



# **ALGORITMA DAN STRUKTUR DATA**

**INF1083**

## **LAPORAN PRAKTIKUM 1 : Data Numerik**

Oleh :

*Akhmad Qasim*

*2211102441237*

Teknik Informatika  
Fakultas Sains & Teknologi  
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

# Laporan Praktikum 1:

# Data Numerik

---

## Pokok Bahasan:

- ❖ Review *Python 3*
- ❖ Tipe Data Numerik

## Tujuan Pembelajaran:

- ✓ Meninjau kembali Bahasa Pemrograman *Python 3*.
- ✓ Mengenal tipe data numerik.
- ✓ Menggunakan *syntax* numerik *Python 3*.

## Tugas Pendahuluan:

1. Jelaskan parameter yang terdapat di *syntax* dibawah ini:

- 1.1. `int(x, base)`

Sintaks `int(x, base)` adalah fungsi Python bawaan yang mengubah angka atau string dari angka menjadi bilangan bulat. Argumen pertama `x` adalah angka atau string yang ingin diubah menjadi bilangan bulat. Basis argumen kedua (opsional) adalah basis representasi angka dalam `x` dan nilai standarnya adalah 10. Jika basis ditentukan, `x` harus berupa representasi string dari angka dalam basis yang diberikan. Misalnya, `int("1010", 2)` akan mengubah string biner "1010" menjadi bilangan bulat desimal 10, dan `int("10", 16)` akan mengubah string heksadesimal "10" menjadi bilangan bulat desimal 16.

- 1.2. `float(x)`

Sintaks `float(x)` adalah fungsi Python bawaan yang mengubah representasi angka atau string dari angka menjadi angka floating-point. Argumen `x` bisa berupa bilangan bulat, representasi string dari angka titik-mengambang, atau tipe lain yang bisa dikonversi menjadi pelampung. Jika argumennya bukan angka atau string yang dapat diubah menjadi float, pengecualian `ValueError` akan dimunculkan. Misalnya, `float(10)` akan mengonversi bilangan bulat 10 menjadi bilangan titik-mengambang 10.0, dan `float("3.14")` akan mengubah string "3.14" menjadi bilangan titik-mengambang 3.14.

- 1.3. `complex(real[,imag])`

Kompleks sintaksis(`nyata[, imag]`) adalah fungsi Python bawaan yang membuat bilangan kompleks dari bagian nyata dan bagian imajiner opsional. Argumen `real` adalah bilangan (integer atau floating-point) yang mewakili bagian real dari

bilangan kompleks. Argumen opsional *imag* (default 0) adalah angka yang mewakili bagian imajiner dari bilangan kompleks. Misalnya, `kompleks(1, 2)` membuat bilangan kompleks dengan bagian riil 1 dan bagian imajiner 2, yang dapat direpresentasikan sebagai  $1 + 2j$ , dan `kompleks(3)` membuat bilangan kompleks dengan bagian riil 3 dan bagian imajiner 0, yang dapat direpresentasikan sebagai  $3 + 0j$ .

2. Sebutkan dan jelaskan metode lain untuk merepresentasikan 3 tipe data numerik (selain yang terdapat pada contoh diatas)!

Di Python, selain tipe data `int`, `float`, dan `complex`, ada beberapa cara lain untuk merepresentasikan data numerik, yaitu:

- **Bool**: Tipe data boolean yang dapat mengambil nilai `True` atau `False`.
- **Desimal**: Tipe data yang menyediakan representasi angka desimal yang akurat. Digunakan dalam kalkulasi finansial atau moneter yang membutuhkan tingkat presisi yang tinggi.
- **Pecahan (Fraction)**: Tipe data yang mewakili bilangan rasional sebagai pecahan. Digunakan untuk merepresentasikan bilangan rasional yang tidak dapat direpresentasikan dengan tepat sebagai bilangan `float`.
- **BigInt**: Tipe data yang menyediakan sembarang bilangan bulat presisi, yang berarti bahwa ukuran nilai bilangan bulat hanya akan dibatasi oleh memori komputer yang tersedia.
- **float16, float32, float64**: Bilangan Presisi selain dari bilangan floating-point.
- **int8, int16, int32, int64**: Bilangan Presisi selain integer.
- **uint8, uint16, uint32, uint64**: Bilangan Presisi dari *unsigned* integer.

