PERTEMUAN 6

FUNGSI REKURSIF

A. TUJUAN PEMBELAJARAN

Setelah mempelajari Materi ini diharapkan mahasiswa mengerti mengenai fungsi, dan mampu menggunakan dan membuat program dengan fungsi rekursif dengan Python.

B. URAIAN MATERI

1. Definisi Fungsi

Melakuakan suatu proses yang memiliki kaitan adalah merpaka tujuan fungsi dimana adalah suatu blok kode yang terstruktur dan dapat memanggil kembali apabila diperlukan. Dalam Python terdapat fungsi bawaan seperti print() dan banyak lagi lainnya, akan tetapi bisa juga dengan fungsi yang dibuat sendiri oleh programmer

Python Functions suatu blok pernyataan terkait yang dirancang untuk melakukan tugas komputasi, logis, atau evaluatif. Idenya adalah untuk menempatkan beberapa tugas yang umum atau berulang kali dilakukan bersama-sama dan membuat fungsi sehingga alih-alih menulis kode yang sama lagi dan lagi untuk input yang berbeda, dimana dapat melakukan panggilan fungsi untuk menggunakan kembali kode yang terkandung di dalamnya berulang kali. Fungsi dapat built-in atau user-defined. Ini membantu program untuk menjadi ringkas, tidak berulang, dan terorganisir. Memanggil Fungsi dimana setelah membuat fungsi kita dapat menyebutnya dengan menggunakan nama fungsi diikuti oleh kurung yang mengandung parameter dari fungsi tertentu.

a. Argumen dari Sebuah Fungsi

Argumen adalah nilai-nilai yang dilewatkan di dalam kurung fungsi. Sebuah fungsi dapat memiliki sejumlah argumen yang dipisahkan oleh koma.

b. Jenis Argumen

Python mendukung berbagai jenis argumen yang dapat diteruskan pada saat panggilan fungsi. Mari kita bahas setiap jenis secara detail.

c. Argumen default

Argumen default ialah parameter dimana mengasumsikan nilai default apabila nilai tidak disediakan dalam pemanggilan fungsi untuk argumen tersebut. Contoh berikut ini menggambarkan argumen default. Seperti argumen default C ++, sejumlah argumen dalam suatu fungsi dapat memiliki nilai default. Tetapi begitu kita memiliki argumen default, semua argumen di sebelah kanannya juga harus memiliki nilai default.

d. Argumen kata kunci

Idenya adalah untuk memungkinkan penelepon untuk menentukan nama argumen dengan nilai-nilai sehingga penelepon tidak perlu mengingat urutan parameter.

e. Argumen panjang variabel

Dalam Python, kita dapat meneruskan sejumlah argumen variabel ke fungsi menggunakan simbol khusus. Ada dua simbol khusus:

args (Argumen Non-Kata Kunci)

**kwargs (Argumen Kata Kunci)

f. Docstring

String pertama setelah fungsi disebut string Dokumen atau Docstring secara singkat. Ini digunakan untuk menggambarkan fungsi fungsi. Penggunaan docstring dalam fungsi adalah opsional tetapi dianggap sebagai praktik yang baik. Sintaks di bawah ini dapat digunakan untuk mencetak docstring fungsi:

Syntax: print(function_name.__doc__)

g. Pernyataan kembali

Pernyataan pengembalian fungsi digunakan untuk keluar dari fungsi dan kembali ke fungsi mengembalikan nilai atau item data yang ditentukan ke penelepon.

h. Syntax: return [expression_list]

Pernyataan pengembalian dapat terdiri dari variabel, ekspresi, atau konstanta yang dikembalikan ke akhir eksekusi fungsi. Jika tidak ada yang di atas hadir dengan pernyataan kembali, objek None dikembalikan.

Apakah Fungsi Python Melewati Referensi atau melewati nilai? Di dalam Python setiap nama variabel adalah referensi. Ketika kita meneruskan variabel ke fungsi, referensi baru ke objek dibuat. Parameter yang lewat di Python sama dengan referensi yang lewat di Java. Dalam Python, fungsi anonim berarti bahwa fungsi tanpa nama. Seperti yang sudah kita ketahui, kata kunci def digunakan untuk menentukan fungsi normal dan kata kunci lambda digunakan untuk membuat fungsi anonim. Silakan lihat ini untuk detailnya.

