Functions

Python Open Lab

What is a function

- In simple terms, a *function* is a device that groups a set of statements so they can be run more than once in a program—a packaged procedure invoked by name.

- Functions also can compute a result value and let us specify parameters that serve as function inputs and may differ each time the code is run

Functions we have used

- So many…
- len(list)
- print(list)
- list.insert(index, value)
- list.find(x)
- …

Why we need a function

- Maximizing code reuse
    - function A()
    - function B():
        - call A()
    - function C():
        - call A()
- Minimizing redundancy

Why we need a function

- Procedural decomposition
- Task: make a piazza
  - mix the dough — def mixDough()
  - roll it — def roll()
  - add topping — def addTopping()
  - bake — bake()

Declaration

- function definition
- def function(parameter1, parameter2…):

    do something

    return value

Example

- print hello world

```
def printHelloWorld():

    print(“hello world”)

printHelloWorld()
```

The third line is very important : to call the function

Use after declaration

- What will happen if we do in this way?

```
printHelloWorld()

def printHelloWorld():

    print(“hello world”)
```

Use parameter

- use a single parameter

```
def function(param):
    do something with param


#call the function
function(“a”)
function(1)
function(<a list>)
```

Use parameters

- use multiple parameters

```
def function(param1, param2):
    do something with params


#call the function
function(“a”, “b”)
function(1, 2)
function(<a list>, <a dictionary>)
```

- the parameters passed are retrieved by order in the function

Example

- Calculator

```
def addTwoNumber(x,y):
    num1 = float(x)
    num2 = float(y)
    print(num1+num2)
def minusTwoNumber(x,y):
    num1 = float(x)
    num2 = float(y)
    print(num1-num2)

#call them
addTwoNumber(1.5, 2.3)
minusTwoNumber(1.5. 2.3)
```

Exercise

- Implement your print() function

- Use a name other than print()

Return value

- Pass a task to a function
- Want to know the result of the task
  - We can use print()
  - But what if we need the result to do next step calculation?

Return value

```
def function(param1, param2):

    do something with param2

    return result

result = function(x,y)

print(result)
```

Two sums

- Add two numbers and return their sum
- Add another two numbers and return their sum
- Compare two sums and print the bigger one

Two sums

```
def sum(x, y):
    return float(x)+float(y)

num1 = sum(1.0, 2.5)    #num1 = 3.5
num2 = sum(2.4, 1.6)    #num2 = 4.0
if num1 > num2:
    print(num1)
else:
    print(num2)
```

Two sums

```
def sum(x, y):
    return float(x)+float(y)
def compareNums(x, y):
    e1 = float(x)
    e2 = float(y)
    if e1 > e2:
        print(e1)
    else:
        print(e2)
num1 = sum(1.0, 2.5)    #num1 = 3.5
num2 = sum(2.4, 1.6)    #num2 = 4.0
compareNums(num1, num2)
```

Main function

- The entrance of program
  - if _ _ main _ _ == “_ _ main _ _ ”:
    - do something
- Advantage of using main function is that we organize all code in functions

Main function

- Make pizza
- if _ _ main _ _ == “_ _ main _ _ ”:

    mixDough()

    roll()

    addTopping()

    bake()

- Function is the best way to organize your code!

Exercise

- Build a calculator which supports add, subtract, multiplication, division and return related result
- num1 = add(1.9,2.3)
- num2 = minus(num1, 3.4)
- num3 = multiple(num2, 1.5)
- num4 = divide(num3,2.0)

Reference

- Learning Python(Fifth Edition, Mark Lutz)
  - Chapter 16, pp. 473-478