**PEMOGRAMAN BERORIENTASI OBJEK**

**JOB SHEET 1 : DASAR PEMOGRAMAN BAHASA PYTHON**

****

**Mata Kuliah :** Pemograman Berorientasi Objek

**Dosen :** Prayitno, S.ST., M.T., Ph.D

|  |  |
| --- | --- |
| **Nama** | **:** Abimanyu Gilar Waluyo |
| **NIM** | **:** 4.33.24.0.01 |
| **Kelas** | **:** TI-1A |

**PROGRAM STUDI D4-TEKNOLOGI REKAYASA KOMPUTER**

**JURUSAN TEKNIK ELEKTRO**

**POLITEKNIK NEGERI SEMARANG**

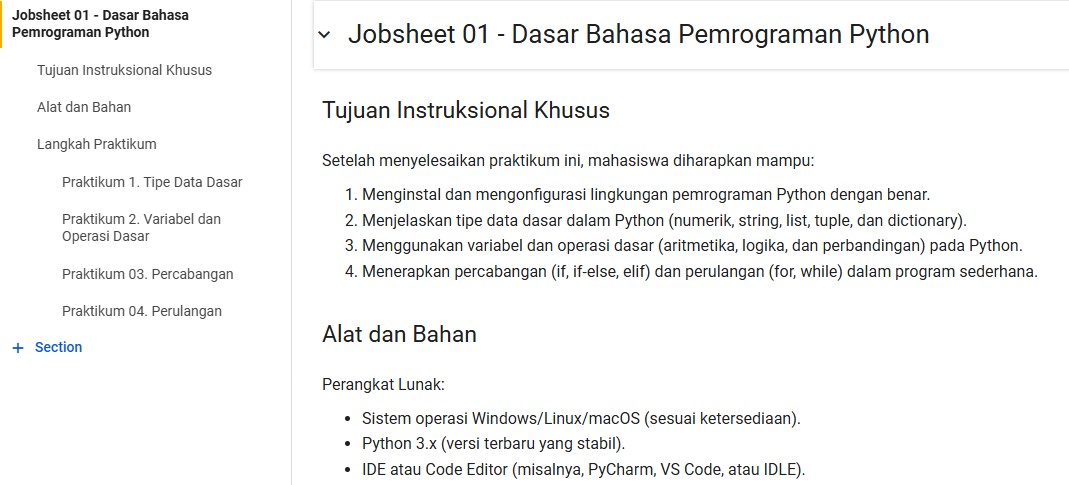
**2024/2025**

**POLITEKNIK NEGERI SEMARANG JURUSAN TEKNIK ELEKTRO**

# PROGRAM STUDI STR TEKNOLOGI REKAYASA KOMPUTER

**JOBSHEET 01:**

**DASAR BAHASA PEMROGRAMAN PYTHON**



**Dosen:**

Ir. Prayitno, S.ST., M.T., Ph.D.

**Nama Mahasiswa :** Abimanyu Gilar Waluyo

**NIM Mahasiswa :** 4.33.24.0.01

**Tahun Akademik 2025**

## A. Tujuan Instruksional Khusus

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

1. Menginstal dan mengonfigurasi lingkungan pemrograman Python dengan benar.
2. Menjelaskan tipe data dasar dalam Python (numerik, string, list, tuple, dan dictionary).
3. Menggunakan variabel dan operasi dasar (aritmetika, logika, dan perbandingan) pada Python.
4. Menerapkan percabangan (*if*, *if-else*, *elif*) dan perulangan (*for*, *while*) dalam program sederhana.

## B. Alat dan Bahan

**Perangkat Lunak**:

* Sistem operasi Windows.
* Github
* Python 3.13.2
* IDE VS Code dengan extension jupyter notebook.

**Perangkat Keras**:

* Komputer/Laptop dengan spesifikasi memadai untuk menjalankan Python.

**Bahan Pendukung**:

* Modul/slide perkuliahan yang menjelaskan dasar pemrograman Python.
* Koneksi internet (opsional, untuk referensi tambahan).

## C. Dasar Teori Instalasi Python

* Pastikan Python 3.x telah terinstal.
* Verifikasi instalasi dengan menjalankan perintah python --version (di terminal/command prompt) untuk memastikan versi yang digunakan.

## Struktur Dasar Program Python

* File Python umumnya berekstensi .py.
* Sebuah program dapat terdiri dari sekumpulan pernyataan yang dijalankan secara berurutan.