2. Memanggil Function/Fungsi

Mendefinisikan fungsi hanya memberinya nama, penentuan parameter yang harus dimasukkan dalam fungsi dan struktur blok kode. Setelah struktur dasar fungsi diselesaikan, dapat menjalankannya dengan menyebutnya dari fungsi lain atau langsung dari prompt Python. Berikut ini adalah contoh untuk memanggil fungsi printme()

```
1 def printme( str ):
2    "This prints a passed string into this function"
3    print str
4    return;
5
6 # sekarang memanggil fungsi printme
7 printme("Ini panggilan pertama untuk fungsi user defined !")
8 printme("mamanggil kembali untuk fungsi yang sama")
9
```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Ini panggilan pertama untuk fungsi user defined !

mamanggil kembali untuk fungsi yang sama

3. Melewati Referensi Vs Nilai

Seluruh parameter (argumen) Python dilewatkan dengan referensi. Ini berarti apabila mengubah apa parameter mengacu dalam fungsi, perubahan juga mencerminkan kembali dalam fungsi panggilan. Misalnya sebagai berikut.

```
def changeme( listku ):
 "Mengubah list  yang terdapat  dalam fungsi ini"
 listku.append([5,6,7,8]);
 print "Nilai di dalam fungsi: ", listku
 return

# sekarang memanggil fungsi changeme
listku = [50,60,70];
changeme( listku );
print "Nilai di luar fungsi: ", listku

```

Menjelaskan bahwa mempertahankan referensi objek yang dilewati dan menambahkan nilai dalam objek yang sama. Jadi hasilnya berikut :

Nilai di dalam fungsi: [50, 60, 70, [5, 6, 7, 8]]

Nilai di luar fungsi: [50, 60, 70, [5, 6, 7, 8]]

Ada satu contoh di mana argumen sedang dilewatkan oleh referensi dan referensi sedang ditimpa di dalam fungsi yang disebut.

```
def changeme( listku ):
   "This changes a passed list into this function"
   listku = [5,6,7,8]; # This would assig new reference in mylist
   print "Nilai di dalam fungsi: ", listku
   return

# sekarang dapat memanggil fungsi changeme
listku = [50,60,70];
changeme( listku );
print "Nilai diluar fungsi: ", listku

```

Parameter *listku* bersifat lokal untuk fungsi changeme. Mengubah listku dalam fungsi tidak mempengaruhi *listku.* Fungsi ini tidak menyelesaikan apa-apa dan akhirnya ini akan menghasilkan hasil berikut :

Nilai di dalam fungsi: [5, 6, 7, 8]

Nilai diluar fungsi: [50, 60, 70]

a. Argumen Fungsi

Programmer dapat memanggil fungsi dengan menggunakan jenis argumen formal berikut.

1) Argumen yang diperlukan
2) Argumen kata kunci
3) Argumen default
4) Argumen panjang variabel

b. Argumen yang diperlukan

Argumen yang diperlukan adalah argumen yang diteruskan ke fungsi dalam urutan posisi yang benar. Di sini, jumlah argumen dalam panggilan fungsi harus sesuai persis dengan definisi fungsi. Untuk memanggil fungsi *printme(),*Anda pasti perlu melewati satu argumen, jika tidak, itu memberikan kesalahan sintaks sebagai berikut

```
1 def printme( str ):
2    "This prints a passed string into this function"
3    print str
4    return;
5
6 # Now you can call printme function
7 printme()
8
```

c. Argumen kata kunci

Argumen kata kunci terkait dengan panggilan fungsi. Ketika Anda menggunakan argumen kata kunci dalam panggilan fungsi, penelepon mengidentifikasi argumen dengan nama parameter. Hal ini memungkinkan untuk melewatkan argumen atau menempatkan mereka keluar dari urutan karena interpreter Python dapat menggunakan kata kunci yang disediakan untuk mencocokkan nilai dengan parameter. Programmer juga dapat membuat panggilan kata kunci ke fungsi *printme()* dengan cara berikut :

```
def printme( str ):
   "Mencetak  sebuah posisi string ke dalam fungsi"
   print str
   return;

# Sekarang dapat memanggil fungsi printme
printme( str = " String Ku")

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

String Ku

Contoh berikut memberikan gambaran yang lebih jelas. Perhatikan bahwa urutan parameter tidak penting.

