

INTRODUCTION TO COMPUTATIONAL MODELS WITH PHYTON

Tugas Mata Kuliah SK5003 Pemrograman dalam Sains
Summary Chapter I dan II

Mohammad Rizka Fadhli (Ikang)
20921004@mahasiswa.itb.ac.id

03 September 2021

Contents

1 CHAPTER I	5
1.1 <i>Introduction</i>	5
1.2 <i>Computer Problem Solving</i>	5
1.3 <i>Elementary Concepts</i>	6
1.4 <i>Developing Computational Models</i>	8
1.5 Contoh Kasus	9
1.5.1 <i>Converter Celcius ke Farenheit</i>	10
1.5.2 Perhitungan Luas dan Keliling Lingkaran	11
1.6 Kategori <i>Computational Models</i>	12
1.7 <i>Software Life Cycle</i>	12
1.8 <i>Design Modular</i>	13
1.9 Bahasa Pemrograman	14
1.9.1 <i>Interpreter Python</i>	15
1.9.2 <i>Compilers</i>	16
1.10 Presisi, Akurasi, dan Galat	16
1.10.1 Jenis-Jenis Galat	17
2 CHAPTER II	18
2.1 <i>Introduction</i>	18
2.2 <i>Simple Operations</i>	19
2.2.1 <i>Mathematical Operations</i>	19
2.2.2 <i>Advance Mathematical Operations</i>	20
2.2.3 <i>Scientific Notation</i>	21
2.3 <i>Programs</i>	21
2.3.1 Pendefinisian Data	21
2.3.2 <i>Assignment Variable</i>	22
2.3.3 Perintah Dasar <i>Input</i> dan <i>Output</i>	22
2.3.4 Contoh Membuat <i>Script</i>	23
2.4 Contoh Program Sederhana	24
2.4.1 <i>Converter Celcius ke Farenheit</i>	24
2.4.2 Penghitung <i>Euclidean Distance</i>	25

2.5 Struktur Umum Program Python	27
2.6 Pendefinisian <i>Function</i>	28

List of Figures

1	Proses Problem Solving	5
2	Flow Melakukan Research	6
3	Elemen Computational Thinking	7
4	Integrasi Sains Komputasi	8
5	Development of Computational Models	8
6	Model development and Abstract Levels	9
7	Framework Problem - Output	10
8	Waterfall Model	12
9	Fase Lain dalam Sorftware Life Cycle	13
10	Konsep Decomposition	14
11	Contoh Python yang Dijalankan di Command Prompt	15
12	Contoh Python yang Dijalankan di Spyder IDE	15
13	Presisi, Akurasi, dan Galat	16
14	Cara Scripting di Python	18
15	Python CLI via Terminal	18
16	Mathematical Operations	19
17	Advance Mathematical Operations	20
18	Run .py di Terminal Linux	23
19	Run Converter Celcius Farenheit di Terminal Linux	24
20	Ilustrasi Perhitungan Jarak	25
21	Run Program Jarak di Terminal Linux	26
22	Struktur Program Python	27
23	Run Program Celcius Farenheit dalam Function	29

1 CHAPTER I

Problem Solving dan Computing

1.1 *Introduction*

Dalam hidupnya, manusia pasti akan berhadapan dengan masalah. Tidak sedikit permasalahan yang membutuhkan penyelesaian secara komputasi sedangkan otak manusia memiliki keterbatasan dalam melakukan komputasi yang rumit. Oleh karena itu, mereka menciptakan suatu *tools* yang dapat membantu mereka menyelesaikan masalah tersebut, salah satunya adalah *computer solution* seperti *computer program*.

Computer program berisi data dan sekumpulan perintah berupa algoritma untuk melakukan *well-defined tasks*.

Computer program pada dasarnya menjalankan *computational model*, yakni implementasi dari model matematika yang diformulasikan untuk mencari solusi permasalahan. Biasanya *computational model* membutuhkan *resource* komputasi yang tinggi.

1.2 *Computer Problem Solving*

Problem solving adalah proses membuat solusi komputer dari masalah nyata. Hal yang paling menantang dari proses ini adalah memilih metode yang tepat untuk menyelesaikan permasalahan tersebut.

Sebagai mana yang kita tahu ada istilah *no free lunch*, artinya metode penyelesaian setiap permasalahan adalah *unique*¹.

