LAPORAN PRAKTIKUM METODE NUMERIK

DISUSUN OLEH :

NAMA : Ammar

NIM : 2024573010129

KELAS : TI 2B

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER

POLITEKNIK NEGERI LHOKSEUMAWE

2025/2026

LAPORAN PENGESAHAN

Nomor Praktikum	: 01
Judul Praktikum	: Basis Python
Nama Praktikan	: Ammar
NIM	: 2024573010129
Jurusan	: Teknologi Informasi dan Komputer
Prodi	: Teknik Informatika
Tanggal Praktikum	: 10 September 2025
Tanggal Penyerahan	: 1 Oktober 2025
Nilai	:
Keterangan	:

Buket Rata, 3 Oktober 2024

Dosen pengajar

Radhiyatammardhiyah, SST, M.Sc

NIP.197008021999031001

DAFTAR ISI

PENDAHULUAN

Perkembangan teknologi komputasi telah memberikan dampak yang sangat besar dalam bidang sains, teknik, dan pendidikan. Salah satu bahasa pemrograman yang banyak digunakan dalam menyelesaikan persoalan numerik maupun perhitungan ilmiah adalah **Python**. Python dipilih karena bersifat *open source*, memiliki sintaks yang sederhana, mudah dipelajari, serta didukung oleh berbagai pustaka (*library*) yang kuat dalam bidang komputasi, seperti *NumPy*, *SciPy*, dan *Matplotlib*.

Sebelum menggunakan Python untuk menyelesaikan permasalahan numerik yang lebih kompleks, diperlukan pemahaman dasar mengenai struktur dasar pemrogramannya. Beberapa konsep utama yang perlu dipahami antara lain:

1. **Variabel** – digunakan untuk menyimpan data dan nilai tertentu yang dapat berubah sesuai kebutuhan program.
2. **Tipe Data** – meliputi bilangan bulat (*integer*), bilangan riil (*float*), teks (*string*), hingga tipe data koleksi seperti *list*, *tuple*, dan *dictionary*.
3. **Operator dan Ekspresi** – digunakan dalam operasi aritmatika maupun logika untuk membangun perhitungan.
4. **Percabangan (Kondisi/If-Else)** – digunakan untuk mengambil keputusan berdasarkan suatu kondisi tertentu.
5. **Perulangan (Looping)** – memungkinkan program menjalankan instruksi berulang kali, misalnya dengan *for* atau *while*.
6. **Fungsi** – sekumpulan instruksi yang dikemas menjadi satu kesatuan agar program lebih terstruktur, efisien, dan mudah dipelihara.

Dengan memahami dasar-dasar tersebut, mahasiswa dapat membangun pondasi pemrograman yang kuat sebelum mengaplikasikan Python pada penyelesaian metode numerik. Praktikum ini diharapkan mampu memberikan gambaran awal mengenai cara kerja Python serta melatih keterampilan dalam menuliskan kode program sederhana, sehingga nantinya dapat diaplikasikan dalam pemecahan persoalan matematika dan teknik yang lebih kompleks.

BAB I
DAFTAR TEORI

1. 1 Pengertian Python

- Sejarah singkat Python
- Karakteristik Python (interpreted, high-level, multiparadigm, open-source)
- Kelebihan Python dalam bidang komputasi numerik

1. 2 Struktur Dasar Python

- Cara penulisan kode (indentasi sebagai penentu blok program)
- Komentar dalam Python

1. 3 Variabel dan Tipe Data

- Variabel: pengertian dan aturan penulisan nama variabel
- Tipe data primitif: integer, float, string, boolean
- Tipe data koleksi: list, tuple, dictionary, set

1. 4 Operator

- Operator aritmatika (+, -, *, /, **)
- Operator logika (and, or, not)
- Operator perbandingan (==, !=, >, <, >=, <=)

1. 5 Kondisi (Percabangan)

- if, elif, else
- Contoh penerapan pengambilan keputusan

1. 6 Perulangan (Looping)

