Lesson 5
Function

Kiểm tra bài cũ

- Hỏi lại về hash

- Hash trong python

- Giải thích thêm về cách lưu trữ của dic, lý do tại sao dùng dic lại rất nhanh

Why

Viết chương trình giải các bài toán sau đây:

a. Nhập một dãy n số nguyên và tính tổng dãy số vừa nhập

b. Nhập một dãy m số nguyên và tính trung bình cộng dãy số vừa nhập

Why

```
nhap m:3
nhap so:3
nhap so:5
nhap so:2
tong day vua nhap la: 10
nhap n:4
nhap so:5
nhap so:3
nhap so:7
nhap so:2
trung binh cong day vua nhap la: 4.25
```

Why

```
m = int(input('nhap m:'))
array_m = []
for v in range(m):
   number = input('nhap so:')
   array_m.append(int(number))

tong_m = 0
for v in array_m:
   tong_m += v

print('tong day vua nhap la:', tong_m)

n = int(input('nhap n:'))
array_n = []
for v in range(n):
   number = input('nhap so:')
   array_n.append(int(number))

tong_n = 0
for v in array_n:
   tong_n += v
print('trung binh cong day vua nhap la:', tong_n / len(array_n))
```

Nhận xét gì về code bên?

Nếu thay đổi yêu cầu, nhập số thực thay vì số nguyên thì sao?

Why

- Phần nhập mảng bị lặp lại nhiều lần

- Phần tính tổng đã làm ở phần a nhưng phần b vẫn phải tính lại

- Khi cần sửa để nhập số thực thì phải sửa 2 chỗ

- Code nhìn lằng nhằng, rối, khó sửa

Why

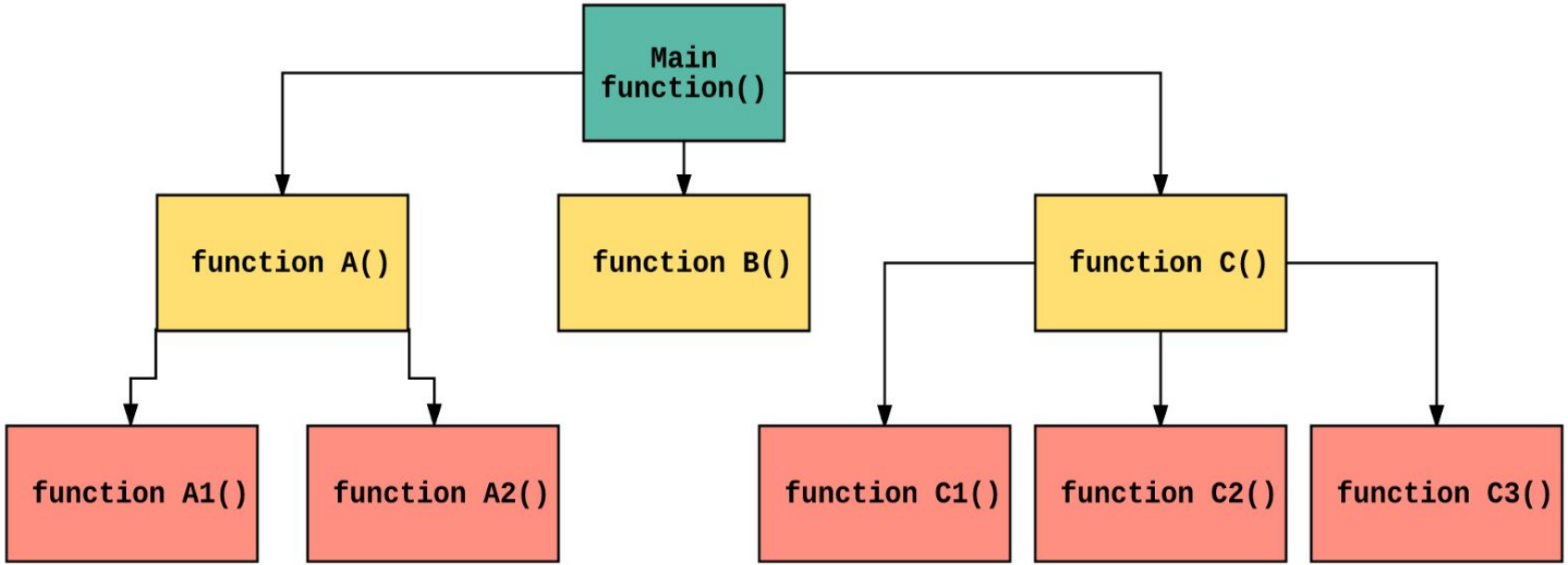
```
25
26  def input_array():
27      """
28      Hàm không tham số dùng để nhập một dãy số từ bàn phím
29      :return: Trả về một list các phần tử đã nhập
30      """
31      arr = []
32      m = int(input('nhap so phan tu:'))
33      for v in range(m):
34          number = input('nhap so:')
35          arr.append(int(number))
36      return arr
37
38  def sum_array(arr):
39      """
40      Hàm có tham số, tính tổng của danh sách các số nguyên truyền vào
41      :param arr: danh sách các số cần tính tổng
42      :return: Tổng của dãy số arr
43      """
44      tong = 0
45      for v in arr:
46          tong += v
47      return tong
48
49  array_m = input_array()
50  print('tong day so vua nhap:', sum_array(array_m))
51
52  array_n = input_array()
53  print('trung binh cong day so vua nhap:', sum_array(array_n) / len(array_n))
```

