Basics of Programming

L05: Functions

Aug/Sep, 2019

Dr. Ram P Rustagi
Professor, CSE Dept
KRP, KSGI
rprustagi@ksit.edu.in

Resources and Acknowledgements

- https://www.python-course.eu/passing_arguments.php
- A first course in programming
  - https://introcs.cs.princeton.edu/python/home/
  - https://introcs.cs.princeton.edu/java/home/
- Python for everybody
  - https://www.py4e.com
- Turtle Graphics
  - https://docs.python.org/3/library/turtle.html

Function: Parameter Passing

- Two common strategies
  - Call by value (aka pass by value)
  - Call by reference (aka pass by reference)
- Call by value
  - Used in C/C++
  - used in Java/python for primitive variables
  - A local copy of variable/expression is passed
  - Changes can be made locally,
    - but does not affect original variable on return

Function: Parameter Passing

- Call by reference
  - The invoked function gets an implicit reference to the argument.
    - does not get the copy of the argument.
    - Any changes made locally
      - affect the original variable on return
    - Thus, funtion can modify the argument and it is reflected back on the argument in the calling function
  - Advantages & disadvantages
    - Time and space efficiency - no copying of arguments
      - Accidental changes in the function can be catastrophic
- C/C++: Supported via means of passing address
- Java: automatic passes a copy of reference to the object.

Parameter Passing in Python

- Call by value
  - Used for primitive types
- Call by object reference (similar to java)
  - used for non-primtive types
  - A copy of reference is provided,
  - immutable objects can't be changed.
    - will result in error on changing
    - e.g. tuples
  - mutable objects can be changed
    - lists, (e.g. string), dictionary etc.

Example: Pass by Obj Reference

```
def ops(obj):
   // could be any object e.g. list
   obj.append("new")  #modifies orig obj
   obj = [None,None]  # replaces the obj ref.
   // orig obj remain as before including modification

list=["a","b","c"]
ops(obj)
print(obj)  # print orig list + "new"
```

Pass by Obj Reference: string

```
def ops(str):
    // str in an immutable object.
    str.append("new")  #modifies orig obj

list=["a","b","c"]
ops(list)  # works fine
print(list) #a, b, c, new
mystr="cse"
ops(mystr)    # gives error, string is immutable.
```

Pass by Obj Reference: string

- ops(list)

Function as Arguments

- function name is no different than other argument

```
def mops(fn,a,b):
  return fn(a,b)

def add(a,b):
  return a+b

def mult(a,b):
  return a*b

mops(add,5,10)
mops(mult,5,10)
```

Home Work

- `H01:`

Questions