Tipe data ini tersedia di berbagai *library* seperti `numpy`, `decimal`, dan `fractions`. Pilihan tipe data bergantung pada kebutuhan khusus dari masalah yang ingin dipecahkan, seperti bilangan presisi yang diperlukan, ukuran data, dan apakah data tersebut *unsigned* atau tidak.

### Percobaan & Latihan:

1.

1.1.  $Ha = x + y$

```
4 Ha = x + y
5 print("Ha =", Ha)
```

$x$  dan  $y$  masing-masing diberi nilai 12 dan 37. Kemudian,  $Ha = x + y$  menjumlahkan nilai  $x$  dan  $y$  dan memberikan hasilnya ke variabel  $Ha$ . Jadi,  $Ha = 12 + 37 = 49$ . Artinya nilai  $Ha$  memiliki nilai 49.

### 1.2. $H_b = H_a + 10.5$

```
7 Hb = Ha + 10.5
8 print("Hb =", Hb)

Hb = 59.5
```

Variable  $H_b = H_a + 10.5$  mengambil nilai dari variabel  $H_a$  (yang memiliki nilai 49) dan menambahkan 10.5 ke dalamnya, lalu menugaskan hasilnya ke variabel  $H_b$ . Di sini,  $H_a$  memiliki nilai 49, yang merupakan bilangan bulat. Operator penjumlahan  $+$  menjumlahkan nilai  $H_a$  dan 10,5, yang merupakan bilangan floating. Hasil dari operasi ini akan berupa bilangan floating-point dan disimpan dalam variabel  $H_b$ . Jadi,  $H_b = 49 + 10,5 = 59,5$ . Artinya variable  $H_b$  akan bernilai 59,5.

### 1.3. $H_c = \text{int}(H_b)$

```
10 Hc = int(Hb)
11 print("Hc =", Hc)

Hc = 59
```

Variable  $H_b = 59.5$  merupakan bilangan floating-point. Variable  $H_c = \text{int}(H_b)$  mengubah bilangan floating-point menjadi bilangan bulat menggunakan fungsi  $\text{int}()$ , dan memberikan hasilnya ke variabel  $H_c$ . Di Python, fungsi  $\text{int}()$  dibulatkan ke bawah ke bilangan bulat terdekat jika argumennya adalah angka floating. Jadi,  $H_c = \text{int}(59.5) = 59$ . Artinya variable  $H_c$  akan bernilai 59.

### 1.4. $H_d = H_a / H_b$

```
13 Hd = Ha / Hb
14 print("Hd =", Hd)

Hd = 0.8235294117647058
```

Variable  $H_a$  memiliki nilai 49, yang merupakan bilangan bulat. Variable  $H_b$  memiliki nilai 59,5, yang merupakan bilangan floating-point. Kode  $H_d = H_a / H_b$  membagi nilai  $H_a$  dengan nilai  $H_b$  dan menugaskan hasilnya ke variabel  $H_d$ . Karena  $H_a$  dan  $H_b$  adalah nilai numerik, maka nilainya akan diubah menjadi floating dan mengembalikan hasil floating. Jadi,  $H_d = 49,0 / 59,5 = 0.8235294117647058$ . Ini berarti variable  $H_d$  akan menjadi 0.8235294117647058.

### 1.5. $\text{isinstance}(H_d, \text{int})$

```
16 print(isinstance(Hd, int))

False
```

Kode  $\text{isinstance}(H_d, \text{int})$  di Python memeriksa apakah nilai yang disimpan dalam variabel  $H_d$  adalah turunan dari tipe  $\text{int}$  (integer). Fungsi  $\text{isinstance}()$  mengambil

dua argumen: argumen pertama adalah nilai yang ingin kita periksa tipenya, dan argumen kedua adalah tipe yang ingin kita periksa. Dalam hal ini, Hd adalah nilai yang ingin kita periksa, dan int adalah tipe yang ingin kita periksa. Jika Hd bilangan bulat, maka `isinstance(Hd, int)` akan mengembalikan True. Jika tidak, jika Hd bukan bilangan bulat, maka `isinstance(Hd, int)` akan mengembalikan False. Karena Hd memiliki nilai 0.8235294117647058, maka akan mengeluarkan output false.