## Tipe Data Dasar

* **Number**: Tipe data numerik seperti int, float, dan complex.
* **String**: Teks yang diapit tanda kutip (single atau double). Contoh: "Hello World".
* **List**: Sekumpulan data yang dapat diubah (mutable), ditulis di dalam tanda kurung siku [ ]. Contoh: [1, 2, 3].
* **Tuple**: Mirip list, namun bersifat tidak dapat diubah (immutable), ditulis dalam tanda kurung ( ). Contoh: (1, 2, 3).
* **Dictionary**: Sekumpulan pasangan *key-value*, ditulis dalam kurung kurawal { }.

Contoh: {"nama": "Ali", "umur": 20}.

## Variabel dan Operasi Dasar

* Penulisan variabel tidak memerlukan deklarasi tipe data eksplisit. Contoh:

python CopyEdit nama = "Budi" umur = 19

* Operasi dasar meliputi **aritmetika** (+, -, \*, /, //, %), **perbandingan** (==, !=, <, >, <=, >=), dan **logika** (and, or, not).

## Percabangan (Conditionals)

* *If*: Digunakan untuk mengeksekusi suatu blok kode jika kondisi bernilai benar.

python

CopyEdit if x > 0:

print("x adalah bilangan positif")

* *If-else*: Menyediakan blok alternatif jika kondisi salah.
* *Elif*: Menangani banyak kondisi.

## Perulangan (Loops)

* **For**: Digunakan untuk mengiterasi *sequence* (list, tuple, string, dsb.).

python CopyEdit

for i in range(5):

print(i)

* **While**: Berjalan selama kondisi bernilai benar. Berguna saat jumlah iterasi belum pasti.

## D. Langkah Praktikum Praktikum 01. Tipe data dasar

Pada praktikum ini, mahasiswa akan mempelajari tipe-tipe data dasar yang digunakan dalam pemrograman Python, mulai dari integer, float, dan complex untuk nilai numerik, hingga boolean, string, list, tuple, dictionary, serta set yang memiliki karakteristik serta fungsi tersendiri. Mahasiswa juga akan mempraktikkan cara melakukan konversi tipe data (misalnya dari string ke integer), sehingga dapat lebih memahami bagaimana Python secara dinamis menangani berbagai format data. Pemahaman terhadap tipe data dasar ini menjadi landasan penting dalam mengembangkan logika program yang efisien dan terstruktur, sekaligus mempersiapkan mahasiswa untuk mengimplementasikan konsep pemrograman berorientasi objek di tahap lanjutan.

Kode Praktikum:

|  |
| --- |
| 01: # 1. Number (int, float, complex)  02: # Integer (int)  03: angka\_int = 10  04: print("angka\_int =", angka\_int, "->", type(angka\_int)) 05:  06: # Float  07: angka\_float = 3.14  08: print("angka\_float =", angka\_float, "->", type(angka\_float)) 09:  10: # Complex  11: angka\_complex = 2 + 3j  12: print("angka\_complex =", angka\_complex, "->", type(angka\_complex)) 13:  14: # 2. Boolean  15: is\_active = True  16: print("is\_active =", is\_active, "->", type(is\_active)) 17:  18: # 3. String  19: teks = "Hello, Python!"  20: print("teks =", teks, "->", type(teks)) 21:  22: # 4. List  23: # List adalah tipe data terurut dan dapat diubah (mutable)  24: daftar\_angka = [1, 2, 3, 4, 5]  25: print("daftar\_angka =", daftar\_angka, "->", type(daftar\_angka)) 26:  27: # 5. Tuple  28: # Tuple adalah tipe data terurut tetapi tidak dapat diubah (immutable) |
| 29: koordinat = (10, 20)  30: print("koordinat =", koordinat, "->", type(koordinat)) 31:  32: # 6. Dictionary  33: # Dictionary menyimpan data dalam pasangan key-value  34: data\_mahasiswa = {  35: "nama": "Andi",  36: "nim": "A11.2022.12345",  37: "jurusan": "Teknik Informatika"  38: }  39: print("data\_mahasiswa =", data\_mahasiswa, "->", type(data\_mahasiswa)) 40:  41: # 7. Set  42: # Set adalah tipe data yang tidak terurut, unik (tiap elemen hanya muncul 1x)  43: himpunan\_angka = {1, 2, 3, 2, 1}  44: print("himpunan\_angka =", himpunan\_angka, "->", type(himpunan\_angka)) 45:  46: # 8. Contoh penggunaan konversi tipe data  47: nilai\_str = "100"  48: print("\nnilai\_str =", nilai\_str, "->", type(nilai\_str))  49: nilai\_int = int(nilai\_str) # konversi string ke integer  50: print("nilai\_int =", nilai\_int, "->", type(nilai\_int)) |

## Praktikum 02. Variabel dan Operasi dasar

Pada praktikum ini, mahasiswa akan mempelajari cara mendeklarasikan variabel tanpa harus menentukan tipe data secara eksplisit di Python. Beragam operasi dasar—seperti aritmetika (penjumlahan, pengurangan, perkalian, pembagian), perbandingan (==, !=, >, <, >=, <=), dan logika (and, or, not)—akan diterapkan untuk memanipulasi nilai yang disimpan dalam variabel. Dengan mempraktikkan berbagai operasi tersebut, mahasiswa dapat memahami bagaimana Python menangani ekspresi matematika dan logika untuk membentuk fondasi pengembangan aplikasi yang lebih kompleks.