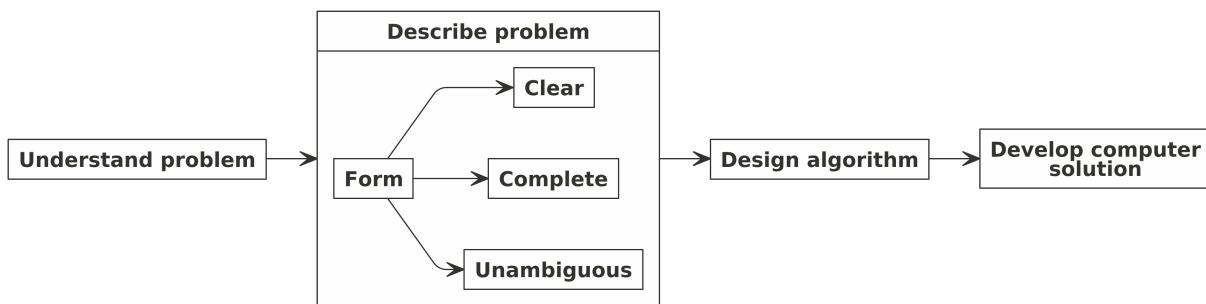


Figure 1: Proses Problem Solving

Algoritma adalah sekumpulan baris perintah untuk melakukan suatu proses komputasi dari suatu *data* input sehingga menghasilkan data *output*. Algoritma bisa dituliskan dalam bentuk *pseudo-code* atau *flowcharts*.

¹ *No free lunch*: https://ikanx101.com/blog/no_free-lunch/

Program komputer ditulis berdasarkan algoritma yang telah dibuat sebelumnya dengan bahasa pemrograman tertentu.

1.3 Elementary Concepts

Model adalah representasi dari sistem atau masalah. Bisa jadi model hanya berisi bagian tertentu saja dari sistem atau masalah. Suatu model bisa juga lebih sederhana dari masalah sebenarnya asalkan masih relevan dalam beberapa aspek.

Bagaimana cara kita membuat model tersebut?

Sebenarnya dalam membuat suatu model dari permasalahan yang ada, prosesnya mirip dengan melakukan penelitian pada umumnya². Setidaknya ada beberapa tahapan sebagai berikut:

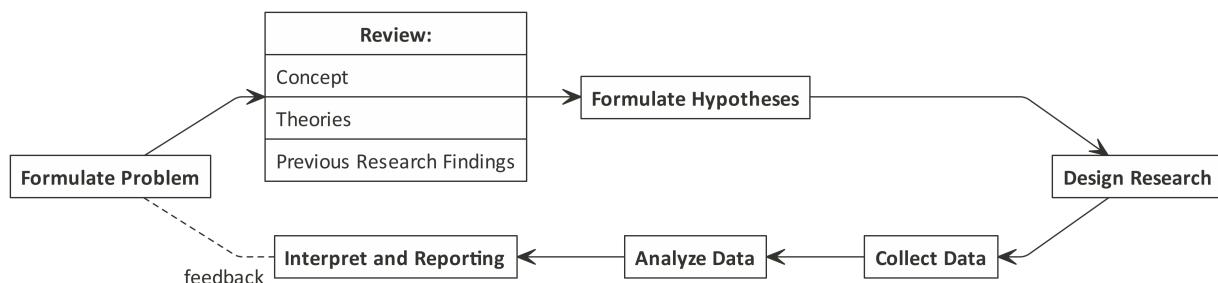


Figure 2: Flow Melakukan Research

Salah satu tahap kritis yang paling penting adalah tahapan pertama, yakni **formulasi masalah**.

Salah dalam memformulasikan masalah akan menimbulkan masalah bagi keseluruhan penelitian.

Di dalam *computational modelling*, formulasi masalah biasa disebut dengan *abstraction*. *Abstraction* yang baik harus bisa mendapatkan elemen esensial dari permasalahan atau sistem.

Setelah kita berhasil melakukan *abstraction*, kita harus mulai berpikir bagaimana memformulasikan masalah tersebut dan mulai mencari solusi komputasinya. Proses ini disebut dengan *computational thinking*.

²Flow melakuan penelitian: <https://ikanx101.com/blog/tujuan/>

Ada **empat pilar utama** dalam *computational thinking*:

1. Dekomposisi.
 - Memecah masalah besar ke masalah-masalah yang lebih kecil sehingga lebih bisa di-manage.
2. *Pattern recognition*.
 - Menganalisa dan melihat apakah ada pola atau pengulangan.
3. *Algorithm design*
 - Menuliskan langkah-langkah dalam bentuk formal.
4. *Abstraction*
 - Memisahkan mana yang *important*, mana yang *less important*.