```
def printinfo( nama, umur ):
   "Mencetak  sebuah posisi string ke dalam fungsi"
   print "Nama: ", nama
   print "Umur: ", umur
   return;

# Sekarang dapat memanggil fungsi  printinfo
printinfo( umur=20, nama="Emi" )


```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Nama: Emi

Umur: 20

d. Argumen default

Argumen default adalah argumen yang mengasumsikan nilai default jika nilai tidak disediakan dalam fungsi panggilan untuk argumen tersebut. Contoh berikut memberikan ide tentang argumen default, mencetak usia default jika tidak lulus

```
1 def printinfo( nama, umur = 25 ):
2    "Mencetak  sebuah posisi string ke dalam fungsi"
3    print "Nama: ", nama
4    print "umur ", umur
5    return;
6
7 # sekarang dapat memanggil fungsi printinfo
8 printinfo( umur=20, nama="Emi" )
9 printinfo( nama="Emi" )
10
11
```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Nama: Emi

umur 20

Nama: Emi

umur 25

e. Argumen Panjang Variabel

Kemungkinan memproses fungsi untuk lebih banyak argumen daripada yang ditentukan saat mendefinisikan fungsi. Argumen ini disebut argumen *variabel-panjang* dan tidak disebutkan dalam definisi fungsi, tidak seperti argumen yang diperlukan dan default. Sintaksis untuk fungsi dengan argumen variabel non-kata kunci adalah sebagi berikut.

```
def functionname([formal_args,] *var_args_tuple ):
   "function_docstring"
   function_suite
   return [expression]
```

Tanda bintang (*) ditempatkan di depan nama variabel yang memegang nilai semua argumen variabel nonkeyword. Tuple ini tetap kosong jika tidak ada argumen tambahan yang ditentukan selama panggilan fungsi. Berikut ini adalah contoh sederhana

```
def printinfo( arg1, *vartuple ):
   " mencetak argumen melalui  variabel"
   print "Hasilnya: "
   print arg1
   for var in vartuple:
      print var
   return;

# sekarang dapat memanggil fungsi printinfo
printinfo( 20 )
printinfo( 90, 80, 70 )

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Hasilnya:

20

Hasilnya:

90

80

70

f. Fungsi *Anonim*

Fungsi-fungsi ini disebut anonim karena mereka tidak dinyatakan dengan cara standar dengan menggunakan kata kunci *def.* Anda dapat menggunakan kata kunci *lambda* untuk membuat fungsi anonim kecil.

1) Bentuk Lambda dapat mengambil sejumlah argumen tetapi mengembalikan hanya satu nilai dalam bentuk ekspresi. Mereka tidak dapat berisi perintah atau beberapa ekspresi.

2) Fungsi anonim tidak bisa menjadi panggilan langsung untuk dicetak karena lambda membutuhkan ekspresi.

3) Fungsi Lambda memiliki ruang nama lokal mereka sendiri dan tidak dapat mengakses variabel selain yang ada dalam daftar parameter mereka dan yang ada di ruang nama global.

4) Meskipun tampaknya lambda adalah versi satu baris dari suatu fungsi, mereka tidak setara dengan pernyataan inline di C atau C ++, yang

tujuannya adalah dengan melewati alokasi tumpukan fungsi selama pemanggilan karena alasan kinerja.

g. Sintaksis

Sintaks fungsi *lambda* hanya berisi satu pernyataan, yaitu sebagai berikut :

lambda [arg1 [,arg2,.....argn]]:expression

Berikut ini adalah contoh untuk menunjukkan bagaimana bentuk fungsi *lambda* bekerja

```
jumlah = lambda arg1, arg2: arg1 + arg2;

# sekarang dapat memanggil  sebuah fungsi  sum as
print " Total  nilai: ", jumlah( 5, 10 )
print " Total  nilai: ", jumlah( 15, 22 )

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Total nilai: 15

Total nilai: 37

h. Pernyataan kembali

Pernyataan kembali [ekspresi] keluar dari fungsi, opsional melewati kembali ekspresi ke penelepon. Pernyataan kembali tanpa argumen sama dengan mengembalikan tidak ada. Semua contoh di atas tidak mengembalikan nilai apa pun. Anda dapat mengembalikan nilai dari fungsi sebagai berikut :

```
def jumlah( arg1, arg2 ):
   # Menambah kedua parameter dan mngembalikan."
   total = arg1 + arg2
   print "di dalam function : ", total
   return total;