#### 1.6. `isinstance(Hd, float)`

```
18 print(isinstance(Hd, float))
```

```
True
```

Kode `isinstance(Hd, float)` di Python memeriksa apakah nilai yang disimpan dalam variabel Hd adalah turunan dari tipe float (floating-point number). Fungsi `isinstance()` mengambil dua argumen: argumen pertama adalah nilai yang ingin kita periksa tipenya, dan argumen kedua adalah tipe yang ingin kita periksa. Dalam hal ini, Hd adalah nilai yang ingin kita periksa, dan float adalah tipe yang ingin kita periksa. Jika Hd adalah bilangan floating-point, maka `isinstance(Hd, float)` akan mengembalikan True. Jika tidak, jika Hd bukan angka floating, maka `isinstance(Hd, float)` akan mengembalikan False. Karena Hd memiliki nilai 0.8235294117647058, maka akan mengeluarkan output true.

```
20 if isinstance(Hd, int):
21     print("Hd adalah bilangan integer")
22 else:
23     print("Hd adalah bilangan float")
```

#### 1.7.

```
Hd adalah bilangan float
```

Kode dalam Python ini memeriksa jenis nilai yang disimpan dalam variabel Hd. Variable Hd memiliki nilai 0,8235294117647058, yang merupakan bilangan floating-point. Kode menggunakan fungsi `isinstance()` untuk memeriksa apakah Hd merupakan turunan dari tipe int (integer). Jika Hd adalah bilangan bulat, maka `isinstance(Hd, int)` akan mengembalikan True dan kode di dalam blok pertama akan dieksekusi. Jika tidak, maka `isinstance(Hd, int)` akan mengembalikan False dan kode di dalam blok kedua (blok else) akan dieksekusi. Dalam hal ini, kode akan menampilkan "Hd adalah bilangan float".

## 2.

#### 2.1. `(999).bit_length()`

```
1 print((999).bit_length())
```

```
10
```

Eksresi `(999).bit_length()` dalam Python mengembalikan jumlah bit yang

diperlukan untuk mewakili bilangan bulat dalam format biner. Metode `bit_length()` adalah metode bawaan di Python yang mengembalikan jumlah bit yang diperlukan untuk mewakili bilangan bulat dalam format biner, tidak termasuk angka nol di depan. Metode ini dipanggil pada objek bilangan bulat dan mengembalikan nilai bilangan bulat. Dalam hal ini, 999 adalah bilangan bulat, dan ekspresi `(999).bit_length()` memanggil metode `bit_length()` pada bilangan bulat ini. 999 direpresentasikan dalam biner sebagai `0b1111100111`, yang panjangnya 10 bit. Jadi, ekspresi `(999).bit_length()` akan bernilai 10.

## 2.2. `(998).bit_length()`

```
3 print((998).bit_length())  
10
```

Ekspresi `(998).bit_length()` dalam Python mengembalikan jumlah bit yang diperlukan untuk mewakili bilangan bulat dalam format biner. Metode `bit_length()` adalah metode bawaan di Python yang mengembalikan jumlah bit yang diperlukan untuk mewakili bilangan bulat dalam format biner, tidak termasuk angka nol di depan. Metode ini dipanggil pada objek bilangan bulat dan mengembalikan nilai bilangan bulat. Dalam hal ini, 998 adalah bilangan bulat, dan ekspresi `(998).bit_length()` memanggil metode `bit_length()` pada bilangan bulat ini. 998 direpresentasikan dalam biner sebagai `0b1111100110`, yang panjangnya 10 bit. Jadi, ekspresi `(998).bit_length()` akan bernilai 10.