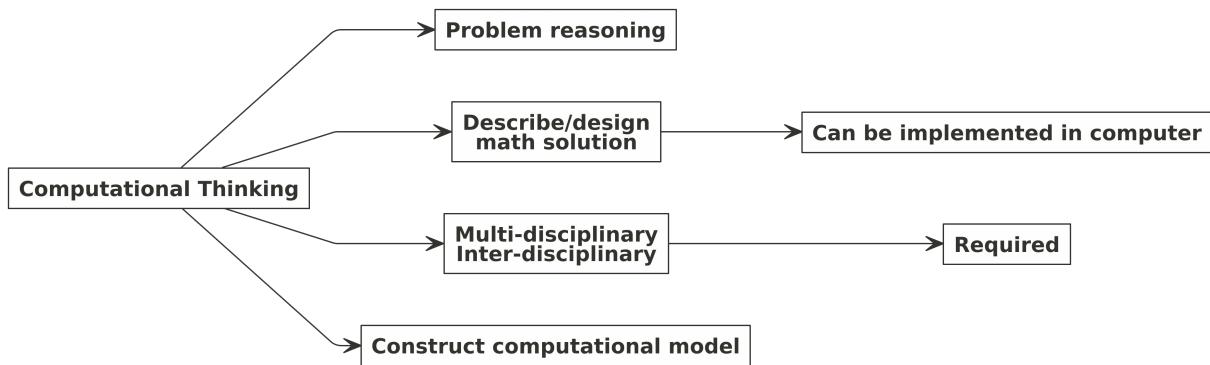


Figure 3: Elemen Computational Thinking

Setelah kita memformulasikan masalah dan membuat modelnya, untuk mendapatkan solusi kita bisa menempuh cara matematis. Sebagai contoh kita bisa mencari solusi dari model *predator-prey*³ dengan menurunkan sendiri persamaan diferensial yang ada. Namun ada kalanya kita tidak bisa melakukan hal tersebut sehingga perlu ada penyelesaian dengan pendekatan numerik. Contoh sederhana adalah penggunaan Metode Newton yang memanfaatkan *Taylor's Series* dan iterasi untuk mendapatkan akar suatu persamaan⁴.

Dari sinilah muncul istilah *computational science* (sains komputasi).

Sains komputasi menggabungkan komsep dan prinsip dari matematika dan *computer science* untuk diaplikasikan di bidang sains lain atau *engineering*.

³Contoh model persamaan diferensial yang terkenal: https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations

⁴Metode Newton: https://ikanx101.com/blog/newton_method/

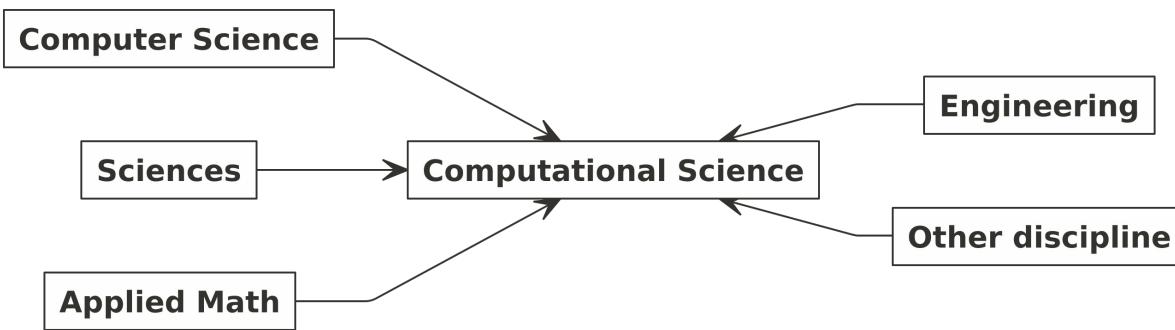


Figure 4: Integrasi Sains Komputasi

1.4 *Developing Computational Models*

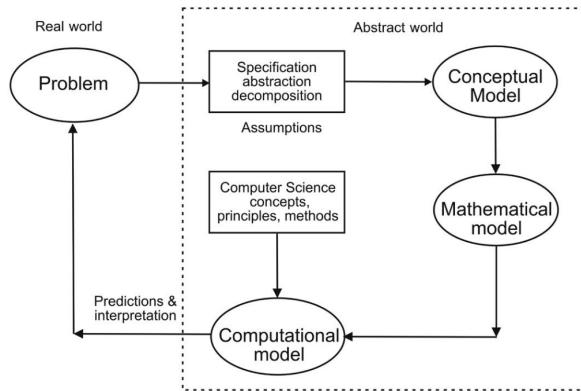


Figure 5: Development of Computational Models

Computational model dibangun secara iteratif. Maksudnya pada saat kita membuat modelnya, perlu ada proses *fine tuning* (penyempurnaan) berkelanjutan agar model tersebut bisa merepresentasikan masalah atau sistem dengan baik.

