Recursion

Raja Oktovin

Case Study

- Factorial
- Fibonacci
- Exercise 1: Misteri, Rahasia, Exponent
- Stack Trace and Error
- Palindrome
- Exponent
- Exercise 2: Recursive Binary

Factorial

Definisi. Faktorial dari bilangan asli n kita notasikan $n!$ dan untuk $n \geq 1$ didefinisikan sebagai hasil kali dari 1 hingga n**. Untuk $\boldsymbol{n = 0}$, didefinisikan $\boldsymbol{0! = 1}$.**

Contoh: $\boldsymbol{4! = 4 \times 3 \times 2 \times 1 = 24}$

$$\boldsymbol{5! = 5 \times 4! = 5 \times (24) = 120}$$

- Faktorial adalah salah satu **fungsi** yang bersifat rekursif.
- Apa maksudnya rekursif? Segala sesuatu yang di dalam definisinya ada dirinya sendiri (tapi mungkin dalam bentuk yang lebih sederhana).

Contoh: Definisi factorial berikut bersifat non-rekursif

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

Definisi rekursif dari factorial adalah sebagai berikut:

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

- Perhatikan bahwa perhitungan dengan definisi "tidak lengkap"

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

Akan mengakibatkan infinite computation (that never ends)

$$3! = 3 \times 2!$$
$$2! = 2 \times 1!$$
$$1! = 1 \times 0!$$
$$0! = 0 \times (-1)!$$
$$\text{dst}$$

- Definisi rekursif untuk factorial yang lengkap adalah

Jika $n = 0$, maka $n! = 1$

dan

Jika $n > 0$, maka $n! = n \times (n - 1)!$

Jika $n = 0$, maka $n! = 1$ #base case

dan

Jika $n > 0$, maka $n! = n \times (n-1)!$ #recursive case

Fibonacci

- $F_0 = 1$
- $F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$ untuk setiap $n > 1$

```
def fib(n):
    if n==0 or n==1:
        return 1
    return fib(n-1)+fib(n-2)

def fib_mem(n, mem):
    if n in mem: #to avoid redundant calls
        return mem[n]
    mem[n] = fib_mem(n-1, mem) + fib_mem(n-2, mem)
    return mem[n]

mem = {0: 1, 1:1}
print(fib_mem(100, mem)) #very fast; using memoized computation
print(fib(100)) #takes very long time
```

Stack conditions for fib(6)

See the board.

PR

```
def misteri(m,n): #asumsikan m,n >= 0
    if m==0:
        return 0
    return n + misteri(m-1, n)

def rahasia(m,n): #asumsikan m,n >= 0
    if m < n:
        return 0
    return 1 + rahasia(m-n, n)

#Apa yang akan dilakukan oleh fungsi tersebut?
#Jelaskan cara kerjanya kenapa bisa hasilnya seperti itu!

print(misteri(10,10))
print(rahasia(25,7))
```

Stacktrace

- Jika program kita mengalami runtime error, program akan memberi tahu baris berapa terjadi eror.
- Jika eror terjadi pada baris di dalam suatu fungsi, program akan mencetak stacktrace berisi di bagian mana dari program fungsi tersebut dipanggil saat terjadi eror hingga pada level program utama (disebut <module>).
- Ini sangat berguna terutama jika suatu fungsi dipanggil di beberapa tempat; mungkin kesalahan terjadi di pemanggilan pertama, kedua, ketiga, dst. mungkin hingga terakhir.

Program contains error. What is the message?

```
1  def fun1():
2      fun2()
3
4  def fun2():
5      fun3()
6
7  def fun3():
8      print(1/0)
9
10 fun1()
11 fun2()
12 fun3()
```

Program contains error. What is the message?

```
1  def fun1():
2      fun2()
3
4  def fun2():
5      fun3()
6
7  def fun3():
8      print(1/0)
9
10 fun1()
11 fun2()
12 fun3()
```

Studi Kasus

Studi Kasus: Memeriksa Palindrom

```
def isPalindrome(kata):
    print(f"isPalindrome({kata}) dipanggil")
    if len(kata) == 0:
        return True
    if kata[0] == kata[-1]:
        return isPalindrome(kata[1:-1])
    else:
        return False

print(isPalindrome("KATAK"))
print(isPalindrome("PENAPPNEP"))
print(isPalindrome("PENAPPANEP"))
```

Studi Kasus: Count Recursive

```
def hitung(lst, x):
    print(lst, x)
    if len(lst) == 0:
        return 0
    else:
        if (x == lst[0]):
            return 1 + hitung(lst[1:], x)
        else:
            return hitung(lst[1:], x)

print(hitung([1,3,2,3,4], 3))
```

PR

Buat fungsi rekursif **terkecil(lst)** yang mengembalikan elemen terkecil dari lst. Diasumsikan lst berisi integer.

terkecil([1,2,3,5,-1]) mengembalikan -1