Kode praktikum:

|  |
| --- |
| 01: # -\*- coding: utf-8 -\*- 02:  03: # 1. Pendeklarasian Variabel  04: nama = "Budi"  05: umur = 20  06: tinggi = 170.5 07: is\_student = True 08:  09: print("Nama =", nama)  10: print("Umur =", umur)  11: print("Tinggi =", tinggi, "cm") 12: print("Mahasiswa =", is\_student) 13:  14: # Python tidak memerlukan deklarasi tipe data secara eksplisit.  15: # Tipe data variabel akan menyesuaikan nilai yang diberikan.  16:  17: # 2. Operasi Aritmetika  18: a = 10 19: b = 3 20:  21: penjumlahan = a + b # Tambah  22: pengurangan = a - b # Kurang  23: perkalian = a \* b # Kali |
| 24: pembagian = a / b # Bagi (hasil float)  25: pembagian\_bulat = a // b # Bagi (bulat)  26: modulus = a % b # Sisa bagi  27: pangkat = a \*\* b # Pemangkatan 28:  29: print("\nOPERASI ARITMETIKA")  30: print("a =", a, ", b =", b)  31: print("Penjumlahan =", penjumlahan)  32: print("Pengurangan =", pengurangan)  33: print("Perkalian =", perkalian)  34: print("Pembagian =", pembagian)  35: print("Pembagian Bulat =", pembagian\_bulat)  36: print("Modulus =", modulus)  37: print("Pangkat =", pangkat) 38:  39: # 3. Operasi Perbandingan  40: # Menghasilkan nilai Boolean (True/False)  41: lebih\_besar = a > b  42: kurang\_dari = a < b  43: sama\_dengan = a == b  44: tidak\_sama = a != b  45: lebih\_besar\_sama = a >= b 46: kurang\_sama = a <= b 47:  48: print("\nOPERASI PERBANDINGAN")  49: print("a > b =", lebih\_besar)  50: print("a < b =", kurang\_dari)  51: print("a == b =", sama\_dengan)  52: print("a != b =", tidak\_sama)  53: print("a >= b =", lebih\_besar\_sama) 54: print("a <= b =", kurang\_sama) 55:  56: # 4. Operasi Logika 57: # and, or, not 58:  59: x = True 60: y = False 61:  62: logika\_and = x and y  63: logika\_or = x or y 64: logika\_not\_x = not x 65:  66: print("\nOPERASI LOGIKA")  67: print("x =", x, ", y =", y)  68: print("x and y =", logika\_and) 69: print("x or y =", logika\_or) 70: print("not x =", logika\_not\_x) 71:  72: # 5. Contoh penggunaan di dalam percabangan 73: if a > b and b > 0:  74: print("\nKondisi terpenuhi: a lebih besar dari b, dan b masih positif.") 75: else:  76: print("\nKondisi tidak terpenuhi atau b <= 0.") |

## Praktikum 03. Percabangan

Praktikum ini berfokus pada penerapan struktur kontrol aliran program berbasis kondisi, meliputi penggunaan if sederhana, if-else, if-elif-else, serta if bersarang (nested if). Mahasiswa akan mempelajari cara menggabungkan operasi logika (and, or, not) dengan percabangan untuk mengolah beragam skenario keputusan dalam program. Pendekatan ini memperkenalkan bagaimana jalannya program dapat berbeda tergantung pada kondisi yang diperiksa.