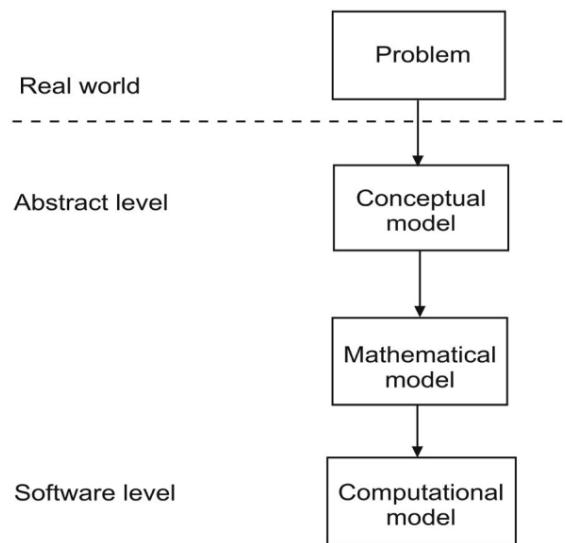


Figure 6: Model development and Abstract Levels

1.5 Contoh Kasus

Pada bagian ini saya sarikan contoh kasus yang ada pada buku, yakni:

1. *Converter Celcius ke Farenheit.*
2. Perhitungan luas dan keliling lingkaran.

Untuk memudahkan proses *summary*, saya akan gunakan *framework* sebagai berikut:

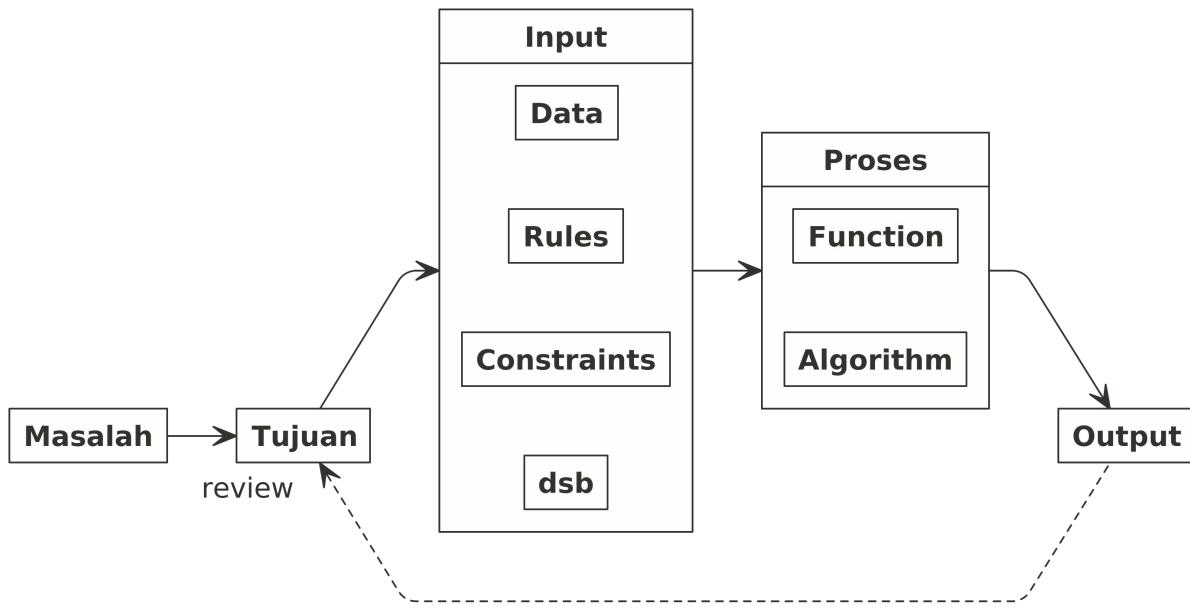


Figure 7: Framework Problem - Output

1.5.1 Converter Celcius ke Farenheit

Initial Problem Statement Turis Eropa yang datang ke Amerika perlu mengetahui temperatur di kotanya berkunjung. Mereka butuh informasi temperatur dalam Farenheit sementara informasi temperatur yang ada dalam Celcius.