# Now you can call sum function
total = jumlah( 7, 21 );
print "di luar function : ", total

```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

di dalam function : 28

di luar function : 28

i. Lingkup Variabel

Semua variabel dalam suatu program mungkin tidak dapat diakses di semua lokasi dalam program itu. Ini tergantung di mana Anda telah menyatakan variabel. Ruang lingkup variabel menentukan bagian dari program di mana Anda dapat mengakses pengenal tertentu. Ada dua lingkup dasar variabel dalam Python yaitu variabel global dan lokal

j. Variabel global versi Lokal

Variabel diartikan dalam badan fungsi mempunyai ruang lingkup lokal, dan yang didefinisi di luaryang mempunyai ruang lingkup global. Artinya dalam variabel lokal hanya mampu diakses di dalam fungsi dimana dideklarasikan, sedangkan variabel global mampu diakses pada semua badan program oleh semua fungsi. Ketika memanggil fungsi, variabel yang dinyatakan di dalamnya dibawa ke ruang lingkup. Berikut ini adalah contoh sederhana dibawah ini.

```
1 total = 0; # ini variabel global
2 # disini mendefinisikan fungsi
3 def jumlah( arg1, arg2 ):
4    # Add both the parameters and return them."
5    total = arg1 + arg2; # Here total is local variable.
6    print "Di dalam total fungsi lokal : ", total
7    return total;
8
9 # Now you can call sum function
10 jumlah( 7, 12 );
11 print "Di luar total fungsi lokal : ", total
12
13
```

Ketika kode di atas dieksekusi, ia menghasilkan hasil berikut :

Di dalam total fungsi lokal : 19

Di luar total fungsi lokal : 0

4. Rekursif

Python menerima rekursi fungsi, yang berarti fungsi dapat memanggil dirinya sendiri. Rekursi merupakan dasar ilmu matematika dan pemrograman umum. Berarti sebuah fungsi memanggil dirinya sendiri. dan punya manfaat

dapat mengulang/looping data untuk mencapai hasil. Pengembang harus sangat berhati-hati dengan rekursi karena bisa sangat mudah untuk menyelinap ke dalam menulis fungsi yang tidak pernah berakhir, atau salah satu yang menggunakan kelebihan jumlah memori atau kekuatan prosesor. Namun, ketika ditulis rekursi benar dapat meakukan pendekatan yang sangat efisien dan matematis-elegan untuk pemrograman.

Pada contohnya, tri_recursion () merupakan fungsi dimana telah definisikan untuk memanggil dirinya sendiri ("recurse"). Penggunakn variabel k sebagai data, yang decrements(-1) setiap kali berulang. Rekursi berakhir ketika kondisinya tidak lebih besar dari 0 (yaitu ketika 0). Untuk pengembangan baru, perlu beberapa waktu untuk mencari tahu cara kerjanya, cara terbaik untuk mengetahuinya ialah mengujinya dan memodifikasi.

Penjelasan tentang sesuatu yang mengacu pada dirinya sendiri disebut definisi rekursif. Di formulasi terakhir kami, algoritma pencarian biner menggunakan deskripsinya sendiri. Sebuah "panggilan" untuk pencarian biner "berulang" di dalam definisi karenanya, label “Definisi rekursif.” Sekilas, Anda mungkin berpikir definisi rekursif hanya omong kosong. Pasti Anda pernah memiliki seorang guru yang bersikeras bahwa Anda tidak dapat menggunakan satu kata pun di dalamnya definisi? Itu disebut definisi melingkar dan biasanya tidak terlalu berarti kredit pada ujian.

Dalam matematika, bagaimanapun, definisi rekursif tertentu digunakan sepanjang waktu. Selama kita berhati-hati dalam perumusan dan penggunaan definisi rekursif, mereka bisa sangat berguna dan sangat kuat. Rekursif klasik Contoh dalam matematika adalah efinisi faktorial. Didefinisikan faktorial dari nilai seperti ini:

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

Misalnya, menghitung 5!

$$5! = 5(4)(3)(2)(1)$$

Harus dingat bahwa mengimplementasikan program untuk menghitung faktorial menggunakan loop sederhana yang mengakumulasi produk faktorial.

Melihat perhitungan 5! hubungan ini memberi cara lain untuk mengungkapkan apa adanya yang dimaksud dengan faktorial secara umum.

Rekursi sebagai contohnya, saat dua cermin berada paralel antara satu dengan yang lain, gambar yang tertangkap adalah suatu bentuk rekursi tak-terbatas. Secara spesifik hal ini mendefinisikan suatu instansi tidak terbatas dengan menggunakan ekpresi terbatas beberapa instansi bisa merujuk ke instansi lainnya, tetapi dengan suatu cara sehingga tidak ada perulangan atau keterkaitan yang tidak terbatas dapat terjadi. Berikut ini contoh sederhana fungsi rekursif.

Python:

```
def fungsiRekursif():
    print("Hello, ini fungsi rekursif")
    fungsiRekursif()
```

Javascript:

```
function fungsiRekursif(){
    console.log("Hello, ini fungsi rekursif");
    fungsiRekursif();
}
```

Contoh fungsi di atas menampilkan teks "Hello, ini fungsi rekursif" terus menerus, disebabkan pemanggilan diri sendiri tanpa berhenti. Fungsi rekursif mampu menyelesaika berbagai persoalan sebagai contoh dalam perhitungan bilangan fibbonaci dan faktorial. Rekursif dapat menggunakan untuk menghitung deret fibonacci. Deret fibonacci merupakan deret angka sederhana yang susunan angka penjumlahan dari dua angka sebelumnya (0,1,1,2,3,5,8,13,21,…dst).

Deret Fibonacci didefinisikan sebagai:

$F_n = F_{n-1} + F_{n-2}$

dimana $F_0 = 0$

and $F_1 = 1$

Membuat fungsi fibonacci pada python cukup mudah. berikut syntaxnya.

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

fib(7)
```

Output yang dihasilkan adalah 13.untuk fungsi fibonacci diatas.

Dapat dilihat bahwa definisi rekursif tidak melingkar karena setiap aplikasi menyebabkan kita meminta faktorial dari bilangan yang lebih kecil. Akhirnya kita dapatkan down to, yang tidak memerlukan aplikasi definisi lain. Ini adalah disebut kasus dasar untuk rekursi. Ketika rekursi berhenti, kita mendapatkan sebuah ekspresi tertutup yang bisa langsung dihitung. Semua definisi rekursif yang baik memiliki karakteristik utama berikut:

a. Ada satu atau lebih kasus dasar yang tidak memerlukan rekursi.

b. Semua rantai rekursi akhirnya berakhir di salah satu kasus dasar.

Cara termudah untuk menjamin bahwa kedua syarat ini terpenuhi adalah dengan membuatnya pastikan bahwa setiap rekursi selalu terjadi pada versi yang lebih kecil dari masalah aslinya. Versi sangat kecil dari masalah yang dapat diselesaikan tanpa rekursi kemudian menjadi kasus dasar. Persis seperti inilah definisi faktorial bekerja.

Program rekursif harus dinyatakan dalam prosedur atau fungsi, karena hanya prosedur dan fungsi yang dapat dipanggil dalam sebuah program. Fungsi atau Prosedur Rekursif disusun oleh dua komponen :

a. Basis (komponen basis) fungsi : Menghentikan rekursif dan memberi nilai yang terdefinisi

b. Aturan rekursif (komponen induksi) fungsi : mendefinisikan dengan dirinya sendiri

Aturan rekursif dapat memungkinkan komputasi tak berhingga bila komputasi tidak mencapai komponen basis. Pemrogram harus berhati-hati dalam membuat definisi rekursif. Definisi harus menunjukkan pergerakan yang semakin menuju ke objek yang lebih sederhana sehingga memungkinkan

deinisi tersebut berhenti. Studi Kasus Rekursif pada contoh Program menghitung factorial :

Permasalahan : 5! = 5 x 4 x 3 x 2 x 1

= 120

```
2
3 a = raw_input("masukkan nilai faktorial yang ingin dicari =")
4 c = int(a)
5 for i in range(c):
6         if i==0:
7
8            c=c*1
9         else :
10           c=c*i
11           print (c)
```

```
In [4]: %run "D:\kuliah\STRUKTUR-DATA\Python\faktorial.py"

masukkan nilai faktorial yang ingin dicari =5
5
10
30
120

In [5]:
```

Hal perlu diperhatkansaat melakukan proses rekursif yakni dengan memastikan batas atau kondisi berhenti berproses pada rekursif. Pada fungsi fact di atas kondisi berhenti jika n=0. Sebelum menerapkan proses rekursif harus yakin bahwa syarat berhenti dapat dicapai, jangan tidak henti. Untuk itu, berbagai kondisi yang dapat menyebabkan rekursif tak berujung harus dapat ditangani (contoh apabila masukkan n<0). Proses rekursif tak berujung akan menyebabkan memori habis, dan akan menglami gangguan.

