

Function dan Procedure

Bagian 2



Kategori Function

1

Standard Library Function

fungsi-fungsi yang telah disediakan oleh Interpreter Python dalam file-file atau librarynya.

2

Programme-Defined Function

function yang dibuat oleh programmer sendiri. Function ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library buatan programmer itu sendiri

Variabel-Length Arguments

Argumen yang memungkinkan fungsi menerima jumlah argumen yang tidak terbatas.

python

```
def sum_all(*args):  
    total = 0  
    for num in args:  
        total += num  
    return total  
  
print(sum_all(1, 2, 3)) # Output: 6
```

python

```
def print_info(**kwargs):  
    for key, value in kwargs.items():  
        print(f"{key}: {value}")  
  
print_info(name="Alice", age=30, city="New York")  
# Output:  
# name: Alice  
# age: 30  
# city: New York
```


Statement Lamda

- Selain statement def, Python juga menyediakan suatu bentuk ekspresi yang menghasilkan objek fungsi. Karena kesamaanya dengan tools dalam bahasa Lisp, ini disebut Lambda.
- Seperti def, ekspresi ini menciptakan sebuah fungsi yang akan dipanggil nanti, tapi mengembalikan fungsi dan bukan untuk menetapkan nama.
- Inilah sebabnya mengapa kadang-kadang lambda dikenal sebagai anonym (yakni, tidak disebutkan namanya) fungsi.

Statement Lamda

- Dalam prakteknya, mereka sering digunakan sebagai cara untuk inline definisi fungsi, atau untuk menunda pelaksanaan sepotong kode.
- Bentuk umum lambda adalah kata kunci lambda, diikuti oleh satu atau lebih argument (persis, seperti daftar argument dalam tanda kurung di def header), diikuti oleh ekspresi setelah tanda titik dua:

python

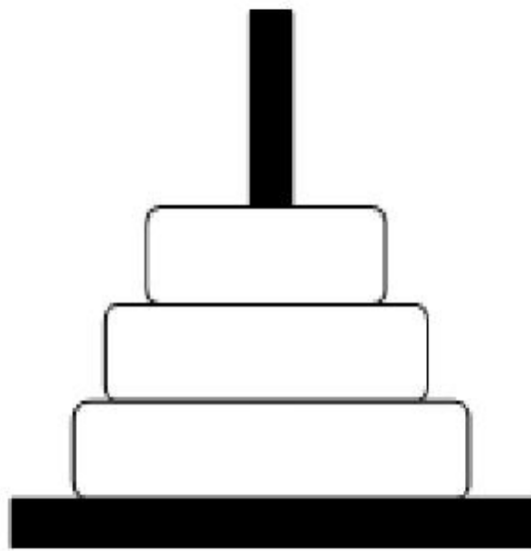
 Copy code

```
lambda arguments: expression
```

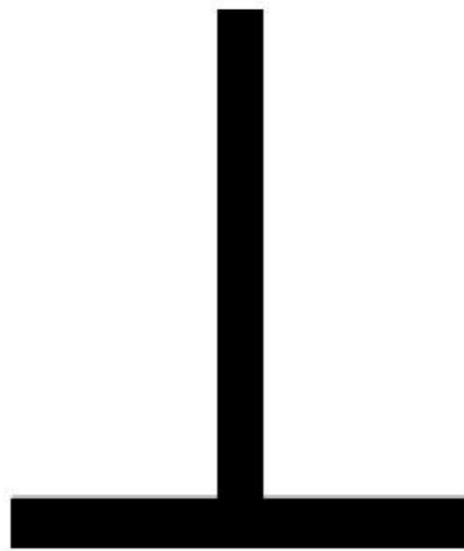
Lambda vs Def

Lambda memiliki perbedaan dengan def antara lain :

