

**LAPORAN PRAKTIKUM ALGORITMA DAN PEMROGRAMAN
KOMPUTER
MODUL 9
FUNCTION AND MODULES**

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah
Praktikum Basis Data



Disusun Oleh :

AHSAN MAULANA RIZQI

104062400071

S1 BD 04 B

**PROGRAM STUDI S1 BISNIS DIGITAL
TELKOM UNIVERSITY PURWOKERTO
2025**

1. Dasar Teori

1.1 Function

Function atau fungsi adalah blok kode terorganisir yang dapat digunakan kembali untuk melakukan sebuah tindakan/action. Fungsi terdiri dari dua jenis, yaitu fungsi yang mengembalikan nilai dan fungsi yang tidak mengembalikan nilai. Perbedaan sederhana dari kedua jenis fungsi tersebut adalah fungsi yang mengembalikan nilai menggunakan keyword return sedangkan fungsi yang tidak mengembalikan nilai biasanya tidak menggunakan keyword return. Salah satu manfaat dari membuat function adalah kode program lebih terstruktur dan reusable atau bisa digunakan berulang kali.

Keuntungan penggunaan fungsi antara lain sebagai berikut:

- Mengurangi duplikasi kode.
- Membuat masalah yang rumit menjadi lebih sederhana.
- Meningkatkan kejelasan kode.
- Penggunaan kembali kode.

Aturan sederhana untuk mendefinisikan fungsi dengan Python adalah sebagai berikut:

- Fungsi dimulai dengan “def”, kata kunci diikuti oleh nama fungsi dan tanda kurung ()
- Setiap parameter masukan atau argument harus ditempatkan di dalam tanda kurung tersebut
- Blok kode dalam setiap fungsi dimulai dengan titik dua : dan indentasi
- Pernyataan kembali (return) [ekspresi] keluar dari sebuah fungsi, secara opsional menyampaikan kembali ekspresi ke pemanggil
- Pernyataan pengembalian tanpa argumen sama dengan return None
- Fungsi harus didefinisikan sebelum penggunaannya

1.2 Modules

Module adalah objek Python dengan atribut yang diberi nama dan bisa dijadikan referensi. Module adalah file yang terdiri dari kode program Python yang bisa mendefinisikan fungsi, kelas, dan variabel. Module bisa berinteraksi dengan module yang lain layaknya seperti kode program Python biasa yang disimpan dengan ekstensi .py.

Module adalah file Python dengan ekstensi .py yang dapat diimpor ke dalam file sumber Python lain. Module juga bisa menyertakan kode yang bisa dijalankan (“runnable”).

Module berisi:

- a. Definisi dan implementasi kelas
- b. Variabel, dan
- c. Fungsi yang dapat digunakan di dalam program lain

Python module yang sering digunakan adalah sebagai berikut:

- sys: informasi tentang Python itu sendiri (path, dll.).
- os: fungsi sistem operasi.
- os.path: alat nama path portabel.
- shutil: utilitas untuk menyalin file dan pohon direktori.
- cmp: utilitas untuk membandingkan file dan direktori.
- glob: menemukan file yang cocok dengan pola wildcard.
- re: pencocokan string ekspresi reguler.
- time: penanganan waktu dan tanggal.
- datetime: implementasi cepat penanganan tanggal dan waktu.
- pdb: debugger.
- hotshot: pembuatan profil kode.
- math, cmath: fungsi matematika (riil dan kompleks) lebih cepat untuk skalar.
- random: generator acak (demikian pula).
- gzip: membaca dan menulis file yang di-gzip.
- struct: fungsi untuk mengemas dan membongkar struktur data biner.
- types: nama untuk semua tipe Python standar.
- doctest, unittest: modul yang memfasilitasi pengujian unit.
- getopt, optparse: utilitas untuk menangani penguraian argumen tingkat shell.
- pickle, cpickle, marshal, shelve: digunakan untuk menyimpan objek dan kode ke dalam file.
- StringIO, cStringIO: objek seperti string yang dapat dibaca dan ditulis sebagai file (misalnya, file dalam memori).

