印format代入句子
- 如果不return的话 result就会是None 一旦return了之后 如果冒号底下return后面 还有需要运作的东西就不会继续下去了 就直接会运行其他的代码
- return其实是个function

- y = x * x
- return y
- toSquare = 10
- result = square(toSquare)
- print("The result of {} squared is {}".format(toSquare, result))

- a function that accumulates

- function中包含function
- def total(lst):
- tot=0
- for num in lst:

- tot=tot+num
- return tot
- y=tot([1,5,7])
- print(y)
- function composition 组合
 - 给每个char赋值并找到最多的那个

```

1 def most_common_letter(s):
2     frequencies = count_freqs(s)
3     return best_key(frequencies)
4
5 def count_freqs(st):
6     d = {}
7     for c in st:
8         if c not in d:
9             d[c] = 0
10            d[c] = d[c] + 1
11    return d
12
13 def best_key(dictionary):
14     ks = dictionary.keys()
15     best_key_so_far = list(ks)[0] # Have to turn ks into a real list
16     for k in ks:
17         if dictionary[k] > dictionary[best_key_so_far]:
18             best_key_so_far = k
19     return best_key_so_far
20
21 print(most_common_letter("abbbbbbbbbbbccccccccccccccccccddddd"))
22

```

- 拿到问题的时候要decompose 问题 首先要拆解 然后分开编function
- 就算第二行中的count_freq(s)在之前没有被定义 但是它是inside function的 所以没关系
- mutable objects and side effects
 - 如果 function makes a change to immutable object like a list or a dictionary 一个函数对不可变的对象比如说list or dictionary更改
 - 这里是说y=5但是 是global的 你对local怎么定义y的值并不影响global的值是5 所以这个print y出来的不是10而是5这里的local y和global y不是同一个东西 只不过恰好value一样

Python 3.6

```

1 def double(y):
2     y = 2 * y
3
4 def changeit(lst):
5     lst[0] = "Michigan"
6     lst[1] = "Wolverines"
7
8 y = 5
9 double(y)
10 print(y)
11
12 mylst = ['our', 'students', 'are']
13 changeit(mylst)
14 print(mylst)

```

Print output (drag lower right corner to resize)

5

Frames

Global frame

double

changeit

y 5

mylst

Objects

function double(y)

function changeit(lst)

list

0 "our" 1 "students" 2 "are" 3 "awesome"

→ line that has just executed

→ next line to execute

Python 3.6

```

1 def double(y):
2     y = 2 * y
3
4 def changeit(lst):
5     lst[0] = "Michigan"
6     lst[1] = "Wolverines"
7
8 y = 5
9 double(y)
10 print(y)
11
12 mylst = ['our', 'students', 'are']
13 changeit(mylst)
14 print(mylst)

```

Print output (drag lower right corner to resize)

Global frame

double

changeit

y 5

function double(y)

function changeit(lst)

double

y 10

Return value None

→ line that just executed

→ next line to execute

Step 7 of 15

[Python Tutor by Philip Guo](#)

[Customize visualization \(NEW!\)](#)

Activity: 1 -- ActiveCode (ac11_12_1)

- 上图进行到这里的时候 return value是None
- variable是local的 但是objects不是 所以上面的例子就是 running double并没有change y但是running changeit却change了mylst 因为objects share了mutable objects
- 所以changeit这个function就是有side effect 他change了像list dic这样不变的东西
- 像这样就是change了global的y了

main() has a side effect on the global variable y.

Python 3.3

```

1 def double(n):
2     global y
3     y = 2 * n
4
5 y = 5
6 double(y)
7 print(y)

```

Step 6 of 7

line that has just executed
next line to execute

Frames

Global frame

double	
y	10

double

n	5
Return value	None

Objects

function double(n)

- 这样就不会change自己list的东西

Python 3.3

```

1 def changeit(lst):
2     lst[0] = "Michigan"
3     lst[1] = "Wolverines"
4     return lst
5
6 mylst = ['106', 'students', 'are', 'awesome']
7 newlst = changeit(list(mylst))
8 print(mylst)
9 print(newlst)

```

Program terminated

line that has just executed
next line to execute

Visualized using Online Python Tutor by Philip Guo

Frames

Global frame

changeit	
mylst	
newlst	

function changeit(lst)

list

0	1	2	3
"106"	"students"	"are"	"awesome"

list

0	1	2	3
"Michigan"	"Wolverines"	"are"	"awesome"

Objects

• Tuple

- tuple packing
 - julia = ("Julia", "Roberts", 1967, "Duplicity", 2009, "Actress", "Atlanta, Georgia")和julia = "Julia", "Roberts", 1967, "Duplicity", 2009, "Actress", "Atlanta, Georgia"是一样的
 - 只要python期望一个值，如果提供了多个表达式，用逗号分隔，它们就会自动打包成一个tuple。
 - 例子
 - def circleInfo(r):
 - """ Return (circumference, area) of a circle of radius r """
 - c = 2 * 3.14159 * r
 - a = 3.14159 * r * r
 - return (c, a)和return c,a一样的效果 python都会自动识别输出(,)
 - print(circleInfo(10))
 - 输出(62.8318, 314.159)
 -

Define a function called `information` that takes as input, the variables `name`, `birth_year`, `fav_color`, and `hometown`. It should return a tuple of these variables in this order.

Save & Run 2020/3/9 上午10:37:18 - 6 of 6 Show in CodeLens

```

1 tuple=(name, birth_year, fav_color, hometown)
2 def information(tuple):
3     return tuple

```

Activity: 5 -- ActiveCode (ac12_3_3)

Result	Actual Value	Expected Value	Notes
ERROR	None	None	Error: TypeError: information() takes exactly 1 arguments (4 given)

You passed: 0.0% of the tests

13.5. Unpacking Tuples a:

- tuple unpacking

- 例子

- `julia="Julia", "Roberts",1967,"Duplicity",2009,"Actress","Atlanta,Georgia"`
 - `name, birth_year, movie,movie_year,profession,birth_place=julia`
 - 后面这句话的意思就是 `name=julia[0]` `surname=julia[1]`...