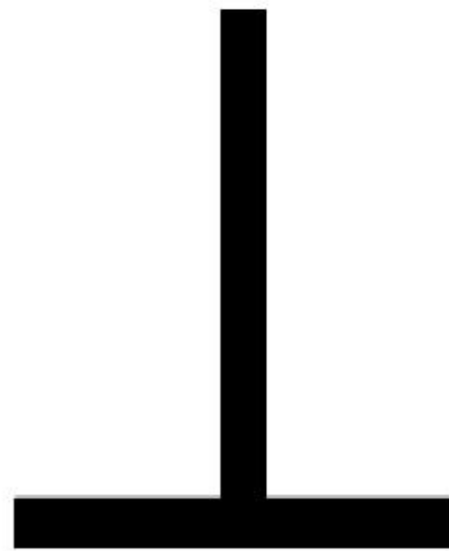
1. Lambda adalah sebuah ekspresi, bukan pernyataan.
 - Karena ini, sebuah lambda dapat muncul di tempat-tempat def tidak diperbolehkan oleh sintaks Python—di dalam daftar harfiah atau pemanggilan fungsi argument, misalnya.
 - Sebagai ekspresi, lambda mengembalikan nilai (fungsi baru) yang opsional dapat diberi nama.
 - Sebaliknya, pernyataan def selalu memberikan fungsi baru ke nama di header, bukannya kembali sebagai hasilnya.
2. Tubuh lambda adalah ekspresi tunggal, bukan satu blok statemen.
 - Tubuh lambda sama dengan apa yang akan dimasukkan ke dalam statement return dalam tubuh def



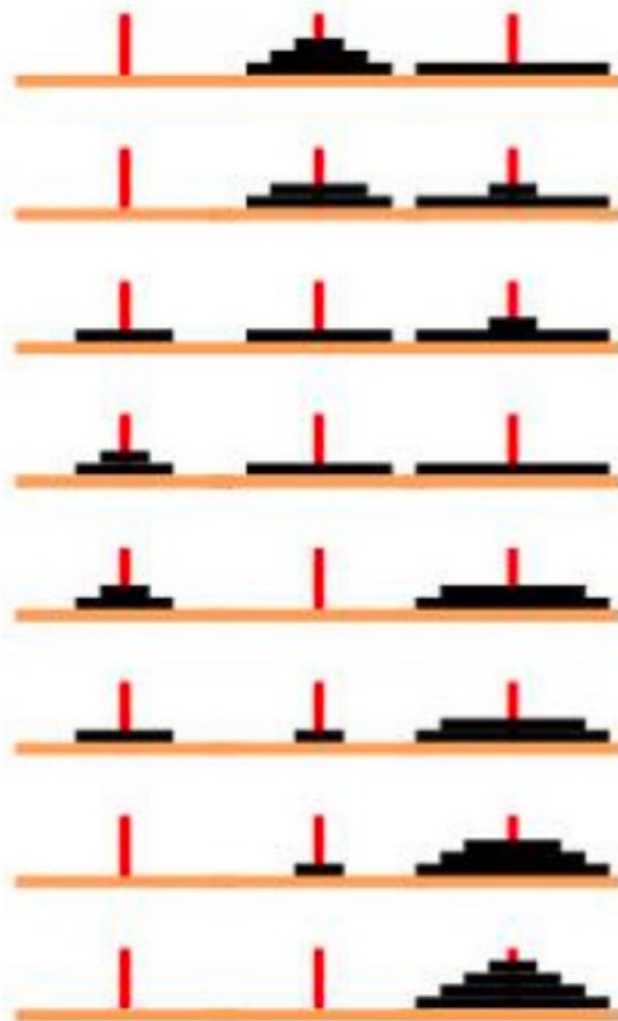
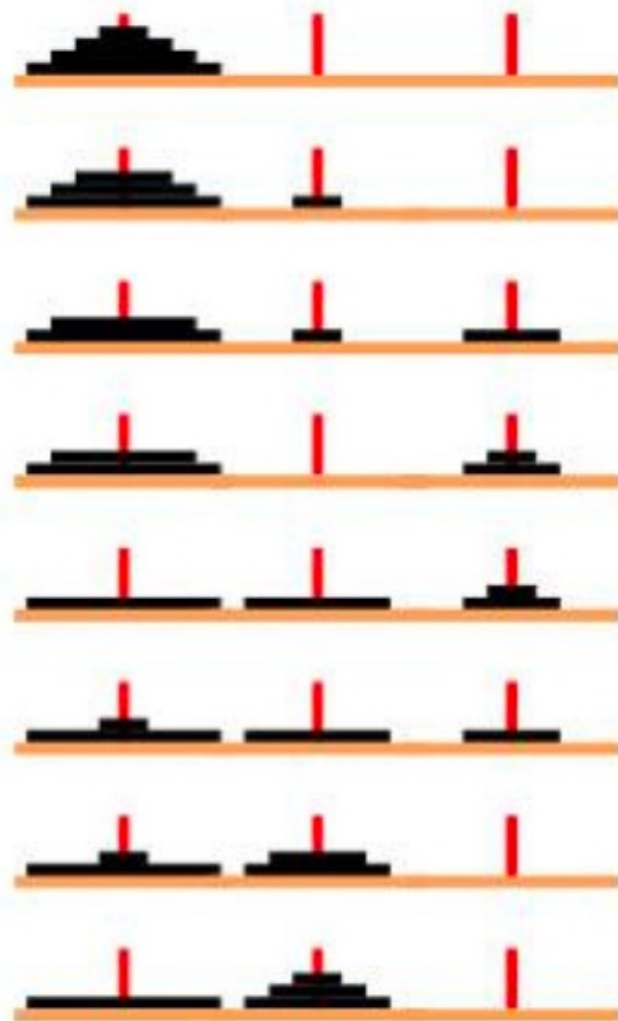
A