2. Pembahasan Tugas Guided

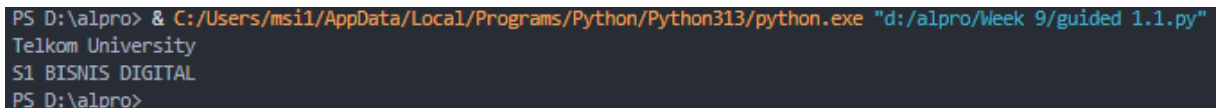
2.1 Fungsi sederhana

Kode pada gambar 1 termasuk sebuah fungsi yang tidak mengembalikan nilai, untuk menjalankan fungsi tersebut dengan menuliskan nama fungsinya. Nama fungsinya namaKampus():. Body fungsi print("Telkom University"), print("S1 BISNIS DIGITAL").



```
alpro - guided
1 def namaKampus():
2     print ("Telkom University")
3     print ("S1 BISNIS DIGITAL")
4
5 namaKampus()
```

Gambar 1. Fungsi sederhana



```
PS D:\alpro> & C:/Users/ms11/AppData/Local/Programs/Python/Python313/python.exe "d:/alpro/Week 9/guided 1.1.py"
Telkom University
S1 BISNIS DIGITAL
PS D:\alpro>
```

Gambar 2. Hasil output kode fungsi sederhana

2.2 Fungsi yang mengembalikan nilai

Kode pada gambar 3 termasuk sebuah fungsi yang mengembalikan nilai, ciri-cirinya adalah menggunakan *keyword return*, untuk menjalankan fungsi tersebut dengan menuliskan nama fungsinya kemudian gunakan fungsi *print(nama_fungsi)*.



```
alpro - guided 1.2.py
1 def luasLingkaran():
2     jari_jari = 2
3     phi = 3.14
4     luas = phi * jari_jari * jari_jari
5     return luas
6
7 hasil = luasLingkaran() # nilai hasil dari variabel harus disimpan ke suatu variabel
8 print (hasil)
9
10 print (luasLingkaran()) # atau langsung dicetak dengan print
```

Gambar 3. Fungsi return

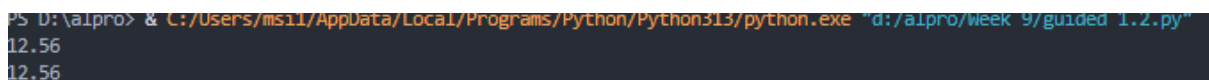
Nama fungsi : luasLingkaran():.

Body function : jari_jari = 2, phi = 3,14, luas = phi * jari_jari * jari_jari.

Return function : luas.

Variabel hasil untuk menyimpan nilai dari fungsi luasLingkaran().

Hasil pada gambar 4 dari print (hasil) dan print (luasLingkaran()).

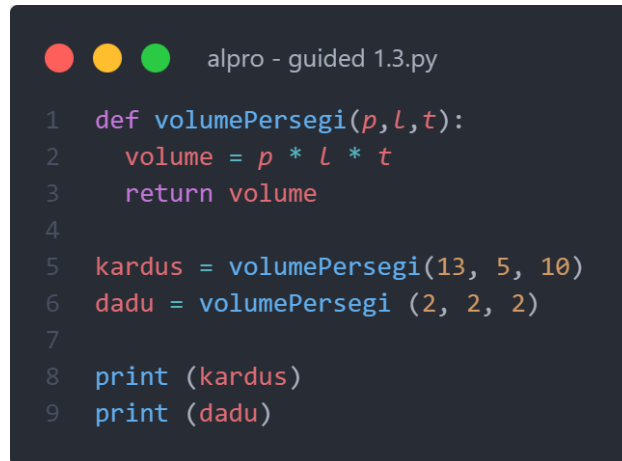


```
PS D:\alpro> & C:/Users/ms11/AppData/Local/Programs/Python/Python313/python.exe "d:/alpro/Week 9/guided 1.2.py"
12.56
12.56
```

