Learn Python Programming

-Functions

TCEF IT Club, 2018

Class Agenda

- Learn Python Functions
- ASCII code
- Revisit String, condition, loop, list, set, dict

What is function?

- Bundle a set of instructions together.

- Benefits :
  - Reuse (Reduce coding time)
  - Make code more readable
  - Reduce debugging time
  - Easier to maintain

How to define a function?

- Function blocks begin with the keyword **def** followed by the function name and parentheses ( ).
- The code block within every function starts with a colon (:) and is indented.
- input parameters or arguments: should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Examples

```
def hello_func():
        pass

hello_func()
```

```
def hello_func():
        print(“hello world!”)

Hello_func()
```

```
def hello_func()
        print(“hello world!”)

hello_func()
```

Best Practice 1 – DRY

Don’t repeat yourself

Why?
When you write code that performs the same tasks over and over again, any modification of one task requires the same change to be made to every single instance of that task! Editing every instance of a task is a lot of work.

#example:print out hello to welcome each student in the dancing class, which only has 4 students.

```
print(“hello, Lily”)
print(“hello, Bill”)
print(“hello, Rick”)
print(“hello, Jim”)
```

- #now, the dancing class is growing, we have 20 students now… what to do?

Function - return

```
#now, let’s change the function a little bit:

def hello_func(name):
   return ("hello,"+ name)

print(hello_func(“Lily”))
```

Function - parameter

```
#now, the dancing class is growing, we have 10 students
now… what to do?

def hello_func(name):
    print("hello,"+ name)

students = ["Lily", "Bill", "Rick", "Jim", "Kevin", "Sarah",
"Monica", "Jill", "Jack", "Eva"]

for name in students:
    hello_func(name)

#now, I want to change “hello” to “welcome”. What to do?
```

Function – default parameter

```
#now, what if I forget to pass the name?

def hello_func(name = ‘John’):
    print("hello,"+ name)

hello_func()
```

Function – keyword parameter

```
def hello_func(name, age):
   print("hello,"+ name + age)

hello_func(name = “John”, age = “10”)
```

Function – variable-length parameter

```
#now, you may need to process a function for more parameters than you specified

def hello_func(school, *names):
    print("School name: "+ school)
    print("Students: ")

    for n in names:
        print (n)

hello_func("Green Hope","John", "Lily", "Kevin")
```

Variables

- Local variables: define inside functions, can’t be accessed from outside
- Global variables: define outside functions, can be accessed inside function

If you want to define global variables inside function, use key work global

```
#local variables and global variables
total = 0;

def sum(var1, var2):
   total = var1 + var2;
   print ("Inside function local total is:", total)
   return total

sum(10, 20)
print ("Outside function global total is:", total)
```

Variables

- Local variables: define inside functions, can’t be accessed from outside
- Global variables: define outside functions, can be accessed inside function

If you want to define global variables inside function, use key work global

```
#local variables and global variables
total = 0;

def sum(var1, var2):
   total = var1 + var2;
   print ("Inside function local total is:", total)
   return total

sum(10, 20)
print ("Outside function global total is:", total)
```

Practice:

```
#practice: given a year and month, print out how many days of the month of that
year.
month_days = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

def is_leap(year):
    """ Return True for leap years, False for non-leap years."""
    return (year%4==0) and (year%100 !=0 or year%400 ==0)

def days_in_month(year, month):
    """Return number of days in that month in that year."""
    if not 1<=month <=12:
        return "Invalid Month"
    if month==2 and is_leap(year):
        return 29

    return month_days[month]

days_in_month(2016, 2)
```