Functions

- Là một đoạn code, thực hiện một tác vụ cụ thể, thường được gọi nhiều lần từ nhiều nơi.

- Khi cần thực hiện những tác vụ đó thì chỉ cần "call" function và nhận kết quả trả về nếu có khi nó thực hiện xong

Functions


Main function()
function A()
function B()
function C()
function A1()
function A2()
function C1()
function C2()
function C3()

Function trong thực tế

- Sự phát triển của lập trình
  - Lập trình phi cấu trúc
  - Lập trình cấu trúc
  - Lập trình Thủ tục & Chức năng
  - Lập trình hướng đối tượng
  - lập trình hướng khía cạnh
- Đa số các ngôn ngữ đề có khái niệm hàm
- Hàm làm nên sự phát triển mạnh mẽ của lập trình

Functions - Cú pháp

```
def function_name([parameters]):
    """
    function document
    :param parameters:
    :return:
    """
    instruction
    [return expression]
```

- Bắt đầu bằng từ khóa def
- Function_name: Dễ hiểu, dễ nhớ, nói lên được chức năng của hàm.
- parameters: Tham số của hàm
- Nên ghi các mô tả về chức năng, input, output cho hàm để dễ đọc, dễ hiểu
- các lệnh nằm trong function được tính từ sau dấu : đến hết khối.
- Một hàm thì có thể return hoặc không. Nếu không ghi gì thì tương đương với return None

Function không tham số

```
def say_hi():

    print('hi')
```

=> Chạy thử !

=> Tại sao?

Function không tham số

Với function cần có hai bước:

1. Khai báo hàm

2. Gọi hàm

- Các lệnh trong hàm chỉ được thực hiện khi hàm được gọi.

Function không tham số

```
def say_hi():

    print('hi')



say_hi()

say_hi()

say_hi()

say_hi()

say_hi()
```

Function không tham số

```
def say_hi():
    print('hi')
    print('bye')
say_hi()
say_hi()
say_hi()
say_hi()
say_hi()
```

- Thêm print('bye')

- Chạy lại

- Nhận xét

Function có tham số - Không return

Xét hàm cộng hai số như sau:

```
def add_two_number():
    a = int(input('nhap so thu nhat:'))
    b = int(input('nhap so thu hai:'))
    print('tong hai so la:', a + b)
```

=> Chạy được!

=> Nhận xét gì?