- for loop
- while loop
- penggunaan break dan continue

1. 7 Fungsi

- Definisi fungsi dengan def
- Parameter dan argumen
- Nilai balik (*return*)
- Pentingnya fungsi dalam modularitas program

1. 8 Library Dasar Python untuk Numerik (opsional jika mau dikaitkan ke metode numerik)

- NumPy: array dan operasi vektor
- Matplotlib: visualisasi data sederhana

BAB II
HASIL

2. 1. Variabel dan tipe data

2.1.1 Variabel

Kode dan Output:

```
x = 5
y = "John"

print(x) # Hasil: 5
print(y) # Hasil: John

# Variabel tidak memiliki tipe data tertentu
x = 4       # x adalah tipe data integer
x = "Sally" # x sekarang adalah tipe data string
print(x) # Hasil: Sally
```

Analisis:

2.1.2 Casting

Kode:

```
# Jika anda ingin menentukan tipe data variabel, ini dapat dilakukan dengan casting

x = str(3)    # x akan menjadi '3'
y = int(3)    # y akan menjadi 3
z = float(3)  # z akan menjadi 3.0
```

Analisis:

2.1.3 Mendapatkan Tipe Data

Kode dan Output:

```python
x = 5
y = "Jogn"
print(type(x)) # hasil: <class 'int'>
print(type(y)) # hasil: <class 'str'>
```

Analisis:

2.1.4 Nama Variabel Multi Kata

Kode:

```python
# Nama Variabel Multi kata

# Variabel Camel
myVariabelName = "John" # Setiap kata setelah kata pertama dimulai dengan huruf kapital

# Variabel Pascal
MyVariabelName = "Doe" # Setiap kata dimulai dengan huruf kapital

# Variabel Snake
my_variabel_name = "Jane" # Setiap kata dipisahkan dengan garis bawah (_)
```

Analisis:

2.1.5 Banyak Nilai Ke Banyak Variabel

Kode Dan Output:

```python
# Banyak Nilai ke Banyak Variabel
x, y, z = "Orange", "Banana", "Cherry"

print(x) # hasil: Orange
print(y) # hasil: Banana
print(z) # hasil: Cherry
```

Analisis:

2.1.6 Satu nilai Ke banyak Variabel

Kode dan Output:

```python
# Satu nilai ke banyak variabel
x = y = z = "Orange"

print(x) # hasil: Orange
print(y) # hasil: Orange
print(z) # hasil: Orange
```

Analisis:

2.1.7 Variabel Keluaran

Kode dan Output:

```python
# Variabel Keluaran (output Variable)

# Untuk menggabungkan variabel keluaran, kita dapat menggunakan tanda koma (,) atau tanda plus (+).
x = "Awesome"

# Contoh penggunaan tanda koma (,)
print("Python is", x) # Output: Python is Awesome

# Contoh penggunaan tanda plus (+)
print("Python is " + x) # Output: Python is Awesome

# Anda juga dapat menggabungkan variabel keluaran dengan teks lain:
y = "Python is "
print(y + x) # Output: Python is Awesome

# Untuk angka, karakter (+) bertindak sebagai operator penjumlahan:
a = 5
b = 10
print(a + b) # Output: 15
```

Analisis:

2.1.8 Variabel Global

Output dan Kode:

```python
# Variabel yang dibuat di luar fungsi disebut variabel global
x = "awesome"

Windsurf: Refactor | Explain | Generate Docstring | ×
def myfunc():
  print("Python is " + x) # Akan mencetak var global = awesome
myfunc()

Windsurf: Refactor | Explain | Generate Docstring | ×
def myfunc2():
  x = "fantastic"
  print("Python is " + x) # Akan mencetak var lokal = fantastic
myfunc2()

print("Python is " + x) # Akan tetap sama = awesome
```

Analisis:

2.1.9 Angka Python

Kode dan Output:

```python
# Ada tiga tipe numerik di Python: int, float, dan complex.

x = 5           # int
y = 3.14        # float
z = 1j          # complex

# Untuk Memeriksa tipe data, gunakan fungsi type()
print(type(x))  # Output: <class 'int'>
print(type(y))  # Output: <class 'float'>
print(type(z))  # Output: <class 'complex'>

# Convert type data
m = float(a)        # Convert int ke float
n = int(d)          # Convert float ke int
o = complex(a)      # Convert int ke complex
print(m, n, o)      # Output: 1.0, 1, (1+0j)
print(type(m), type(n), type(o))  # Output: <class 'float'>, <class 'int'>, <class 'complex'>
```

Analisis:

2.1.10 Angka Acak

Kode Output:

```python
# Random Number (Angka Acak)

import random

print(random.randrange(1, 10))  # Menghasilkan angka acak antara 1 dan 9
```

Analisis:

2. 2 String

2.2.1 Memasukkan String ke Variabel

Kode Output:

```python
# Insert String to Variable  (Memasukkan String ke Variabel)

a = "hello"
print(a) # Output: hello

# String Multibaris
b = """Lorem ipsum dolor sit amet,
    consectetur adipiscing elit,
    sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.""" # Bisa menggunakan """ atau '''

print(b) # Output: Lorem ipsum dolor sit amet,
         # consectetur adipiscing elit, sed do eiusmod tempor incididunt
         # ut labore et dolore magna aliqua.
```

Analisis:

2.2.2 String adalah Array

Kode:

```python
# String adalah sebuah array dari karakter, namun karena python tidak memiliki tipe data char,
# maka karakter tersebut dianggap sebagai string dengan panjang 1 (a = 'a', b = 'b').
a = "Hello, World!"

# kita dapat mengakses karakter dalam string menggunakan indeks
print(a[1]) # Output: e -> Karena index dimulai dari 0
```

Analisis:

2.2.3 Perulangan pada string

Kode dan Output:

```
# Perulangan pada String
# Karena String adalah array, kita dapat melakukan perulangan
# pada setiap karakter dalam String menggunakan perulangan for.

A = "Banana"
for x in A:
    print(x) # Output: B, a, n, a, n, a -> dimana per char satu baris
```

Analisis:

2.2.4 Memeriksa panjang String

Kode dan Output:

```
# Panjang String Bisa di periksa dengan:
a = "Ammar"
print(len(a))  # Output: 5
```

Analisis:

2.2.5 Memeriksa String

Kode:

```
# Untuk Memeriksa apakah sebuah kata atau huruf ada dalam string tertentu:
txt = "Hello, welcome to my world."
print("Hello" in txt)  # Output: True

# atau, jika ingin menghasilkan output tertentu
if "Hello" in txt:
    print("Hello ada didalam variabel txt.") # Output: Hello ada didalam variabel txt
```

Analisis:

2.2.6 Mengiris Kata

Output dan hasil:

```python
# Di Python kita bisa mengiris kata dari sebuah Str,
# dengan menentukan index awal dan index akhir
txt = "Im live a happy life."
print(txt[3:7])     # Output: live
print(txt[10:20])   # Output: happy life
```

Analisis:

2.2.7 Modifikasi String dan

2.2.8 Menghapus spasi putih dan

2.2.9 Memisahkan string dan

2.2.10 Menggabungkan string

Kode dan Output:

```python
# Modifikasi String
a = "Ammar Shiddiq"

# Huruf Besar
print(a.upper()) # Output -> AMMAR SHIDDIQ

# Huruf kecil
print(a.lower()) # Output -> ammar shiddiq

# Menghapus Spasi Putih = adalah spasi pada awal dan akhir string
print(a.strip()) # Output -> "Ammar Shiddiq" bukan " Ammar Shiddiq"

# Memisahkan String
b = "Hello, Ammar"
print(b.split(",")) # Output -> ['Hello', ' Ammar']

# Menggabungkan String = bisa menggunakan operator tambah (+)
c = "Hello"
d = "World"
e = c + ", " + d # tambahkan spasi dan koma
print(e) # Output -> Hello, World
```