5. Anonymous/Lambda Function

Anonymous Function atau Lambda Function adalah fungsi yang didefinisikan tanpa nama. fungsi biasa diawali dengan def, akan tetapi lambda function diawali dengan lambda. pada dasarnya fungsi lambda memiliki perilaku yang sama dengan fungsi biasa, kita bisa memasukkan berapapun parameter akan tetapi hanya dapat memiliki satu ekspresi atau perintah dan

mengembalikan nilainya. bahkan fungsi ini ditulis dalam satu baris kode saja. berikut adalah syntax lambda function.

Diawali dengan keyword lambda lalu parameter, titik dua, dan perintah. Kita dapat memasukkan fungsi ini kedalam variabel agar dapat digunakan. berikut contoh fungsi lambda.

```
double = lambda x: x * 2
x = lambda a, b : a * b
print(double(5))
print(x(5, 6))
```

Output yang dihasilkan 10 30 untuk fungsi lambda diatas. biasanya lambda function digunakan pada fungsi yang parameternya adalah fungsi seperti map(), dan filter()

6. Fungsi dalam Python

Fungsi yang didefinisikan dalam fungsi lain dikenal sebagai fungsi dalam atau fungsi bersarang. Fungsi bersarang dapat mengakses variabel dari lingkup melampirkan. Fungsi bersarang digunakan sehingga dapat dilindungi dari segala sesuatu yang terjadi di luar fungsi, yang berguna dalam penentuan fungsi dimana sebagai peneyedia yang dierlukan programmer. Berikut adalah aturan sederhana untuk menentukan fungsi dalam Python.

a. Blok fungsi diawali dengan key word def selanjutnya diikui dengan nama fungsi dan tanda kurung (()).

b. Masing-masing parameter input atau argumen penempartannya dalam kurung. Programmer bisa membuat parameter di tanda kurung

c. Pernyataan kesatu dari fungsi berupa pernyataan opsional - string dokumentasi fungsi atau *docstring.*

d. Blok kode dalam setiap fungsi diawali dengan dengan tanda (:) serta diindentasinya.

e. Pernyataan kembali [ekspresi] keluar dari fungsi, opsional melewati kembali ekspresi celled. Pernyataan kembali tanpa argumen sama dengan mengembalikan tidak ada.

Sintaksis

```
def namafungsi( parameter ):
   "function_docstring"
   function_suite
   return [ekspresi]
```

Secara default, parameter memiliki perilaku posisional dan Anda perlu memberi tahu mereka dalam urutan yang sama dengan yang didefinisikan.

Contoh

Fungsi berikut mengambil string sebagai parameter input dan mencetaknya pada layar standar.

```
def printme( str ):
   "This prints a passed string into this function"
   print str
   return
```

C. LATIHAN SOAL

1. Jelaskan menurut Anda mengenai fungsi!
2. Jelaskan perbedaan program dengan rekursif dan interatif!
3. Jelaskan kekurangan dan kelebihan apabila menggunakan rekursif dan interatif!
4. Buatlah program pemangkatan dengan menggunakan rekursif!
5. Tuliskanlah sebuah fungsi yang menjumlahkan dua buah integer, dan menghasilkan sebuah integer, dengan membuat definisi rekursif dari penjumlahan!

D. REFERENSI

Amanulhaq, A. A. (2021). Implementasi Algoritma Image Hashing Dan Hamming Distance Untuk Deteksi Kemiripan Gambar. *repository ITS*.

Basant Agarwal, B. B. (2018). Hand-On Data Structures and Algorithms With Python. London: Packt Publishing.

Emi Sita Eriana, A. Z. (2021). Praktikum Algoritma dan Pemrograman. Tangerang Selatan: Unpam Press.

Jodi, u. R. (2020). Algoritma Dan Struktur Data.

Mohamad Aslam Katahman, M. F. (2021). Pembangunan Aplikasi Realiti Terimbuh Untuk Pengenalan Struktur Data. Information Technology And Computer Science.

Nasrullah, A. H. (2021). IMPLEMENTASI ALGORITMA DECISION TREE UNTUK KLASIFIKASI PRODUK LARIS. Jurnal Ilmiah Ilmu Komputer i.

Peng Qi, Y. Z. (2020). Stanza: A Python Natural Language Processing Toolkit for Many Human Languages.

Sianipar, R. H. (2013). Pemrograman & Struktur Data C: Belajar dari Contoh Untuk Programmer Pemula Maupun Berpengalaman. Penerbit INFORMATIKA, 2013.

Thanaki, J. (2017). Python Natural Language Processing. Mambai.

Zein, A. (2018). Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka OPENCV dan DLIB PYTHON. Sainstech : Jurnal Penelitian dan Pengkajian Sains dan Teknologi.