Dari *statement* di atas, kita masukkan ke dalam *framework* sebagai berikut:

- **Masalah**
 - Turis Eropa biasa menggunakan unit Farenheit sedangkan di Amerika menggunakan unit Celcius. Mereka perlu mengetahui berapa suhu udara di kotanya berkunjung (dalam Farenheit) menggunakan informasi temperatur dalam Celcius.
- **Tujuan**
 - Melakukan konversi temperatur Farenheit ke Celcius.
- **Input**
 - Informasi yang dimiliki saat ini adalah temperatur udara dalam Celcius, misal dinotasikan sebagai C .
- **Proses**
 - Untuk melakukan konversi, saya akan membuat fungsi berikut ini: $F = \frac{9}{5}C + 32$.
- **Output**

- Hasil akhir yang diharapkan adalah temperatur udara dalam unit Farenheit, yakni F .

Secara *simple*, algoritmanya adalah sebagai berikut:

```
INPUT C
COMPUTE F = (9/5)*C + 32
OUTPUT F
```

Key Take Points Kasus ini adalah salah satu contoh kasus yang sangat *clear* dalam hal komputasi (proses perhitungan) sehingga kita sudah tidak perlu lagi melakukan *fine tuning* terhadap algoritma yang ada.

Untuk melakukan proses perhitungan, saya menggunakan fungsi matematis hubungan antara F dan C . Fungsi matematis inilah yang sudah kita pelajari di kalkulus.

1.5.2 Perhitungan Luas dan Keliling Lingkaran

Initial Problem Statement Hitung luas dan keliling lingkaran dari suatu lingkaran berjari-jari r .

Dari *statement* di atas, kita masukkan ke dalam *framework* sebagai berikut:

- **Masalah**

- Menggunakan informasi berupa jari-jari r , kita harus menghitung luas dan keliling lingkaran.

- **Tujuan**

- Menghitung luas lingkaran.
- Menghitung keliling lingkaran.

- **Input**

- Informasi yang dimiliki saat ini adalah jari-jari r .

- **Proses**

- Untuk menghitung luas, kita gunakan fungsi $L = \pi r^2$.
- Untuk menghitung keliling, kita gunakan fungsi $K = 2\pi r$.

- **Output**

- Hasil akhir yang diharapkan adalah luas L dan keliling K .

Secara *simple*, algoritmanya adalah sebagai berikut:

```
INPUT r
COMPUTE L = pi * r^2
      K = 2 * pi * r
OUTPUT L
      K
```

1.6 Kategori *Computational Models*

Seperti halnya model matematika, ada dua pendekatan yang bisa digunakan dalam membuat suatu fungsi, yakni:

1. *Continuous model*

- Salah satu contoh dari model ini adalah model yang menggunakan persamaan diferensial.

2. *Discrete model*

- Salah satu contoh dari model ini adalah pendekatan distribusi *Poisson*⁵ untuk data gol tercipta dalam sebuah pertandingan sepakbola.

1.7 Software Life Cycle

Dalam *software development*, kita bisa menggunakan *waterfall model*. Maksudnya adalah fase berikutnya tidak boleh jalan sebelum fase sebelumnya selesai.

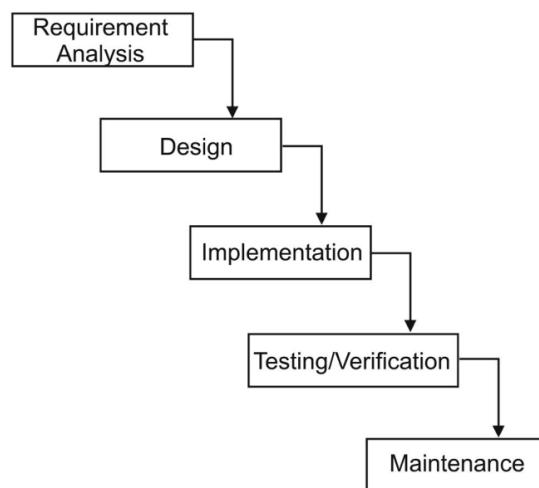


Figure 8: Waterfall Model

⁵https://en.wikipedia.org/wiki/Poisson_distribution

Selain fase yang ada di *waterfall model*, ada fase lainnya yakni:

