Python Functions

Computing for Data Analytics (CPSC 4800)

Mourad Bouguerra
mbouguerra@langara.ca

Langara College

June, 2022

Lesson's Outline

1. **Lesson's Learning Objectives**

2. **Python Functions**
   - Naming Conventions
   - Using Functions
     - Arbitrary Function Arguments
     - Arbitrary Function Keyword Arguments
   - Recursion

Learning Objectives

Learning Objectives

✎ Upon completion of this lesson, you will learn:

- ❒ How to use Python functions?
- ❒ What is a recursive function?
- ❒ What are the disadvantages of using recursion instead of iteration?

Functions

Functions

- Functions can be used as building blocks of Python program
- Functions are central to
  - structured programming paradigm

What is a Function?

- A function ,routine, subroutine, procedure, or method is
  - a named block statement that performs
    - a specific task

Functions

What is a Function?

- Each Python function must have a unique name
  - A valid identifier like variable name
- Python function body is a block statement
  - must be indented
- A Python function should perform a specific task
  - not many tasks
- You can pass any data type to Python function
  - data passed to a function are called arguments
- A Python function can return any data type

Naming Conventions

Identifiers

- An identifier should be descriptive and readable
- An identifier should comply with the Python style guide
  - Python Enhancement Proposal (PEP 8)[a]

[a] https://peps.python.org/pep-0008/

Naming Convention	Example
lowercase	computingfordataanalytics
UpperCamelCase	ComputingForDataAnalytics
lowerCamelCase	computingForDataAnalytics
UPPERCASE_WITH_UNDER_SCORE	COMPUTING_FOR_DATA_ANALYTICS
lowercase_with_under_score	computing_for_data_analytics

Python Enhancement Proposal (PEP 8)

Function Naming Convention

- A Python function[a]

[a] It is called a method in object-oriented terminology.

Identifier	Naming Convention	Example
Module	lowercase_with_underscores	`hello_world.py`
Variable	lowercase_with_underscores	`pep_8_url`
Constant	UPPERCASE_WITH_UNDER_SCORE	`EULER_NUMBER`
Function	lowercase_with_underscores	`display_menu()`

Class Activity

✎ Which of the following function names is compliant with PEP 8?

① NumberOfBytes()
② getNumberOfBytes()
③ get_number_of_bytes()
④ GETNUMBEROFBYTES()

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Using Functions

- A Python function that neither takes data as input nor return any data
- To call a function use its name and parenthesis

```
def display_menu():
    print('\t--------------------------------')
    print('\t\t Menu Choice')
    print('\t--------------------------------')
    print('\t 1 -- Option 1')
    print('\t 2 -- Option 2')
    print('\t 3 -- Option 3')
    print('\t 4 -- Quit')
display_menu()
```

Using Functions

- A Python function with required input, and without returning any data
- To call a function use its name and the required argument in the parenthesis

```
def display_menu_with_header(header):
    ruler = '-'*len(header)
    print(f'\t{ruler}')
    print(f'\t{header}')
    print(f'\t{ruler}')
    print('\t 1 -- Option 1')
    print('\t 2 -- Option 2')
    print('\t 3 -- Option 3')
    print('\t 4 -- Quit')
display_menu_with_header('Reading Data')
```

Using Functions

- A Python function with required input, and without returning any data
- To call a function use its name and the required argument in the parenthesis
  - as key=value pair,i.e., parameter name = argument

```
def display_menu_with_header(header):
    ruler = '-'*len(header)
    print(f'\t{ruler}')
    print(f'\t{header}')
    print(f'\t{ruler}')
    print('\t 1 -- Option 1')
    print('\t 2 -- Option 2')
    print('\t 3 -- Option 3')
    print('\t 4 -- Quit')
display_menu_with_header(header ='Reading Data')
```

Using Functions

- A Python function with optional input, and without returning any data
  - parameter with default value
- Parameters with default values should follow the parameters without default values
- To call a function use its name with or without the optional input

```
def display_menu_with_header(header='Menu Choice'):
    ruler = '-'*len(header)
    print(f'\t{ruler}')
    print(f'\t{header}')
    print(f'\t{ruler}')
    print('\t 1 -- Option 1')
    print('\t 2 -- Option 2')
    print('\t 3 -- Option 3')
    print('\t 4 -- Quit')
display_menu_with_header('Reading Data')
display_menu_with_header()
```

Functions Parameters vs Arguments

- A function parameter
  - the variable listed inside the parentheses in the function definition
- A function argument
  - the value passed to the function when it is called

```
def display_menu_with_header(header):
    pass
# header is the function parameter
display_menu_with_header('Reading Data')
# 'Reading Data' is the function argument
```

Class Activity

✎ What is the output of the following Python script?

```python
def compute_power(n, base =2):
    powers = [base**i for i in range(n)]
    return tuple(powers)
print(f'{compute_power(10)}')
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

✎ What is the output of the following Python script?