B



C



Fungsi Rekursif

- Fungsi Rekursif merupakan suatu fungsi yang memanggil dirinya sendiri.
- Artinya, fungsi tersebut dipanggil di dalam tubuh fungsi itu sendiri.
- Tujuan di lakukan rekursif adalah untuk menyederhanakan penulisan program dan menggantikan bentuk iterasi.
- Dengan rekursi, program akan lebih mudah.

Fungsi Rekursif

Contoh klasik penggunaan fungsi rekursif adalah dalam menghitung faktorial sebuah bilangan. Faktorial dari bilangan bulat positif n (ditulis sebagai $n!$) adalah hasil perkalian semua bilangan bulat positif dari 1 hingga n . Secara matematis, faktorial dinyatakan sebagai berikut:

$$n! = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$$

Fungsi Rekursif

python


 Copy code

```
def recursive_function(parameters):  
    if base_case_condition:  
        return base_case_value  
    else:  
        return recursive_function(modified_parameters)
```

Dalam fungsi rekursif, perhatikan bahwa base case sangat penting untuk menghindari infinite recursion. Setiap panggilan rekursif harus mendekati base case untuk memastikan bahwa rekursi berhenti pada suatu titik.

Fungsi Rekursif

python


 Copy code

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

Contoh penggunaan

```
print(factorial(5)) # Output: 120
```

python

 Copy code

```
def factorial(n):  
    result = 1  
    for i in range(1, n + 1):  
        result *= i  
    return result
```

Contoh penggunaan

```
print(factorial(5)) # Output: 120
```

Perbedaan Rekursif dan Iterasi

RECURSION	ITERATIONS
Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri	Iterasi adalah pengulangan dari suatu proses menggunakan struktur loop
Rekursi menggunakan struktur pemilihan	iterasi menggunakan struktur perulangan
Rekursi tak hingga terjadi apabila langkah rekursi tidak pernah memenuhi kondisi pada base case	Perulangan tak hingga terjadi apabila kondisi perulangan tidak pernah bernilai false
Rekursi berhenti apabila kondisi (base case) dikenali	Iterasi berhenti ketika kondisi perulangan bernilai false
Rekursi lebih banyak menggunakan memory daripada iterasi	Iterasi lebih hemat memory
rekursi tak hingga dapat menyebabkan terjadinya crash pada sistem	Perulangan tak hingga akan mengakibatkan penggunaan Siklus CPU secara berulang
Rekursi membuat kode menjadi lebih sederhana	Iterasi membuat kode menjadi lebih panjang

TERIMA KASIH 