Gambar 4. Hasil kode fungsi return

2.3 Fungsi yang mengembalikan nilai dan menggunakan argument

Kode pada gambar 5 termasuk kedalam sebuah fungsi yang mengambil nilai dan nilai tersebut yang berasal dari argumen fungsi akan diproses. Cara untuk menjalankan fungsi tersebut dengan menuliskan kembali nama fungsi tersebut kemudian gunakan fungsi `print (nama_fungsi(argument_1, argument_2, ...))` diikuti dengan nilai argument yang akan diproses.



```
1 def volumePersegi(p,l,t):
2     volume = p * l * t
3     return volume
4
5 kardus = volumePersegi(13, 5, 10)
6 dadu = volumePersegi (2, 2, 2)
7
8 print (kardus)
9 print (dadu)
```

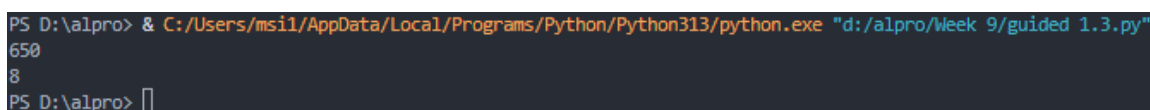
Gambar 5. Code function return with parameter

Nama fungsi : `volumePersegi()`. Argument : `p, l, t`. Body fungsi : `volume : p*l*t`. Nilai yang dikembalikan : `volume`.

Terdapat 2 variabel untuk menyimpan nilai fungsi yang pertama terdapat variabel `kardus = volumePersegi (13, 5, 10)` dengan nilai 13 sebagai arguamen `p`, nilai 5 sebagai argument `l`, dan nilai 10 sebagai argument `t`. Adapun variabel kedua yaitu `dadu = volumePersegi (2, 2, 2)` dengan nilai 2 sebagai argument `p`, nilai 2 sebagai argument `l`, nilai 2 sebagai argument `t`.

Fungsi `print (kardus)` dan `print (dadu)` digunakan untuk memunculkan hasil dari kedua variabel tersebut ke layer atau terminal.

Hasil pada gambar 6 menunjukkan nilai dari perhitungan variabel `volume` yang berisikan perkalian antar variabel dari argument `p,l,t` yang terdapat pada fungsi `volumePersegi()`. Hasil dari variabel pertama adalah 650 yang dihasilkan dari $13*5*10$. Adapun hasil dari variabel kedua adalah 8 yang dihasilkan dari $2*2*2$.



```
PS D:\alpro> & C:/Users/msil/AppData/Local/Programs/Python/Python313/python.exe "d:/alpro/Week 9/guided 1.3.py"
650
8
PS D:\alpro> 
```

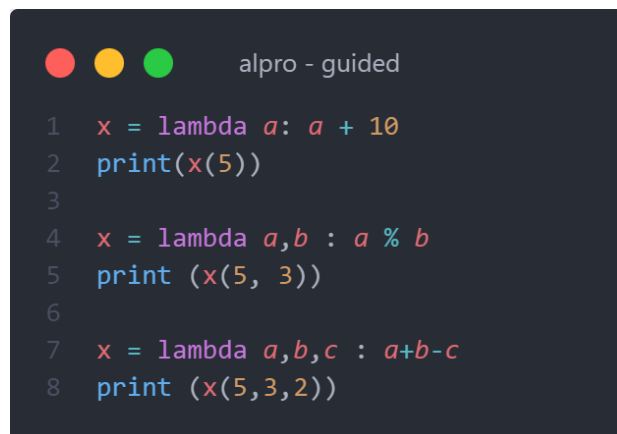
Gambar 6. Output code function return with parameter

2.4 Fungsi lamda

Lambda merupakan *function small anonymous* atau fungsi yang tidak memiliki nama fungsi dan hanya dapat mengproses satu ekspresi. Adapun perbedaan dari fungsi

def adalah fungsi def memiliki nama dan dapat memproses lebih dari satu ekspresi. Keduanya sama sama dapat memiliki lebih dari satu argument.