Analisis:

2. 3 Boolean dan Operator

2.3.1 Nilai boolean

Kode dan Output:

```python
# Boolean terbagi menjadi dua nilai, yaitu True dan False
print(10 > 9)  # True
print(10 == 9)  # False
print(10 < 9)  # False

# Menggunakan pernyataan if untuk mengecek nilai boolean
a = 200
b = 33
if b > a:
    print("b lebih besar dari a")  # Tidak TerOutput
else:
    print("b tidak lebih besar dari a")  # Output: b tidak lebih besar dari a
```

Analisis:

2.3.2 Kebanyakan nilai adalah benar

Kode dan Output:

```python
# Kebanyakan nilai di Python dianggap benar (True), kecuali beberapa nilai yang dianggap salah (False).
bool("abc")  # True
bool(123)  # True
bool(["apple", "cherry", "banana"])  # True

# Nilai-nilai yang dianggap salah (False) meliputi:
bool(False)  # False
bool(None)  # False
bool(0)  # False
bool("")  # False
bool(())  # False
bool([])  # False
bool({})  # False
```

Analisis:

2.3.3 Fungsi dapat mengembalikan boolean

Output dan Kode:

```python
# Fungsi bisa mengembalikan nilai boolean
Windsurf: Refactor | Explain | Generate Docstring | ×
def myfunc():
    return True
print(myfunc()) # Output: True

if myfunc():
    print("YES") # Output: YES
else:
    print("NO") # Tidak TerOutput

# python juga memiliki fungsi bawaan isInstance()
# yautu fungsi yang mengecek apakah objek memiliki tipe tertentu
# dan mengembalikan nilai boolean
x = 200
print(isinstance(x, int)) # Output: True
print(isinstance(x, str)) # Output: False
```

Analisis:

2. 4 List, Tuple, Set, dan Dictionary

2.4.1 List

Kode dan Output:

```python
# List adalah salah satu dari 4 tipe data yang ada di Python untuk menampung sekumpulan data
# List adalah tipe data yang paling fleksibel di Python

listBuah = ["apel", "jeruk", "mangga", "durian"]
print(listBuah) # Output: ['apel', 'jeruk', 'mangga, 'durian']
```

Analisis:

2.4.2 Tuple dan Set

Kode dan Output:

```python
# Tuple adalah kumpulan data yang bersifat immutable (tidak bisa diubah)
# Tuple menggunakan tanda kurung ()
tuplebuah = ("apel", "jeruk", "mangga", "pisang")
print(tuplebuah) # Output: ('apel', 'jeruk', 'mangga', 'pisang')

# Set adalah kumpulan yang tidak berurutan dan tidak memiliki elemen yang duplikat juga tidak memiliki indeks
# Set menggunakan tanda kurung kurawal {}
setbuah = {"apel", "jeruk", "mangga", "pisang"}
print(setbuah) # Output: {'apel', 'jeruk', 'mangga', 'pisang'}
```

Analisis:

2.4.3 Dictionary

Kode Output:

```python
# Dictionary adalah list yang memiliki key dan value
# Key adalah index yang kita buat sendiri
# Value adalah isi dari key tersebut
data_dict = {
    "USA": "Washington DC",
    "Indonesia": "Jakarta",
    "Malaysia": "Kuala Lumpur"
}

for key, value in data_dict.items():
    print(f"Key: {key}, Value: {value}") # Output: Key: USA, Value: Washington DC
                                         # Output: Key: Indonesia, Value: Jakarta
                                         # Output: Key: Malaysia, Value: Kuala Lumpur
```

Analisis:

2. 5 IF... ELSE

2.5.1 Kondisi Python dan Pernyataan IF

Kode dan Output:

```python
# Python mendukung kondisi logis yang biasa dari matematika
# seperti lebih besar dari (>), lebih kecil dari (<), sama dengan (==), tidak sama dengan (!=),
# lebih besar dari atau sama dengan (>=), dan lebih kecil dari atau sama dengan (<=).

# Kondisi ini bisa digunakan dalam berbagai cara, biasanya dalam pernyataan if untuk menentukan alur program.
# Contoh:
a = 33
b = 200
if b > a:
    print("b lebih besar dari a") # Output: b lebih besar dari a
```

Analisis:

2.5.2 elif

Kode:

```
# Elif adalah pernyataan bersyarat tambahan setelah if
# Elif hanya akan dieksekusi jika kondisi if sebelumnya bernilai False
a = 30
b = 20

if a > b:
    print("a lebih besar dari b") # Tidak TerOutput
elif a == b:
    print("a sama dengan b") # Output: a lebih besar dari b

# Jika pada bahasa lain dikenal dengan else if
# Maka pada Python dikenal dengan elif
```

Analisis:

2.5.3 Else

Kode dan Output:

```
# Else adalah sebuah blok opsional yang dapat ditambahkan setelah if atau elif.
# Blok else akan dieksekusi jika semua kondisi pada if dan elif bernilai False

nilai = 75
if nilai >= 85:
    print("Selamat! Anda mendapatkan nilai A.") # Tidak TerOutput
else:
    print("Anda harus belajar lebih giat lagi.") # Output: Anda mendapatkan nilai B
```

Analisis:

2.5.4 IF... Else Pendek

Kode:

```
# If else pendek merupakan cara singkat untuk menulis pernyataan if-else dalam satu baris.
# Ini berguna untuk kondisi sederhana di mana kita ingin menetapkan nilai berdasarkan kondisi tertentu.

# Contoh penggunaan if else pendek dua kondisi
a = 10
b = 10
print("A") if a > b else print("B") # Output: B

# Contoh penggunaan if else pendek tiga kondisi
print("A") if a > b else print("B") if a < b else print ("=") # Output: =
```

Analisis:

2.5.5 Nested IF

Kode:

```python
# Nested if (bersarang) adalah pernyataan if didalam if

x = 41
if x > 10:
    print("x lebih besar dari 10.") # Output: x lebih besar dari 10
    if x > 20:
        print("dan juga lebih besar dari 20.") # Output: dan juga lebih besar dari 20
    else:
        print("Tapi tidak diatas 20.") # Tidak TerOutput
```

Analisis:

2. 6 Perulangan While dan For

2.6.1 Break

Kode:

```python
# Break adalah command untuk menghentikan perulangan meskipun kondisi perulangan benar

i = 0
while i < 20:
    print(i)
    if i == 10:
        break

    i += 1
```

Analisis:

2.6.2 For

Kode:

```
# perulangan for digunakan untuk mengulangi urutan (yaitu daftar, tupel, kamus, set, atau string).

fruits = ["apple", "pine", "mellon"]

for fruit in fruits:
    print(fruit) # Output: apple, pine, mellon
```

Analisis:

2.6.3 Continue

Kode:

```
# Continue adalah command agar pada perulangan dapat melewati iterasi tertentu dan melanjutkan nya lagi

fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        continue
    print(x) # Output: apple, cherry
```

Analisis:

2.6.4 Fungsi range()

Kode:

```
# fungsi range() berfungsi untuk membuat daftar angka
for x in range(6): # range(6) bukan dari 0-6 tetapi 0-5
    print(x) # Output: 0, 1, 2, 3, 4, 5

# fungsi range() default index awal adlah 0, tetapi bisa juga diatur seperti:
for x in range(13-5, 10):
    print(x) # Output: 8, 9

# fungsi range() default menambah (+ 1) di setiap iterasi, tetapi bisa juga diatur seperti:
for x in range(13, 20, 5):
    print(x) # Output: 13, 18
```

Analisis:

2.6.5 Else di perulangan for

Kode:

```
# Else pada perulangan for adalah pernyataan yang akan dieksekusi ketika perulangan for telah selesai.
for x in range(6):
    print(x) # Output: 0, 1, 2, 3, 4, 5
else:
    print("Finally finished!") # Output: Finally finished!

# Note: Else tidak akan dieksekusi jika perulangan for dihentikan dengan break
for x in range(6,10):
    print(x) # Output: 6, 7, 8
    if x == 8: break
else:
    print("Finally finished!") # pernyataan ini tidak akan pernah tereksekusi
```

Analisis:

2.6.6 Nested Loop (perulangan bersarang)

Kode:

```
# PERULANGAN BERSARANG
# perulangan yang didalam perulangan akan tereksekusi setiap sekali perulangan luar.

adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y) # Output: red apple, red banana, red cherry,
                # big apple, big banana, big cherry,
                # tasty apple, tasty banana, tasty cherry
```

Analisis:

2. 7 Fungsi

2.7.1 Membuat fungsi

Kode:

```
# Membuat fungsi
Windsurf: Refactor | Explain | Generate Docstring | ×
def myFunc():
    print("Hello from a function")
```

Analisis:

2.7.2 Memanggil fungsi

Kode:

```python
def my_function():
    print("Hello, World.")

# Memanggil fungsi my_function
my_function() # Output: Hello, World
```

Analisis:

2.7.3 Argumen

Kode:

```python
# Argumen di fungsi
# Argumen di fungsi adalah nilai yang dikirimkan ke fungsi saat fungsi dijalankan
# Argumen di fungsi dapat berupa posisi, kata kunci, atau kata kunci berikut nilai default

Windsurf: Refactor | Explain | Generate Docstring | ×
def hafalan(juz, bro): # Argumen bisa ditambah sebanyak mungkin asal dipisah dengan koma (,)
    print(f"Hafalan {bro}: {juz} Juz")

hafalan(30, "Ammar") # Output: Hafalan Ammar: 30 Juz
# Argumen sering disingkat menjadi args dalam dokumentasi Python.
```

Analisis:

2.7.4 Argumen Berubah-ubah, (*args)

Kode:

```python
# Argumen yang berubah-ubah
# Jika Jumlah argumen tidak diketahui, Anda dapat menggunakan argumen *args
Windsurf: Refactor | Explain | Generate Docstring | ×
def hello(*nama): # dengan menambahkan * di depan argumen maka argumen tersebut menjadi tuple
    print(f"Hello {nama}")

hello("Ammar", "Shiddiq") # Output: Hello ('Ammar', 'Shiddiq')
```

Analisis:

2.7.5 Argumen Kata Kunci

Kode:

```python
# Argumen Kata kunci adalah dengan menambahkan kunci (nama variabel) di depan argumen:
def people(orang1, orang2, orang3):
    print(f"Orang yang mencurigakan adalah: {orang2}")

people(orang1="Ammar", orang2="Bros", orang3="Ramma")
```

Analisis:

BAB III
KESIMPULAN

3. 1. Bahasa Pemograman Python memiliki Syntaks yang sederhana dan mudah dipahami, sehingga cocok digunakan sebagai dasar pembelajaran pemogramaan penyelesaian metode numerik.

3. 2. Konsep dasar pemograman python meliputi variabel, tipe data, operator, kondisi, perulangan, dan fungsi. Pemahaman materi ini penting agar program dapat ditulis secara terstruktur dan efisien.

3. 3. Praktikum ini memberikan pengalaman langsung dalam menuliskan kode Python dasar serta menganalisis hasil eksekusi program, sehingga dapat memperkuat pemahaman teori yang telah dipelajari.

3. 4. Dasar-dasar Python yang telah dipelajari dapat menjadi landasan untuk mengembangkan aplikasi yang lebih kompleks, khususnya pada penerapan komputasi numerik dengan bantuan pustaka seperti NumPy dan Matloplib.