

# Case Study: Python

## Introduction

Dr. Nguyen Hua Phung

HCMC University of Technology, Viet Nam

08, 2020

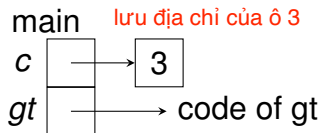
- popular programming language
- created by Guido van Rossum, released in 1991.
- used in many areas: web, software, big data, system scripting,...
- originally implemented in hybrid model
- readability:
  - new line to complete a command
  - indentation to define scope
- static scoping and dynamically typed

## Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```

# Example

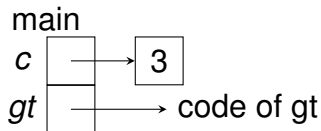
```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



không cho phép can thiệp tới địa chỉ

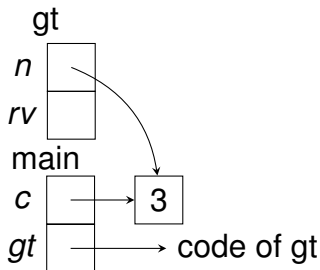
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



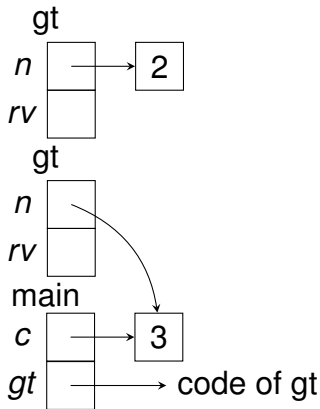
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



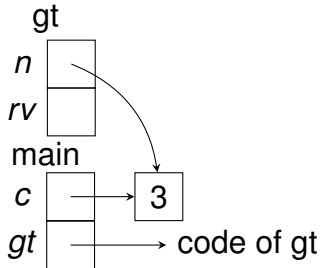
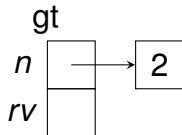
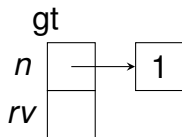
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



# Example

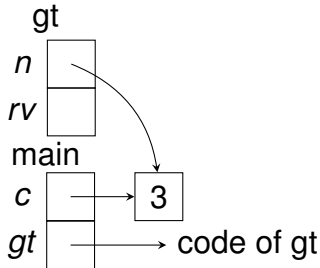
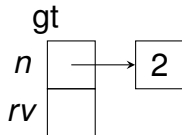
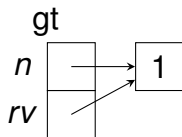
```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```





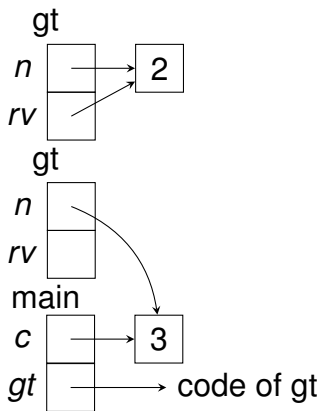
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



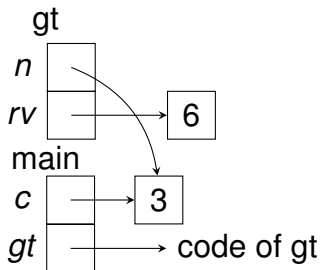
## Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



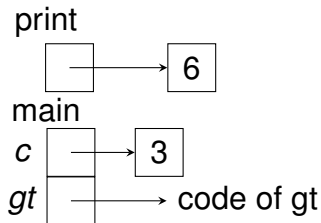
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



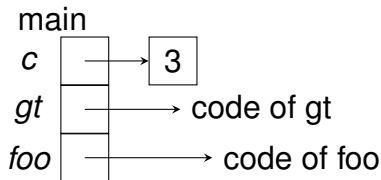
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



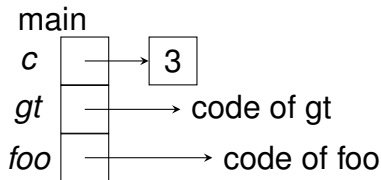
## Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



