The Functools Module

Oleh : Agus Priyanto, M.Kom

INSTITUT TEKNOLOGI TELKOM PURWOKERTO
Smart, Trustworthy, And Teamwork

Outline Materi

- Deret Fibonacci
- Functool Module
  - @lru_cache ()
  - partial()
  - reduce()

Deret Fibonacci

Deret Fibonacci :

Deret Fibonacci adalah suatu deret matematika yang berasal dari penjumlahan dua bilangan sebelumnya.

1, 1, 2, 3, 5, 8, 13, 21…

Rumus : f(n) = f(n-1) + f(n-2)
f(6) = f(6-1) + f(6-2)
= 5 + 3 = 8

Persamaan F(n) dapat dijelaskan sebagai berikut :

- Jika n=0, maka F(0)=0
- Jika n=1, maka F(1)=1
- Jika n>1 berlaku rumus F(n-1) + F(n-2)
- Jika n=2, maka F(2-1) + F(2-2) = F(1) + F(0) = 1 + 0 = 1
- Jika n=3 maka F(3-1) + F(3-2) = F(2) + F(1) = 1 + 1 = 2
- Jika n=4 maka F(4-1) + F(4-2) = F(3) + F(2) = 2 + 1 = 3

Hasil deret bilangan Fibonacci adalah : **0,1,1,2,3, dst**

Testing 1

```
>>> def fibonacci(n):
   if n == 1 :
        return 1
   elif n == 2:
        return 1
   elif n > 2 :
        return fibonacci(n-1)+ fibonacci(n-2)
>>> for n in range (1,11):
        print (n, ":", fibonacci (n))
1 : 1
2 : 1
3 : 2
4 : 3
5 : 5
6 : 8
7 : 13
8 : 21
9 : 34
10 : 55
>>>
```

Perhatikan kecepatan running code program !!!

Testing 2

```
>>> for n in range (1,101):
        print (n, ":", fibonacci (n))
1 : 1
2 : 1
3 : 2
4 : 3
5 : 5
6 : 8
7 : 13
8 : 21
9 : 34
10 : 55
...
33 : 3524578
...
...
>>>
```

Perhatikan kecepatan running code program !!!

Testing 3

```
>>> for n in range (1,501):
        print (n, ":", fibonacci (n))
1 : 1
2 : 1
3 : 2
4 : 3
5 : 5
6 : 8
7 : 13
8 : 21
9 : 34
10 : 55
...
33 : 3524578
...
...
>>>
```

Memoization

- ***Memoization*** adalah teknik yang digunakan dalam komputasi untuk mempercepat program

- Hal ini dilakukan dengan menyimpan hasil perhitungan input yang diproses seperti hasil pemanggilan fungsi

1. Cache Value

```
>>> fibonacci_cache = {}
>>> def fibonacci(n):
        if n in fibonacci_cache :
                return fibonacci_cache[n]
        if n == 1 :
                value = 1
        elif n == 2:
                value = 1
        elif n > 2 :
                value = fibonacci(n-1)+ fibonacci(n-2)
        fibonacci_cache[n] = value
   return value

>>> for n in range (1, 101)
        print (n, ":", fibonacci(n))
>>>
```

2. LRU Cache = Least Recently Used

```
>>> from functools import lru_cache

>>> @lru_cache(maxsize = 1000)
def fibonacci(n):
        if n == 1 :
                return 1
        elif n == 2:
                return 1
        elif n > 2 :
                return fibonacci(n-1)+ fibonacci(n-2)

>>> for n in range (1, 501):
        print (n, ":", fibonacci(n))
>>>
```

```
>>> from functools import lru_cache

>>> @lru_cache(maxsize = 1000)
def fibonacci(n):
   if type (n) != int :
        raise TypeError ("n harus bilangan integer positif)
   if n < 1 :
        raise ValueError ("n harus bilangan integer positif)
   if n == 1 :
        return 1
   elif n == 2:
        return 1
   elif n > 2 :
        return fibonacci(n-1)+ fibonacci(n-2)
>>>
```

Testing 1

```
>>> for n in range (1, 51):
      print (fibonacci(n))
>>>
```

Testing 2

```
>>> for n in range (1, 101):
      print (fibonacci(n+1)/fibonacci (n))
>>>
```

Fungsi `partial()`

```
>>> from functools import partial
>>> def func_multiply (x,y):
        return x * y

>>> func_multiply (3,4)
12
>>> # membuat fungsi partial
>>> partial_multiply = partial (func_multiply, 4)
>>> partial_multiply (4)
16
>>> partial_multiply (3)
12
>>>
```

- Fungsi **partial()** memungkinkan untuk menurunkan fungsi dengan parameter x ke fungsi dengan lebih sedikit parameter dan nilai tetap yang ditetapkan untuk fungsi yang lebih terbatas.

Fungsi `reduce()`

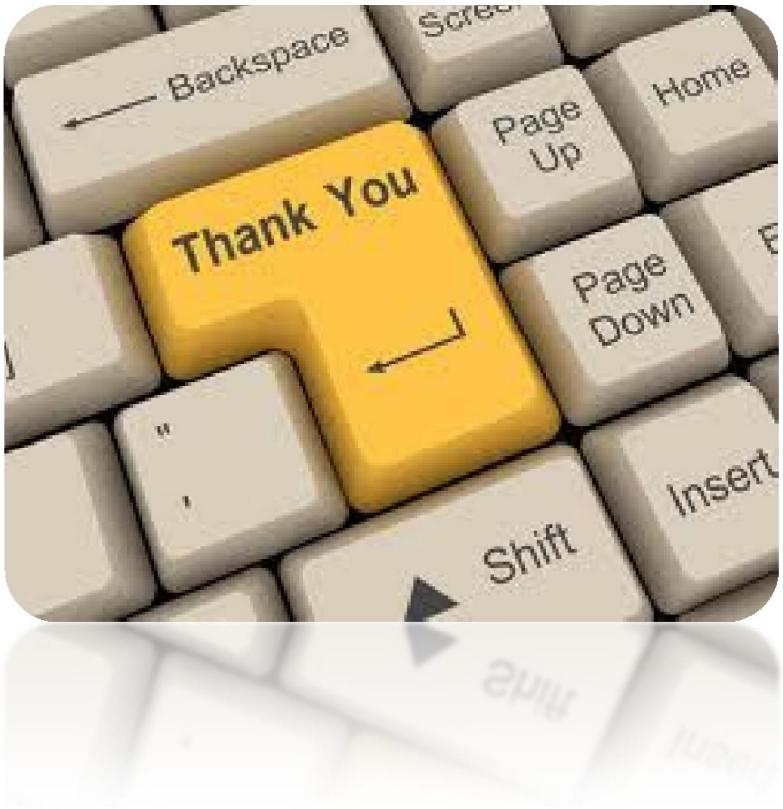
```
>>> form functools import reduce

>>> data = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
>>> multiplier = lambda x, y : x*y
>>> reduce (multiplier, data)
6469693230

>>> product = 1
>>> for x in data :
        product = product * x
>>> product
6469693230
>>>
```

```
>>> form functools import reduce
>>> numbers = [99, 47, 11, 6, 42, 102, 13, 9]
>>> f = lambda a,b : a if a > b else b
>>> max_value = reduce (f, numbers)
>>> print (max_value)
102
>>>
```

reduce function, reduce(function, iterable) menerapkan dua argumen secara kumulatif ke item iterable, dari kiri ke kanan.


Backspace
Thank You
Page Up
Home
Page Down
Insert
Shift