Function có tham số - Không return

- Chỉ chạy được khi nhập trực tiếp hai số. Nếu giả sử đã có 2 số a,b rồi thì vẫn không dùng được hàm này

- Chỉ chạy được trong môi trường console, Hàm này không thể dùng trong môi trường web hoặc UI

=> Chạy được nhưng không dùng lại được nhiều. Hàm này viết ra không mang lại hiệu quả nhiều.

Mục đích chính của hàm là để reuse

Function có tham số - Không return

```
def add_two_number(a, b):

    print('tong hai so la:', a + b)



num1 = int(input('nhap so thu nhat:'))

num2 = int(input('nhap so thu hai:'))

add_two_number(num1, num2)
```

Function có tham số - Không return

- Tham số đầu vào là cách để hàm có thể nhận được các giá trị khác nhau để xử lý trong khi ở thời điểm viết hàm không cần biết chính xác các giá trị này là bao nhiêu.

- Trong thân hàm, các tham số được dùng như các biến

- Sau khi thoát khỏi hàm, các tham số không còn nữa

- Có thể truyền biến, hằng, biểu thức vào hàm. Các lời gọi sau add_two_number(x,y), add_two_number(3*4,6+3), add_two_number(1,2) đều hợp lệ.

Function có tham số - Có return

```
def add_two_number(a, b):

    print('tong hai so la:', a + b)
```

=> Hàm này chỉ tính tổng được cho 2 số, Cần 3 số thì sao???

Function có tham số - Có return

```
def add_two_number(a, b):

   return a + b

num1 = int(input('nhap so thu nhat:'))

num2 = int(input('nhap so thu hai:'))

num3 = int(input('nhap so thu ba:'))

sum_1_2 = add_two_number(num1, num2)

sum_3 = add_two_number(sum_1_2, num3)

print('tong ba so la:', sum_3)
```

Function có tham số - Có return

- Lệnh return phải nằm trong body của hàm

- Lệnh return sẽ trả về kết quả cho lời gọi hàm: ví dụ tong=add_two_number(1,2)

- Trong hàm có thể không có, có một hoặc nhiều lệnh return

- Khi gặp lệnh return thì các câu lệnh sau đó không được hiện nữa

Function có tham số - Có return

```
1   def abs_of_number(a):
2       if a > 0:
3           return a
4           print('tri tuyet doi la:', a)
5       else:
6           return -a
7           print('tri tuyet doi la:', -a)
8       print('tri tuyet doi la:', a)
9
10
11  x = abs_of_number(-12)
12  tong = 12 + abs_of_number(-12)
```

Ex

Viết hàm tên là evaluate, nhận vào 2 số và 1 phép tính. Tính toán phép tính này với hai số đầu vào và trả về giá trị tính được.

ví dụ:

x = evaluate(1,3,'+')

x sẽ có giá trị là 4

Ex

- Một chương trình lớn gồm nhiều hàm, nằm ở nhiều file

- Khi cần dùng hàm ở một file khác ta phải import

- Import là báo cho python biết ta sẽ dùng hàm nào ở file nào

Ví dụ: Tạo một file khác và dùng lại hàm tính tổng đã làm.