Pada gambar 7 terdapat contoh penggunaan fungsi lambda. Yang pertama $x = \text{lambda } a : a + 10$, variabel x untuk menyimpan fungsi lambda yang memiliki argument a dan proses eksekusi $a + 10$. Yang kedua $x = \text{lambda } a, b : a \% b$, variabel x untuk menyimpan fungsi lambda yang memiliki argument a dan b dan proses eksekusi $a \% b$ (a dimoduluskan atau sisa bagi b). dan yang ketiga $x = \text{lambda } a, b, c : a + b - c$, variabel x untuk menyimpan fungsi lambda yang memiliki argument a, b, dan c dan proses eksekusi $a + b - c$.

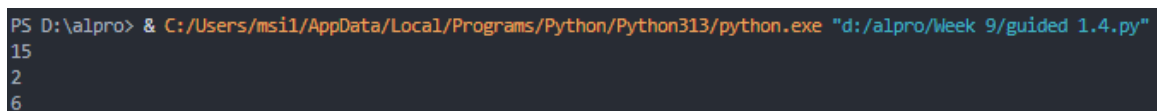


```
alpro - guided
1 x = lambda a: a + 10
2 print(x(5))
3
4 x = lambda a,b : a % b
5 print (x(5, 3))
6
7 x = lambda a,b,c : a+b-c
8 print (x(5,3,2))
```

Gambar 7. Function lambda

Fungsi `print (x(5))`, `print (x(5,3))`, dan `print (x(5,3,2))` digunakan untuk memunculkan hasil dari masing masing fungsi lambda.

Hasil pada gambar 8 merupakan hasil dari 3 contoh penggunaan fungsi lambda. Yang pertama 15 adalah hasil dari variabel argument $a + 10$ yang dimana nilai dari argument adalah 5. Yang kedua 2 adalah hasil dari variabel parameter $a \%$ parameter b yang dimana nilai dari parameter a adalah 5 dan parameter b adalah 3. Yang ketiga 6 adalah hasil dari variabel parameter $a +$ parameter b – parameter c yang dimana nilai dari parameter a adalah 5, parameter b adalah 3, dan parameter c adalah 2.



```
PS D:\alpro> & C:/Users/msil/AppData/Local/Programs/Python/Python313/python.exe "d:/alpro/Week 9/guided 1.4.py"
15
2
6
```

Gambar 8. Output function lambda

2.5 File module : mtk.py

Kode pada gambar 9 adalah kumpulan fungsi yang digunakan sebagai *module*. Terdapat beberapa variabel dan fungsi didalam gambar ???. pertam terdapat kumpulan variabel dengan nama phi dan name yang menyimpan masing masing value yaitu 3.14159 dan “nurman”. Adapun fungsi diantaranya fungsi *hallo()*, fungsi *tambah()* dan fungsi *bagi()*.

```

alpro - mtk.py

1  phi = 3.14159
2  nama = "Budi"
3
4  def hello():
5      print ("Hello Nama saya ", nama, "!")
6
7  def tambah(angka1, angka2):
8      hasil = angka1 + angka2
9      return hasil
10
11 def bagi (angka1, angka2):
12     if angka2 != 0:
13         hasil = angka1 / angka2
14         return hasil
15     else:
16         return "error: Pembagian dengan nol"

```

Gambar 9. Module mtk.py

Dalam fungsi *hallo()* terdapat body function berupa *print* (“Hallo nama saya”, *name*, “!”) yang digunakan untuk memunculkan “hello nama saya (value variabel *nama*) !”. fungsi *tambah (angka_1, angka_2)* yang digunakan untuk menghitung nilai tambah dari sebuah parameter *angka_1* dan *angka_2*, di dalam fungsi ini juga terdapat variabel *hasil* yang digunakan untuk menyimpan perhitungan *angka_1* dan *angka_2*, value dari *hasil* akan dikembalikan menggunakan fungsi *return*. Adapun fungsi *bagi (angka_1, angka_2)* yang digunakan untuk menghitung pembagian antara *angka_1* dan *angka_2*, dalam fungsi ini juga terdapat percabangan dengan kondisi jika *angka_2* tidak sama dengan 0, maka eksekusi variabel *hasil* yang memiliki nilai dari hasil pembagian *angka_1* dan *angka_2* dan kembalikan *value* dari variabel *hasil*, jika salah maka kembalikan nilai “Error : pembagian dengan nol!”.

