Function in Python

Jinho Choi
jinho.choi@colorado.edu
Functions

■ What is function?
   \( f(x) = 2.54 \cdot x \)

■ Why functions?
   inch1 = 1; inch2 = 2
   cm1 = 2.54 \cdot inch1; cm2 = 2.54 \cdot inch2
   cm1 = f(inch1); cm2 = f(inch2)
   cm1 = inch2cm(inch1)
Functions

- \( ls = [1, 2, 3] \)
- \( \text{sum} = \text{max} = \text{min} = 0 \)
- for item in ls:
  - \( \text{sum} += \text{item} \)
  - if item > max: max = item
  - if item < min: min = item
- print sum, max, min
Functions

- \(ls1 = [1, 2, 3]\)
  sum = max = min = 0
  for item in ls:
    sum += item
    if item > max: max = item
    if item < min: min = item
  print sum, max, min

- \(ls2 = [4, 5, 6]\)
  sum = max = min = 0
  for item in ls:
    sum += item
    if item > max: max = item
    if item < min: min = item
  print sum, max, min

- \(ls3 = [7, 8, 9]\)
  sum = max = min = 0
  for item in ls:
    sum += item
    if item > max: max = item
    if item < min: min = item
  print sum, max, min

Code reuse

Code organization
Defining Functions

Function definition begins with “def.”

```
def inch2cm(inch):
    'inch to cm'
    line1
    line2
    return 2.54 * inch
```

Function name and its arguments.
The keyword ‘return’ indicates the value to be sent back to the caller.

The indentation matters…
First line with different indentation is considered to be outside of the function definition.

Colon.
Calling a Function

- The syntax for a function call is:
  ```python
  >>> def inch2cm(inch):
       return 2.54 * inch
  >>> inch2cm(2)
  5.08
  ```

- Parameters in Python are “Call by Assignment.”
  - Sometimes acts like “call by reference” and sometimes like “call by value” in C++. Depends on the data type.
  - We’ll discuss mutability of data types later: this will specify more precisely how function calls behave.
Functions without returns

- All functions in Python have a return value, even ones without a specific “return” line inside the code.

- Functions without a “return” will give the special value `None` as their return value.
  - None is a special constant in the language.
  - None is used like `null`, `void`, or `nil` in other languages.
  - None is also logically equivalent to `False`.
Argument passing and mutability

```python
>>> myint = 5
>>> def add5(x):
    x += 5
    return x

>>> add5(myint)
>>> ?
>>> myint
>>> ?
```
Argument passing and mutability

```python
>>> mylist = [4, 5, 6]
>>> def add5(list):
    for j in range(3):
        list[j] += 5
    return list

>>> add5(mylist)
>>> ?
>>> mylist
>>> ?
```
Variable scope: LEGB

- Local (inside a function)
- Enclosing scope (enclosing functions)
- Module (within a file)
- Built-in
Variable scope

```python
>>> hisname = "Jack"
>>> def newname():
    hisname = "Charles"
    print hisname

>>> newname()
?
>>> hisname
?
```

Two variables with the same name: Scope determined at assignment time
Variable scope

```python
>>> hisname = "Jack"
>>> def newname1():
    #hisname = "Charles"
    print hisname
>>> newname1()
>>> ?
Search from the innermost scope to the outermost scope if variables share the same name
```
Variable scope

```python
>>> hisname = "Jack"
>>> def newname2():
    myname = "Charles"
    print myname

>>> newname2()

>>> ?

>>> ?

>>> myname

>>> ?

Can't access local variables from outside the function
```
Variable scope

```python
>>> hisname = "Jack"
>>> def newname2():
    global myname
    myname = "Charles"
    print myname

>>> newname2()
>>> ?
>>> myname Can access global variables from outside the function
>>> ?
```
Variable scope

- Scope determined by assignment
- Local variables inside a function is destroyed once outside the scope (unless explicitly declared `global`)
- Search a variable from the innermost scope to the outermost scope
- From inside a function can access variables outside the function, but not the other around
Polymorphism

```
>>> def times(x, y):
    print x * y
>>> times(3, 3)
>>> ?
>>> times("wolf! ", 3)
>>> ?
>>> times((‘a’, ‘b’, ‘c’), 3)
>>> ?
>>> times("cry", "wolf")
>>> ?
```

- Polymorphism = one function does more than one thing
Exercise

- Write a function that generates a list of beginIdx, endIdx, and gap.
  - ex) l1 = listGen(0, 10, 2) -> [0, 2, 4, 6, 10]
  - ex) l2 = listGen(0, 10, 3) -> [0, 3, 6, 9]

- Write a function that returns the average of the list
  - ex) avg1 = average(l1) -> 4.4
  - ex) avg2 = average(l2) -> 4.5