## 2.3. `(99).bit_length()`

```
5 print((99).bit_length())  
7
```

Ekspresi `(99).bit_length()` dalam Python mengembalikan jumlah bit yang diperlukan untuk mewakili bilangan bulat dalam format biner. Metode `bit_length()` adalah metode bawaan di Python yang mengembalikan jumlah bit yang diperlukan untuk mewakili bilangan bulat dalam format biner, tidak termasuk angka nol di depan. Metode ini dipanggil pada objek bilangan bulat dan mengembalikan nilai bilangan bulat. Dalam hal ini, 99 adalah bilangan bulat, dan ekspresi `(99).bit_length()` memanggil metode `bit_length()` pada bilangan bulat ini. 99 direpresentasikan dalam biner sebagai `0b1100011`, yang panjangnya 7 bit. Jadi, ekspresi `(99).bit_length()` akan bernilai 7.

## 2.4. `(11).bit_length()`

```
7 print((11).bit_length())  
4
```

Ekspresi `(11).bit_length()` dalam Python mengembalikan jumlah bit yang diperlukan untuk mewakili bilangan bulat dalam format biner. Metode

`bit_length()` adalah metode bawaan di Python yang mengembalikan jumlah bit yang diperlukan untuk mewakili bilangan bulat dalam format biner, tidak termasuk angka nol di depan. Metode ini dipanggil pada objek bilangan bulat dan mengembalikan nilai bilangan bulat. Dalam hal ini, 11 adalah bilangan bulat, dan ekspresi `(11).bit_length()` memanggil metode `bit_length()` pada bilangan bulat ini. 11 direpresentasikan dalam biner sebagai 0b1011, yang panjangnya 4 bit. Jadi, ekspresi `(11).bit_length()` akan bernilai 4.

### 3.

#### 3.1. `bin(i)`

```
3 print(bin(i))
0b1000101011
```

Ekspresi `bin(i)` dalam Python mengembalikan string hasil biner dari bilangan bulat yang diberikan. Fungsi mengambil integer sebagai argumen dan mengembalikan string dalam format 0b diikuti dengan representasi biner dari integer. Dalam hal ini, `i` adalah variabel yang menyimpan nilai integer 555. Ekspresi `bin(i)` memanggil fungsi `bin()` pada `i`, yang mengembalikan string 0b1000100011, yang merupakan representasi biner dari 555.

#### 3.2. `hex(i)`

```
5 print(hex(i))
0x22b
```

Fungsi bawaan `hex()` di Python mengembalikan representasi string hasil heksadesimal dari bilangan bulat pada parameter. Fungsi mengambil bilangan bulat sebagai argumen dan mengembalikan string dalam format 0x diikuti dengan representasi bilangan bulat heksadesimal. Dalam hal ini, `i` adalah variabel yang menyimpan nilai integer 555. Ekspresi `hex(i)` memanggil fungsi `hex()` pada `i`, yang mengembalikan string 0x22b, yang merupakan representasi heksadesimal dari 555.

#### 3.3. `oct(i)`

```
7 print(oct(i))
0o1053
```

Fungsi built-in `oct()` di Python mengembalikan representasi string hasil oktal dari parameter yang diberikan. Fungsi mengambil integer sebagai argumen dan mengembalikan string dalam format 0o diikuti dengan representasi oktal dari integer. Dalam hal ini, `i` adalah variabel yang menyimpan nilai integer 555. Ekspresi `oct(i)` memanggil fungsi `oct()` pada `i`, yang mengembalikan string 0o1033, yang merupakan representasi oktal dari 555.

**Kesimpulan:**

Dalam Python, ada beberapa tipe data numerik yang dapat merepresentasikan angka, antara lain bilangan bulat (int), bilangan titik-mengambang (float), dan bilangan kompleks (kompleks).

- int mewakili bilangan bulat, yaitu bilangan bulat tanpa komponen pecahan. Misalnya, 10, -3, 0.
- float mewakili bilangan real dengan komponen pecahan. Misalnya, 3,14, -0,01, 1,0.
- kompleks mewakili bilangan kompleks, yang terdiri dari bagian nyata dan bagian imajiner. Bilangan kompleks ditulis dalam bentuk  $a + bj$ , dimana  $a$  adalah bagian real dan  $b$  adalah bagian imajiner. Misalnya,  $3 + 4j$ ,  $-1 + 0j$ .