2.6 Import module

Kode pada gambar 11 menunjukkan cara penggunaan *module*. Pada baris pertama pada kode tersebut menunjukkan sebagai perintah untuk melakukan *import module* dan *mtk* sebagai nama dari modul atau nama *file* yang sebelumnya sudah dibuat. Sedangkan *import ** bermaksud untuk melakukan *import* semua fungsi dan variabel yang terdapat pada *module mtk*.



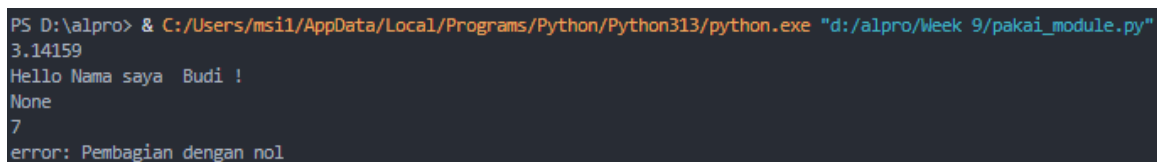
```

alpro - pakai_module.py
1  from mtk import *
2
3  print(phi)
4  print(hello())
5  print(tambah(5, 2))
6  print(bagi(3, 0))

```

Gambar 10. Import module mtk.py part 1

Fungsi `print(phi)`, `hallo()` yang terdapat pada *module mtk*, `print (tambah (5,2))` fungsi `tambah()` yang terdapat pada *module mtk*, `print (bagi (5,2))` fungsi `bagi()` yang terdapat pada *module mtk*. Fungsi ini digunakan untuk memunculkan hasil ke layar ataupun terminal. Hasil pada gambar ?? menunjukkan hasil dari pemanggilan *module mtk*.



```

PS D:\alpro> & C:/Users/msil/AppData/Local/Programs/Python/Python313/python.exe "d:/alpro/Week 9/pakai_module.py"
3.14159
Hello Nama saya Budi !
None
7
error: Pembagian dengan nol

```

Gambar 11. Output import module part 1

Kode pada gambar 12 dan hasil output pada gambar 13 terdapat variabel `phi = 3` dan setelahnya fungsi `print(phi)` dan menghasilkan output 3 kenapa tidak menghasilkan output 3.14159 yang terdapat pada *module mtk*, karena yang di panggil adalah variabel yang terbaru dan data `phi` dari *module mtk* tidak digunakan atau tidak dipanggil.



```

alpro - pakai_module2.py
1  from mtk import *
2
3  phi = 3
4  print (phi)
5
6  bil_1 = int(input("masukkan bilangan 1: "))
7  bil_2 = int(input("masukkan bilangan 2: "))
8
9  print ("Hasil Tambah", tambah(bil_1, bil_2))
10 print ("Hasil Bagi", bagi(bil_1, bil_2))

```

Gambar 12. Import module part 2

Variable `bil_1`: menyimpan input dari pengguna untuk bilangan 1 yang dikonversi menjadi integer dan variabel `bil_2`: menyimpan input dari pengguna untuk bilangan 2 yang dikonversi menjadi integer. Fungsi `print("Hasil Tambah",`

`tambah(bil_1, bil_2))` digunakan untuk mencetak hasil dari fungsi tambah dari file `mtk.py` dengan parameter `bil_1` dan `bil_2` dan fungsi `print("Hasil Bagi", bagi(bil_1, bil_2))` digunakan untuk mencetak hasil dari fungsi bagi dari file `mtk.py` dengan parameter `bil_1` dan `bil_2`.

```
PS D:\alpro> & C:/Users/msil/AppData/Local/Programs/Python/Python313/python.exe "d:/alpro/Week 9/pakai_module2.py"
3
masukkan bilangan 1: 12
masukkan bilangan 2: 12
Hasil Tambah 24
Hasil Bagi 1.0
PS D:\alpro> |
```

Gambar 13. Output import module part 2

3. Pembahasan Tugas Unguided

Buatlah program kalkulator sederhana yang terdiri dari dua file Python: `formula.py` dan `kalkulator.py`.