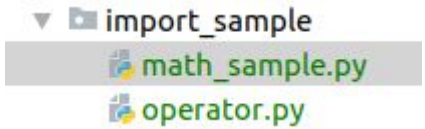
Ex

operator.py

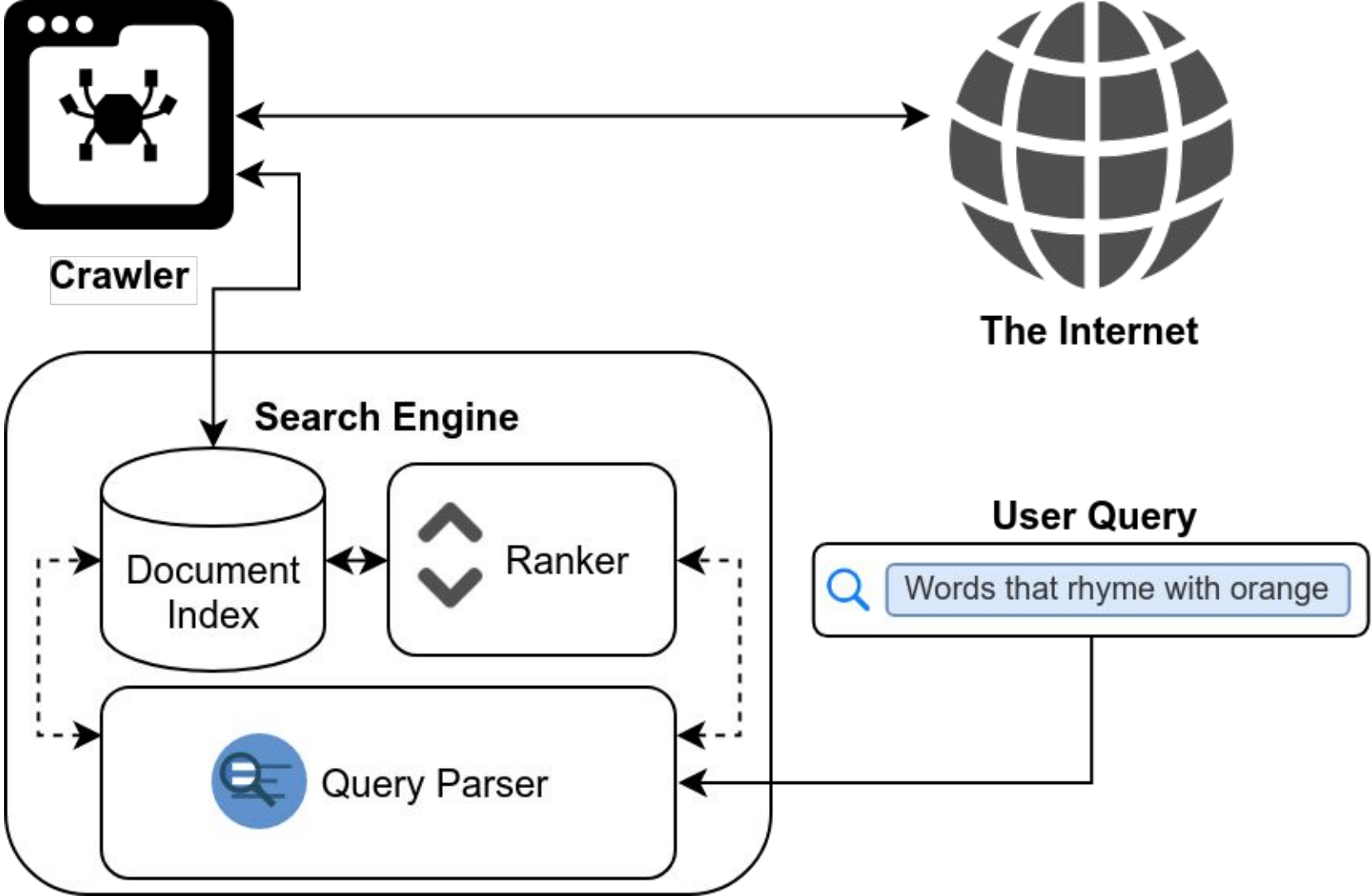
```
1
2   def add(a, b):
3       return a + b
4
5
6   def sub(a, b):
7       return a - b
8
9
10  def mul(a, b):
11      return a * b
12
```

math_sample.py

```
1   from operator import sub, mul
2
3
4   def cal_delta(a, b, c):
5       return sub(mul(b, b), mul(4, mul(a, c)))
6
7
8   print(cal_delta(1, 2, -3))
9
```



Next lesson


Crawler
The Internet
Search Engine
Document Index
Ranker
User Query
Words that rhyme with orange
Query Parser