Kode Praktikum:

|  |
| --- |
| 01: # -\*- coding: utf-8 -\*- 02:  03: # 1. IF Sederhana  04: # Program hanya mengeksekusi blok jika kondisinya True 05:  06: nilai = 85  07: print("Contoh IF Sederhana:")  08: if nilai > 80:  09: print("Selamat! Anda lulus dengan nilai tinggi.\n") 10:  11:  12: # 2. IF-ELSE  13: # Jika kondisi True, eksekusi blok if; jika False, eksekusi blok else 14:  15: umur = 17  16: print("Contoh IF-ELSE:")  17: if umur >= 18:  18: print("Anda sudah cukup umur untuk mendapatkan SIM.") 19: else:  20: print("Anda belum cukup umur untuk mendapatkan SIM.\n") 21:  22:  23: # 3. IF-ELIF-ELSE  24: # Menangani banyak kondisi secara berurutan.  25: # Jika ada kondisi yang terpenuhi, blok yang bersangkutan dieksekusi, 26: # lalu program melewati blok kondisi setelahnya.  27:  28: hari = "Rabu"  29: print("Contoh IF-ELIF-ELSE:")  30: if hari == "Senin":  31: print("Hari Senin - Saatnya kembali bekerja!")  32: elif hari == "Selasa":  33: print("Hari Selasa - Jadwal rapat mingguan.")  34: elif hari == "Rabu":  35: print("Hari Rabu - Ada diskon di beberapa toko.") 36: else:  37: print("Hari lainnya - Atur jadwalmu dengan baik.\n") 38:  39:  40: # 4. IF Bersarang (Nested IF)  41: # Kondisi di dalam kondisi, biasa digunakan jika kita perlu 42: # memeriksa sub-kondisi setelah kondisi pertama terpenuhi. 43:  44: suhu = 35  45: print("Contoh IF Bersarang (Nested IF):")  46: if suhu > 30:  47: print("Cuaca cukup panas.")  48: if suhu > 40:  49: print("Bahkan sangat terik! Disarankan banyak minum air.") 50: else: |
| 51: print("Masih relatif normal, tapi tetap jaga kesehatan.") 52: else:  53: print("Cuaca sepertinya cukup sejuk.\n") 54: 55:  56: # 5. Menggabungkan Percabangan dengan Operasi Logika 57: # Memeriksa beberapa kondisi sekaligus dengan and, or, not 58:  59: nilai\_teori = 75  60: nilai\_praktik = 80  61: print("Contoh IF dengan Operasi Logika AND/OR:")  62: if nilai\_teori >= 70 and nilai\_praktik >= 70:  63: print("Anda lulus karena nilai teori dan praktik memadai.")  64: elif nilai\_teori < 70 and nilai\_praktik < 70:  65: print("Anda perlu meningkatkan nilai teori dan praktik.")  66: elif nilai\_teori < 70:  67: print("Anda perlu meningkatkan nilai teori.") 68: else:  69: print("Anda perlu meningkatkan nilai praktik.\n") 70: 71:  72: # 6. Penggunaan If Ternary (atau Conditional Expression)  73: # Bentuk ringkas: <hasil\_if\_true> if <kondisi> else <hasil\_if\_false> 74:  75: angka = -5  76: print("Contoh If Ternary (Conditional Expression):")  77: status = "Positif" if angka > 0 else "Negatif atau Nol"  78: print("Angka =", angka, "=>", status) |

## Praktikum 04. Perulangan

Pada praktikum ini, mahasiswa akan mendalami cara mengulang eksekusi suatu blok kode berkali-kali menggunakan struktur perulangan for dan while. Beragam fitur lanjutan, seperti break, continue, serta loop bersarang (nested loop), turut diulas untuk memberikan fleksibilitas lebih dalam mengelola logika program. Mahasiswa juga akan mempelajari penggunaan else pada loop yang merupakan fitur unik Python, serta bagaimana memanfaatkan pass sebagai placeholder.

Kode Praktikum:

|  |
| --- |
| 01: # -\*- coding: utf-8 -\*- 02:  03: # 1. FOR Loop dengan range()  04: print("1) FOR loop dengan range()")  05: for i in range(5):  06: print("Perulangan ke-", i)  07: # range(5) menghasilkan nilai 0, 1, 2, 3, dan 4  08: # Sehingga perulangan akan berjalan sebanyak 5 kali 09:  10: print() # pemisah output 11:  12: # 2. FOR Loop untuk mengiterasi List  13: print("2) FOR loop mengiterasi list")  14: buah = ["apel", "mangga", "jeruk", "pisang"]  15: for item in buah:  16: print("Buah:", item)  17: # Loop akan mengeksekusi setiap elemen dalam list "buah" 18:  19: print() |

20:

21: # 3. WHILE Loop

22: print("3) WHILE loop sederhana") 23: count = 0

24: while count < 5:

25: print("count =", count)

26: count += 1 # increment

27: # Perulangan while terus dijalankan selama kondisi (count < 5) bernilai

True 28: 29: print() 30:

31: # 4. BREAK pada Loop

32: print("4) BREAK di dalam loop") 33: for i in range(10):

34: if i == 3:

35: print("Loop dihentikan pada i =", i)

36: break # mengakhiri loop saat i = 3

37: print("i =", i)

38: # Keyword break langsung menghentikan keseluruhan perulangan 39: 40: print() 41:

42: # 5. CONTINUE pada Loop

43: print("5) CONTINUE di dalam loop") 44: for i in range(5):

45: if i == 2:

46: print("Lewati i =", i, "dengan continue")

47: continue # melewati iterasi saat ini dan lanjut ke iterasi berikutnya

48: print("i =", i)

49: # Saat i = 2, baris print("i =", i) tidak akan dieksekusi 50: 51: print() 52:

53: # 6. NESTED Loop (Loop Bersarang)

54: print("6) NESTED loop")

55: for i in range(3): # Loop luar (outer loop)

56: for j in range(2): # Loop dalam (inner loop)

57: print(f"i={i}, j={j}")

58: # Pada setiap iterasi i, loop j akan berjalan dari 0 sampai 1 59: 60: print() 61:

62: # 7. Memanfaatkan ELSE pada Loop

63: print("7) ELSE pada loop for/while")

64: # Python memiliki fitur unik: blok else pada loop

65: # Blok else akan dieksekusi jika loop selesai tanpa di-break.

66:

67: for x in range(3): 68: print("x =", x) 69: else:

70: print("Loop for telah selesai tanpa break.\n") 71:

72: y = 0

73: while y < 3:

74: print("y =", y) 75: y += 1 76: else:

77: print("Loop while telah selesai tanpa break.\n") 78:

|  |
| --- |
| 79: # 8. PASS sebagai placeholder  80: print("8) PASS (placeholder)") 81: for i in range(3):  82: if i == 1:  83: pass # pass tidak melakukan apa-apa, digunakan sebagai placeholder  84: print("i =", i) 85: |

## E. Hasil Praktikum

Lengkapi hasil tabel praktikum berikut:

|  |  |  |
| --- | --- | --- |
| **No.** | **Nama Praktikum** | **Hasil Praktikum** |
| 1 | Tipe data dasar |  |
| 2 | Variable dan operasi dasar |  |
| 3 | Percabangan |  |
| 4 | Perulangan |  |

## F. Penugasan

1. Kumpulkan Laporan Praktikum dari jobsheet ini dalam bentuk Microsoft word sesuai dengan format jobsheet praktikum dan dikumpulkan di web elnino polines. (JANGAN DALAM BENTUK PDF)
2. Kumpulkan luaran kode praktikum dalam bentuk ipynb yang sudah diunggah pada akun github masing-masing. Lampirkan tautan github yang sudah di unggah melalui laman LMS elnino.

**Link Github:**

https://github.com/AbimanyuGilar/Jobsheet-1---Object-Oriented-Programming-Class.git

1. **Buat Program 1**: Buat program untuk menghitung BMI (Body Mass Index) dengan rumus:

𝑏𝑒𝑟𝑎𝑡(𝑘𝑔) 𝐵𝑀𝐼 = 𝑡𝑖𝑛𝑔𝑔𝑖(𝑚)2

* 1. Input: berat (kg) dan tinggi (cm).
  2. Output: nilai BMI dan kategori (Underweight, Normal, Overweight).

**Hasil:**

https://github.com/AbimanyuGilar/Jobsheet-1---Object-Oriented-Programming-Class/blob/main/body%20mass%20index.ipynb

1. **Buat Program 2**: Buat program yang mengecek apakah suatu bilangan genap, ganjil, atau prima.

**Hasil:**

https://github.com/AbimanyuGilar/Jobsheet-1---Object-Oriented-Programming-Class/blob/main/odd-even-prime.ipynb

## G. Kesimpulan

Secara keseluruhan, Python menawarkan sintaks yang sederhana dan mudah dibaca sehingga cocok untuk mempelajari konsep pemrograman dasar. Melalui tipe data dan struktur data seperti list, tuple, serta dictionary, pengelolaan informasi menjadi lebih mudah dan terorganisir. Variabel di Python tidak memerlukan deklarasi tipe data secara eksplisit, sehingga memudahkan proses penulisan program. Sementara itu, pengaturan aliran program dengan percabangan (if-else, elif) dan perulangan (for, while) memberikan fleksibilitas dan efisiensi dalam menangani beragam kasus.

## H. Daftar Pustaka

Lutz, M. (2013). Learning Python (5th Edition). O’Reilly Media.

Downey, A. (2015). Think Python: How to Think Like a Computer Scientist (2nd Edition). Green Tea Press.

Python Software Foundation. Dokumentasi Resmi Python.