```
def compute_power(n, base =2):
    powers = [base**i for i in range(n)]
    return tuple(powers)
print(f'{compute_power(10,base=3)}')
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Using Functions

- A Python function that neither takes data as input nor return any data
- To call a function use its name and parenthesis

```
def display_menu():
    print('\t--------------------------------')
    print('\t\t Menu Choice')
    print('\t--------------------------------')
    print('\t 1 -- Option 1')
    print('\t 2 -- Option 2')
    print('\t 3 -- Option 3')
    print('\t 4 -- Quit')
def get_menu_choice():
    display_menu()
    choice = int(input('\tEnter your choice: '))
    return choice
choice = get_menu_choice()
print(f'\tYou chose option {choice}.')
```

Using Functions

- In the case, you do NOT know how many arguments will be passed to the function
  - add a * before the parameter name in the function definition
- Python naming convention for arbitrary parameters name
  - *args
- The arguments passed are stored in the args Python tuple

```
def compute_average(*args):
    sum = 0
    for x in args:
        sum += x
    return sum/len(args)
print(f'{compute_average(1,2,3)}')
print(f'{compute_average(1,2,3,4,5)}')
```

Using Functions

- In the case, you do NOT know how many keywords arguments will be passed to the function
  - add a ** before the parameter name in the function definition
- Python naming convention for arbitrary parameters name
  - **kwargs
- The arguments passed are stored in the kwargs Python dictionary

```
def get_scores(**kwargs):
    for key,value in kwargs.items():
        print(f'({key},{value})', end=' ')
get_scores(quiz=91,midterm=87)
# Output
# (quiz,91) (midterm,87)
```

Class Activity

✎ What is the output of the following Python script?

```
def get_scores(**kwargs):
    for key,value in kwargs.items():
        print(f'({key},{value})', end=' ')
get_scores(quizzes=91,assignments=97,midterm=87,final=72)
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

✎ What is the output of the following Python script?

```
def get_course_info(**kwargs):
    for key,value in kwargs.items():
        print(f'({key},{value})', end=' ')
get_course_info(title='Computing for Data Analytics',\
                code='CPSC 4800',credit=3,fees=32.40)
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Class Activity

✎ What is the output of the following Python script?

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Recursion

Recursive Function

A function that calls itself

✏ self-reference

Single Recursion

A recursion that contains a single `self-reference`

Multiple Recursion

A recursion that contains a Multiple `self-reference`s

Recursion

Recursive Function

- A problem solving method in tackling complex problems
  - ➡ divide problem into subproblems of the same type
    - a divide & conquer method
- Widely used in many important algorithms

Recursive Function

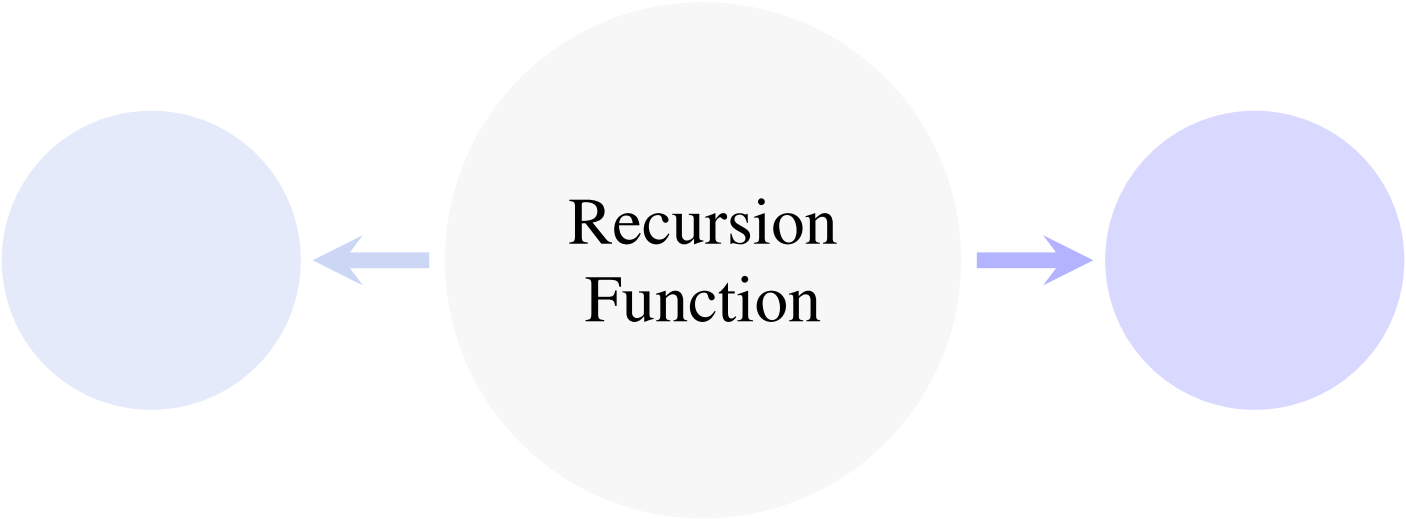
- A recursive function is a function that is
  - defined in terms of itself

$$Tr(n) = \begin{cases} 1 & \text{if } n = 1 \\ Tr(n-1) + n & \text{if } n > 1 \end{cases}$$