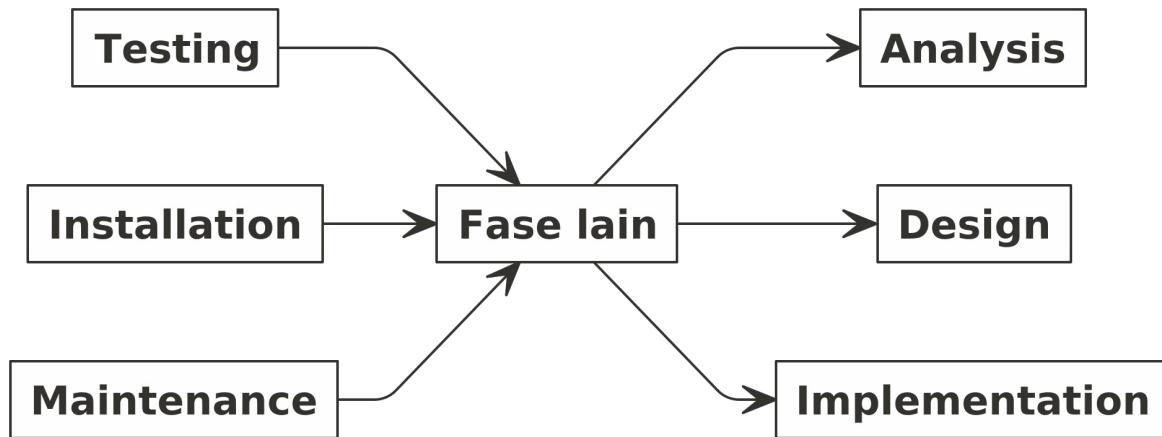


Figure 9: Fase Lain dalam Sorftware Life Cycle

1.8 *Design Modular*

Dalam membuat menghadapi suatu permasalahan yang kompleks, kita bisa menggunakan analogi *chocolate bar*⁶.

If you have to solve a complex problem, you will want to cut it in the smallest pieces as possible, until reaching the most elementary ones, and then expand them little by little to understand the overall problem.

Di dalam buku ini istilah yang digunakan adalah *divide and conquer*. Sejatinya permasalahan yang kompleks bisa dipecah menjadi submasalah-submasalah kecil yang *manage-able*. Proses ini bisa kita sebut sebagai *decomposition*.

Oleh karena itu, *abstraction* dan *decomposition* memegang peranan penting dalam menyelesaikan masalah.

⁶<https://towardsdatascience.com/how-to-solve-complex-problems-efficiently-629c71adcd8d>

Breaking down a problem into a smaller, more manageable, hierarchy of problems.

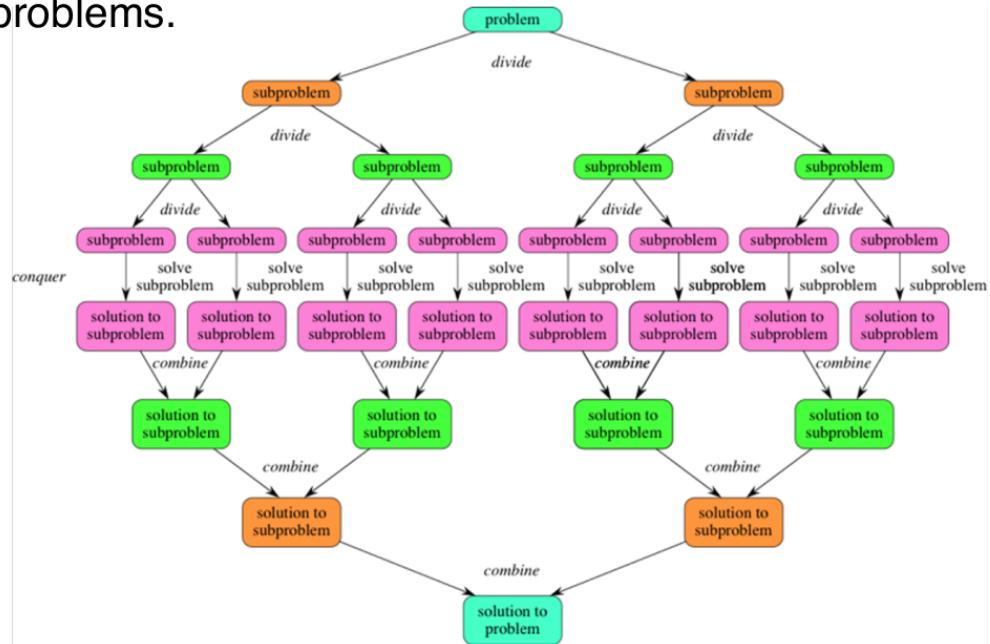


Figure 10: Konsep Decomposition

1.9 Bahasa Pemrograman

Ada banyak bahasa pemrograman di dunia ini. Kita bisa memilih bahasa mana yang sesuai dengan kebutuhan, tujuan, kemampuan, dan permasalahan yang dihadapi. Sebagai contoh:

1. Saat seseorang membutuhkan *output* berupa aplikasi mandiri (*executable file*), tentu dia tidak akan mempertimbangkan bahasa pemrograman MATLAB.
2. Saat seseorang membutuhkan *output* hanya berupa *advance statistical analysis* dari data yang ada, tentu dia tidak akan mempertimbangkan bahasa pemrograman Java atau C.

Salah satu istilah dalam bahasa pemrograman yang sering kita dengar adalah *high level programming language*. Apa artinya?

High level programming language adalah bahasa pemrograman yang bersifat *problem oriented* dan bisa dijalankan tanpa ada keterbatasan di *hardware*.

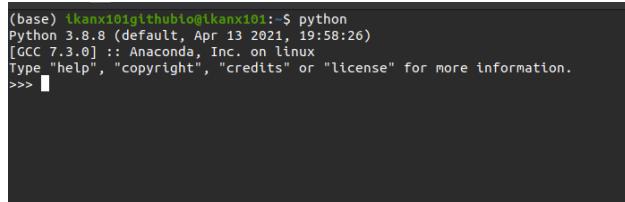
Salah satu ciri lainnya adalah penggunaan skrip yang lebih mirip *human language* dibandingkan *machine language*⁷.

Python merupakan salah satu contoh *high level programming language* yang memiliki **IDE** beragam.