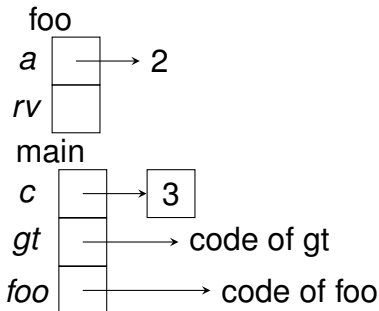
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



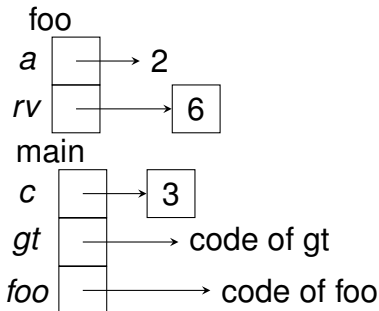
# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



# Example

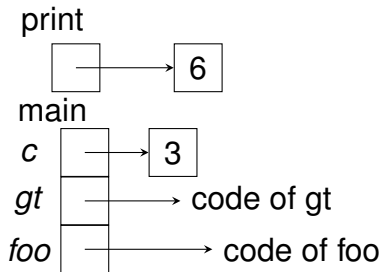
```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```





# Example

```
c = 3
def gt(n):
    if (n == 0 or n == 1):
        return 1
    else:
        return n * gt(n-1)
print(gt(c))
def foo(a):
    return a * c
print(foo(2))
```



## python tutor

- Variables: Like variables in C++
  - `[a - zA - Z_][a - zA - Z0 - 9_]*`
  - case-sensitive phân biệt hoa thường
  - keep a pointer to an object
  - declared by an assignment to the variable
- Comments:
  - Line comment: `#`
  - Block comment: `""" ... """`

- Assignments:
  - `a = 23`
  - `a,b = b,a` (swap)
  - `a = b = c = 23`
- **pass** statement: like empty statement in C++
- **if** statement: use indentation for sub-statements
  - **if** `<exp>`:  
    `<stmt-list>`  
( **elif** `<exp>`:  
    `<stmt-list>`)\*  
( **else** :  
    `<stmt-list>` )?
  - no dangling-else      else nhưng ko xác định dc của if nào

- **while**: loop when the condition is true
  - **while** <exp>:  
    <stmt-list>  
    [ **else** :  
        <stmt-list> ]
  - **else** block is executed when <exp> becomes false
- **for**: iterates over a sequence
  - **for** <var> **in** <sequence>:  
    <stmt-list>  
    [ **else** :  
        <stmt-list> ]
  - **else** block is executed after iterating all elements of <sequence>
- **break**: like C++ => terminating the loop
- **continue**: like C++ => ignoring the rest of the <stmt-list>

```
for x in range(1,11):  
    if (x == 8): break  
    print(x)  
else: print("loop_full")
```

break thoát luôn cả else  
-> ko in loop full

- User-defined function:

```
def <function name>(<parameters>):  
    <stmt–list>
```

- User-defined function:

```
def <function name>(<parameters>):  
    <stmt-list>
```

- Example

```
def giaithua(n):  
    if n == 0 or n == 1: return 1  
    else: return n * giaithua(n-1)
```

- User-defined function:

```
def <function name>(<parameters>):  
    <stmt—list>
```

- Example

```
def giaithua(n):  
    if n == 0 or n == 1: return 1  
    else: return n * giaithua(n-1)
```

- Built-in functions or libraries:

```
import <module>  
from <module> import <items>  
from <module> import *
```



- User-defined function:

```
def <function name>(<parameters>):  
    <stmt—list>
```

- Example

```
def giaithua(n):  
    if n == 0 or n == 1: return 1  
    else: return n * giaithua(n-1)
```

- Built-in functions or libraries:

```
import <module>  
from <module> import <items>  
from <module> import *
```

- Example

```
import functools  
functools.reduce(lambda x,y:x + y,[1,2,3])
```

- [1] Python Tutorial, <http://w3schools.com/python>, 10 08 2020.
- [2] Python Programming Language, <https://www.geeksforgeeks.org/python-programming-language/>, 10 08 2020.
- [3] Python Tutorial, <https://www.tutorialspoint.com/python>, 10 08 2020.
- [4] Introduction to Python 3, <https://realpython.com/python-introduction/>, 10 08 2020.