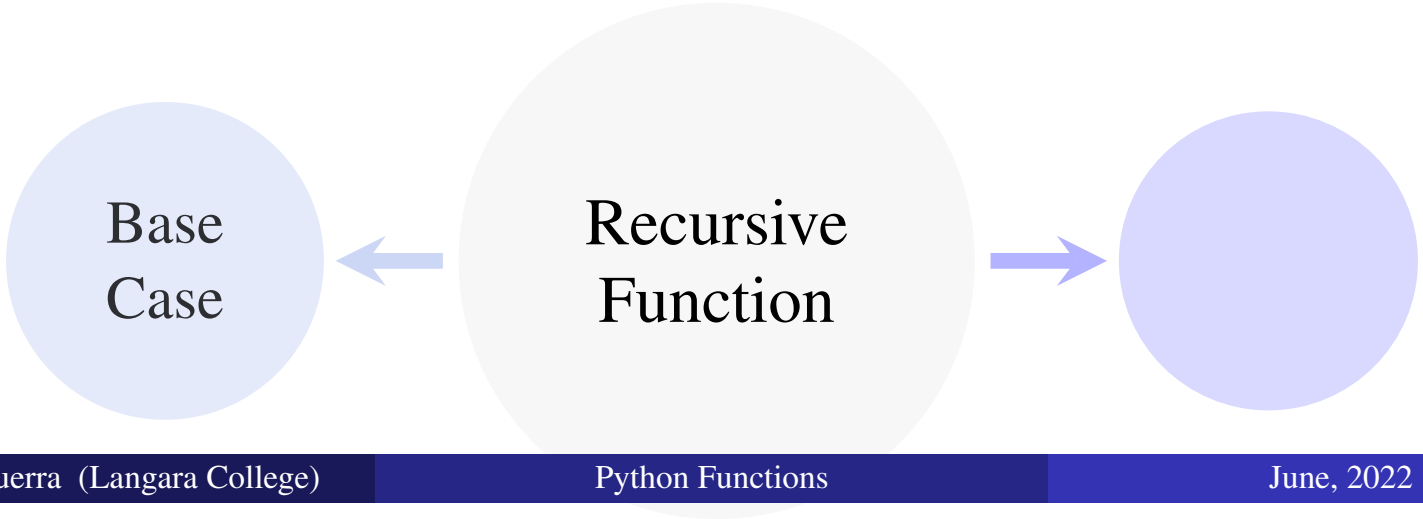
```
def triangular(n):
    if n == 1:
        return n
    return triangular(n-1)+n
```

Recursive Function

- A recursive function must include two cases:



- A recursive function must include two cases:



Recursive Function

- A recursive function must include two cases:

Base (Termination) Case

- NO recursive call

```
    if n == 1:
        return n
```

- Prevents infinite loop
- Terminates the function

Recursive Case

- Does include a recursive call
- A recursive call with a modified argument

```
    return triangular(n-1)+n
```

  - Reduces problem size $n \twoheadrightarrow n - 1$
  - Leads to the base case

Recursion vs Iteration

- Most problems solved by recursive solutions can also be solved by iteration
- Most problems solved by iteration solutions can also be solved by recursion

- Iterative Solution

```
def triangular(n):
    sum = 0
    for i in range(n):
        sum += 1
    return sum
```

- Recursive Solution

```
def triangular(n):
    if n == 1:
        return n
    return triangular(n-1)+
        n
```

Recursion vs Iteration

- Most problems solved by recursive solutions can also be solved by iteration
- Most problems solved by iteration solutions can also be solved by recursion
- Recursion is usually much slower because
  - Overhead associated with each function call
  - New variables must be created on the stack

Class Activity

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Problem

✎ *Give a recursive definition of the following functions*

① $Pow(n, p) = n^p$ *that generates the sequence* $1, n, n^2, n^3, n^4, n^6, \ldots$

② $Factorial(n) = 1 \times 2 \times 3 \times \ldots (n-1) \times n$ *that generates the sequence* $1, 2, 6, 24, 120, \ldots$