- 例子

- `a = 1`
 - `b = 2`
 - `(a, b) = (b, a)`
 - `print(a, b)`
 - 输出2 1

- 例子

- `authors = [('Paul', 'Resnick'), ('Brad', 'Miller'), ('Lauren', 'Murphy')]`
 - `for first_name, last_name in authors:`
 - `print("first name:", first_name, "last name:", last_name)`
 - 输出
 - first name: Paul last name: Resnick
 - first name: Brad last name: Miller
 - first name: Lauren last name: Murphy
 - On the first iteration the tuple `('Paul', 'Resnick')` is unpacked into the two variables `first_name` and `last_name`.

- enumerate枚举法
 - fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
 - for n in range(len(fruits)):
 - print(n, fruits[n])
 - 输出结果和
 - fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
 - for item in enumerate(fruits):
 - print(item[0], item[1]) 一样
 - 和
 - fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
 - for idx, fruit in enumerate(fruits):
 - print(idx, fruit)一样
 - 都是
 - 0 apple
 - 1 pear
 - 2 apricot
 - 3 cherry
 - 4 peach
- v1,v2,v3,v4=1,2,3,4就是一行让每个互相对应的例子
- 例子 items把dic转换为tuple
 - pokemon = {'Rattata': 19, 'Machop': 66, 'Seel': 86, 'Volbeat': 86, 'Solrock': 126}
 - a=pokemon.items()
 - p_names=[]
 - p_number=[]
 - for name,num in a:
 - p_names=p_names+[name]
 - p_number=p_number+[num]
- 不会的

The `.items()` method produces a sequence of key-value pair tuples. With this in mind, write code to create a list of keys from the dictionary `track_medal_counts` and assign the list to the variable name `track_events`. Do **NOT** use the `.keys()` method.

Save & Run

2020/3/9 上午11:08:16 - 6 of 6

Show in CodeLens

```
1
2 track_medal_counts = {'shot put': 1, 'long jump': 3, '100 meters': 2, '400 meters': 1}
3
4 a=track_medal_counts.items()
5 track_events=[]
6
7 for key in track_medal_counts:
8
9     track_events=track_events+[key]
```

Activity: 13 -- ActiveCode (ac12_4_12)

Result	Actual Value	Expected Value	Notes
Pass	['100...ump']	['100...ump']	Testing that track_events was created correctly.
Pass	'.keys()'	"\ntrac...[key]"	Testing your code (Don't worry about actual and expected values).
Pass	'.items()'	"\ntrac...[key]"	Testing your code (Don't worry about actual and expected values).
Fail	'in tr...unts:'	"\ntrac...[key]"	Testing your code (Don't worry about actual and expected values).

Expand Differences

Expand Differences

Expand Differences

Expand Differences

You passed: 75.0% of the tests

- 例子

- `def add(x, y):`

The `.items()` method produces a sequence of key-value pair tuples. With this in mind, write code to create a list of keys from the dictionary `track_medal_counts` and assign the list to the variable name `track_events`. Do **NOT** use the `.keys()` method.

Save & Run

2020/3/9 上午11:08:16 - 6 of 6

Show in CodeLens

```
1
2 track_medal_counts = {'shot put': 1, 'long jump': 3, '100 meters': 2, '400 meters': 1}
3
4 a=track_medal_counts.items()
5 track_events=[]
6
7 for key in track_medal_counts:
8
9     track_events=track_events+[key]
```

Activity: 13 -- ActiveCode (ac12_4_12)

Result	Actual Value	Expected Value	Notes
Pass	['100...ump']	['100...ump']	Testing that track_events was created correctly.
Pass	'.keys()'	'\ntrac...[key]'	Testing your code (Don't worry about actual and expected values).
Pass	'.items()'	'\ntrac...[key]'	Testing your code (Don't worry about actual and expected values).
Fail	'in tr...unts:'	'\ntrac...[key]'	Testing your code (Don't worry about actual and expected values).

Expand Differences

Expand Differences

Expand Differences

Expand Differences

You passed: 75.0% of the tests

- `return x + y`
- `print(add(3, 4))`
- `z = (5, 4)`
- `print(add(z))` # this line causes an error因为会把z当成一个变量而不是一个tuple
- 需要在z前加一个`*print(add(*z))` 让python认出来z是一个tuple
- 例子
 -

• Variables

- variable scoping and side effects
 - local and global variable
 - scope就是python查找的范围?
 - python space有两种
 - 一种是 name space
 - built-in scope 本来就有的 像print len这种function
 - global scope 比如我们自己弄的function
 - local scope 我们自己assign的variable

- 一种是 object space
- 举个例子
 - `some_var=1`
 - 其实是create一个`some_var`在global scope 然后link这个`some_var`去 objects space的1
 - 又比如我要弄一个`some_fun(p1,p2)` function 同时给了3和4这两个值：然后会马上有`v1`和`v2`两个local值创立 之后自己又会自动消失 每次python找东西都会先去local 再global



- 如果`v2`存在both local and global python会自动去local里链接而不会去global里
- 找local: A temporary variable that is only used inside a function
 - `numbers = [1, 12, 13, 4]`
 - `def foo(bar):`
 - `aug = str(bar) + "street"`
 - `return aug`
 - `addresses = []`
 - `for item in numbers:`
 - `addresses.append(foo(item))`
 - local是`bar` 和 `aug`