File `formula.py` berisi beberapa fungsi matematika dasar, yaitu tambah, bagi, pangkat, dan modulo. Anda dapat memodifikasi file sebelumnya. Pastikan setiap fungsi memiliki sanitasi input dan kondisi error jika diperlukan.

Pada file `kalkulator.py`, buatlah program utama yang mengimpor fungsi-fungsi dari `formula.py`. Program ini akan meminta dua input bilangan dari pengguna dan satu input operator berupa string: "+", "x", "^", atau "%". Gunakan percabangan untuk memanggil fungsi yang sesuai berdasarkan operator tersebut dan tampilkan hasilnya.

Test case:

Input	Output
2^3	Hasil: 8
3 % 2	Hasil: 1
7 % 0	Hasil: Error: Modulo dengan nol!
9 – 12	Hasil: Operator tidak dikenal
5 + -2	Hasil: 3

Hint: “*a,op,b = input ().split()*”

Pada gambar 14 menunjukan *module* yang berisi kumpulan fungsi yang terdiri dari fungsi `tambah()`, `kali()`, `pangkat()`, dan `modulus()` yang memiliki body fungsi mengembalikan nilai sesuai dengan nama dari fungsi tersebut. Terdapat fungsi yang memiliki kondisi khusus seperti fungsi `modulus()` yang dimana jika *value* argument variabel *b* sama dengan 0 maka kembalikan “Modulo dengan nol!”.



```

1  def tambah(a,b):
2      return a + b
3
4  def kali(a,b):
5      return a * b
6
7  def pangkat(a,b):
8      return a ** b
9
10 def modulus(a,b):
11     if b == 0:
12         return "Modulo dengan nol!"
13     else:
14         return a % b

```

Gambar 14. Module formula.py

Pada gambar 15 menunjukkan pengmagilan *module formula* yang di inialisasi sebagai *fm*. *While True* digunakan untuk pengulangan tanpa batas. Dua fungsi *print* setelahnya digunakan untuk memunculkan nilai ke layer. *a,op,b = input("Masukkan operasi (a op b): ").split()* digunakan untuk meminta input dari user dan *method split()* berfungsi untuk memisahkan keseluruhan input yang diterima, menjadi bagian-bagian terpisah yang ditandai oleh karakter spasi. Sehingga dengan fitur ini, kita dapat menggunakan format di *Test Case*, yaitu seperti satu baris input, namun yang terbaca oleh komputer bisa terpisah. Lalu variabel *a* dan *b* dikonversi ke tipe data *integer*.



```

1  import formula as fm
2
3  while True:
4      print("\n=====Kalkulator Sederhana=====")
5      print("format kalkulator adalah a operator b")
6      a,op,b = input("Masukkan operasi (a op b): ").split()
7      a = float(a)
8      b = float(b)
9      if op == "+":
10         print("Hasil: ", fm.tambah(a,b))
11     elif op == "^":
12         print("Hasil: ", fm.pangkat(a,b))
13     elif op == "%":
14         if b != 0:
15             print("Hasil: ", fm.modulus(a,b))
16         else:
17             print("Error: ", fm.modulus(a,b))
18     else:
19         print("Operator tidak valid!")
20     print ("=" * 35)
21     lanjut = input("Apakah Anda ingin melanjutkan? (y/n): ")
22     if lanjut.lower() != "y":
23         print("Terima kasih telah menggunakan kalkulator sederhana ini!")
24         break

```

Gambar 15. Code calculator with module formula.py

Fungsi *fm.tambah()*, *fm.pangkat()*, *fm.modulus()* digunakan untuk memanggil fungsi yang terdapat pada *module formula*. Terdapat percabangan dengan kondisi jika *op == "+"* maka munculkan “*Hasil : (hasil dari fungsi tambah())*”, *op == "^"* maka munculkan “*Hasil : (hasil dari fungsi pangkat())*”, *op == "%"* maka munculkan “*Hasil : (hasil dari fungsi modulus())*”.

Hasil pada gambar 16 sudah sesuai dengan *test case* yang sudah ditetapkan.

```
PS D:\alpro> & C:/Users/ms11/AppData/Local/Programs/Python/Python313/python.exe "d:/alpro/Week 9/unguided/kalkulator.py"

=====Kalkulator Sederhana=====
format kalkulator adalah a operator b
Masukkan operasi (a op b): 2 ^ 3
Hasil: 8.0
=====
Apakah Anda ingin melanjutkan? (y/n): y

=====Kalkulator Sederhana=====
format kalkulator adalah a operator b
Masukkan operasi (a op b): 3 % 2
Hasil: 1.0
=====
Apakah Anda ingin melanjutkan? (y/n): y

=====Kalkulator Sederhana=====
format kalkulator adalah a operator b
Masukkan operasi (a op b): 7 % 0
Error: Modulo dengan nol!
=====
Apakah Anda ingin melanjutkan? (y/n): y

=====Kalkulator Sederhana=====
format kalkulator adalah a operator b
Masukkan operasi (a op b): 9 - 12
Operator tidak valid!
=====
Apakah Anda ingin melanjutkan? (y/n): y

=====Kalkulator Sederhana=====
format kalkulator adalah a operator b
Masukkan operasi (a op b): 5 + -2
Hasil: 3.0
=====
Apakah Anda ingin melanjutkan? (y/n): n
Terima kasih telah menggunakan kalkulator sederhana ini!
PS D:\alpro>
```

Gambar 16. Output calculator same with test case

4. Ringkasan

Pada modul ini membahas tentang *function* dan *module* dalam bahasa pemrograman python. *Function* adalah blok nama yang dirancang untuk mengeksekusi perintah tertentu. Dengan menggunakan *function* dapat menyederhanakan tugas yang berulang dan meningkatkan kemudahan untuk uji coba. *Module* adalah file yang berisi kode (fungsi, kelas, variabel, dll) yang dapat diimport dan digunakan dalam program lain..

Di Python, *function* ditulis menggunakan kata kunci *def*, diikuti dengan nama *function* dan tanda kurung. Pada dalam tanda kurung tersebut dapat diberi parameter sebagai input untuk *function* tersebut. *Function* memiliki body yang berfungsi sebagai kode yang akan dieksekusi. Terdapat dua jenis *Function* yaitu *return function* dan *lambda function*

Function memberikan banyak manfaat, seperti membuat program lebih modular, meminimalkan penulisan kode yang berulang, serta meningkatkan keterbacaan program. *Function* juga dapat mengembalikan hasil melalui kata kunci *return*. Dalam praktikum ini juga dipelajari konsep parameter dan argumen, termasuk parameter biasa dan default,

serta konsep scope variabel, yaitu sejauh mana variabel dapat diakses dalam suatu program.

Pada modul ini juga memiliki tugas guided dan unguided yang membantu mahasiswa untuk lebih memahami penggunaan *function*. Sebagai contoh pada tugas guided 2.3 yang menjelaskan bagaimana cara kerja dan penulisan kode program. Sedangkan pada tugas unguided mahasiswa berlatih membuat kalkulator menggunakan *function* dan *module*. Pada tugas unguided melatih mahasiswa untuk membuat kode program sekreatif dan seefisien mungkin.

Dengan mengikuti praktikum ini, mahasiswa diharapkan memahami dasar-dasar penggunaan fungsi, mulai dari cara mendefinisikan hingga memanggil fungsi, serta menyadari pentingnya fungsi dalam menyusun program yang terstruktur, efisien, dan mudah untuk dikembangkan.