⁷<https://www.bbc.co.uk/bitesize/guides/z4cck2p/revision/1>

1.9.1 Interpreter Python

Interpreter adalah program yang digunakan untuk mengecek skrip yang ditulis dan langsung mengeksekusinya.



```
(base) tkanxi01githubio@tkanxi01: ~$ python
Python 3.8.8 (default, Apr 13 2021, 19:58:26)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Figure 11: Contoh Python yang Dijalankan di Command Prompt

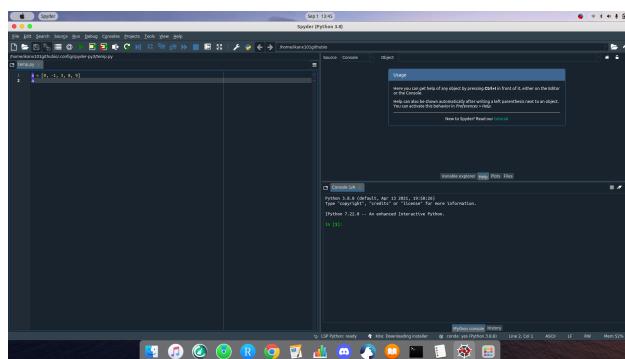


Figure 12: Contoh Python yang Dijalankan di Spyder IDE

1.9.2 Compilers

Berbeda dengan *interpreter*, *compiler* adalah program yang mengubah bahasa pemrograman menjadi bahasa mesin sehingga bisa dijalankan oleh komputer. Salah satu contohnya adalah bahasa pemrograman Pascal yang memungkinkan kita meng-*compile* algoritma menjadi *executable file* (.exe).

1.10 Presisi, Akurasi, dan Galat

Seperti yang telah kita ketahui bersama, solusi yang dihasilkan dalam *computational model* bisa berasal dari metode penyelesaian numerik. Salah satu sifat dasar dari metode numerik adalah **aproksimasi** (pendekatan). Oleh karena itu, kita harus mempertimbangkan galat (*error*) yang ada yakni: akurasi dan presisi.

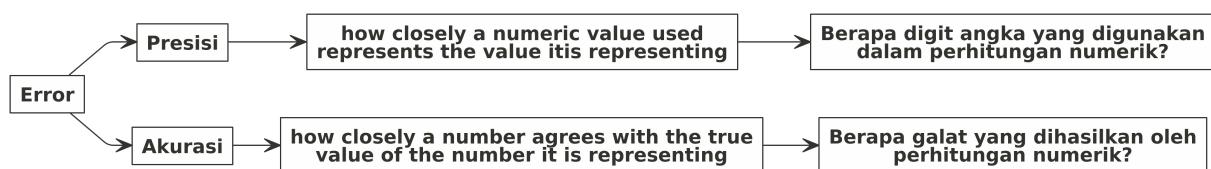


Figure 13: Presisi, Akurasi, dan Galat

Kita bisa menggunakan beberapa definisi galat atau *error* tergantung dengan kebutuhan. Setidaknya ada dua *error* yang biasanya digunakan:

- *Error*: selisih antara *true value* dengan *approximate value*.

Secara matematis kita tuliskan:

$$err = value_{true} - value_{approx}$$

- *Relative error*: ratio dari *error* terhadap *true value*.

Secara matematis kita tuliskan:

$$relative_{err} = \frac{err}{value_{true}} = \frac{value_{true} - value_{approx}}{value_{true}}$$

Salah satu kegunaannya adalah sebagai kriteria penghentian iterasi pada saat kita menggunakan metode numerik tertentu.

Misalkan Kita hendak mencari suatu akar persamaan menggunakan metode Newton. Kita akan *set* terlebih dahulu berapa level akurasi yang masih bisa kita terima sehingga proses iterasi bisa berhenti saat aproksimasi yang dihasilkan sudah **sangat dekat** dengan solusi sebenarnya⁸.

1.10.1 Jenis-Jenis Galat

Berikut adalah beberapa jenis galat yang bisa terjadi saat kita melakukan perhitungan numerik:

1. *Iteration error.*
 2. *Approximation error.*
 3. *Roundoff error.*
- Yakni *error* yang tercipta akibat adanya pembulatan⁹.
 - Contoh:
 - Nilai *exact* dari suatu variabel $x = 1.0104074$
 - Nilai hampiran atau pendekatannya adalah $\hat{x} = 1.0104$
 - Sehingga *error* yang tercipta adalah $\Delta = 1.0104074 - 1.0104$

⁸https://ikanx101.com/blog/newton_method/

⁹[https://hithaldia.in/faculty/sas_faculty/Mrs_Sumana_Mandal/Lecture%20Note%20\(M\(CS\)301%20&%20M\(CS\)401\).pdf](https://hithaldia.in/faculty/sas_faculty/Mrs_Sumana_Mandal/Lecture%20Note%20(M(CS)301%20&%20M(CS)401).pdf)

2 CHAPTER II

Simple Python Programs

2.1 Introduction

Setidaknya ada dua cara bagi kita untuk menuliskan *script* Python, yakni:



Figure 14: Cara Scripting di Python

Pada *summary* chapter ini, saya akan melampirkan *screenshot* atas beberapa *script* yang saya eksekusi di Python CLI menggunakan *terminal*. Versi Python yang saya gunakan adalah versi Python 3.8.8 yang berjalan di OS Ubuntu 20.04.2 LTS.

A screenshot of a Mac OS X Terminal window. The title bar says "Terminal". The window shows the command "python" being run in a terminal session. The output indicates Python 3.8.8 was used, with details about the build date (April 13, 2021) and compiler (GCC 7.3.0). It also mentions Anaconda and Linux. The prompt "Type <help>, <copyright>, <credits> or <license> for more information." is visible. Below the terminal window, the Dock shows various application icons.

```
(base) ikanx101githubio@ikanx101:~$ python
Python 3.8.8 (default, Apr 13 2021, 19:58:26)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Figure 15: Python CLI via Terminal

2.2 Simple Operations

Berikut adalah beberapa perintah atau fungsi dasar di Python.

2.2.1 Mathematical Operations

Kita bisa menggunakan Python untuk melakukan perhitungan seperti halnya *calculator*. Berikut adalah contohnya:

```
(base) ikanx101githubio@ikanx101:~$ python
Python 3.8.8 (default, Apr 13 2021, 19:58:26)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 4
5
>>> 7 * 5
35
>>> 9/ 10
0.9
>>> r = 7
>>> L = 3.14 * r * r
>>> r
7
>>> L
153.86
>>> L % 150
3.8600000000000136
>>> 
```

Figure 16: Mathematical Operations

Jika kita perhatikan dengan baik, saya melakukan *assigning variable* pada *script* di atas dengan menggunakan perintah `=`. Python juga bisa meng-handle data berupa bilangan kompleks¹⁰.

Bilangan kompleks biasa dinotasikan sebagai $z = a + bi$. Di Python didefinisikan sebagai `complex(x,y)` untuk x dan y suatu bilangan tertentu.

Misalkan:

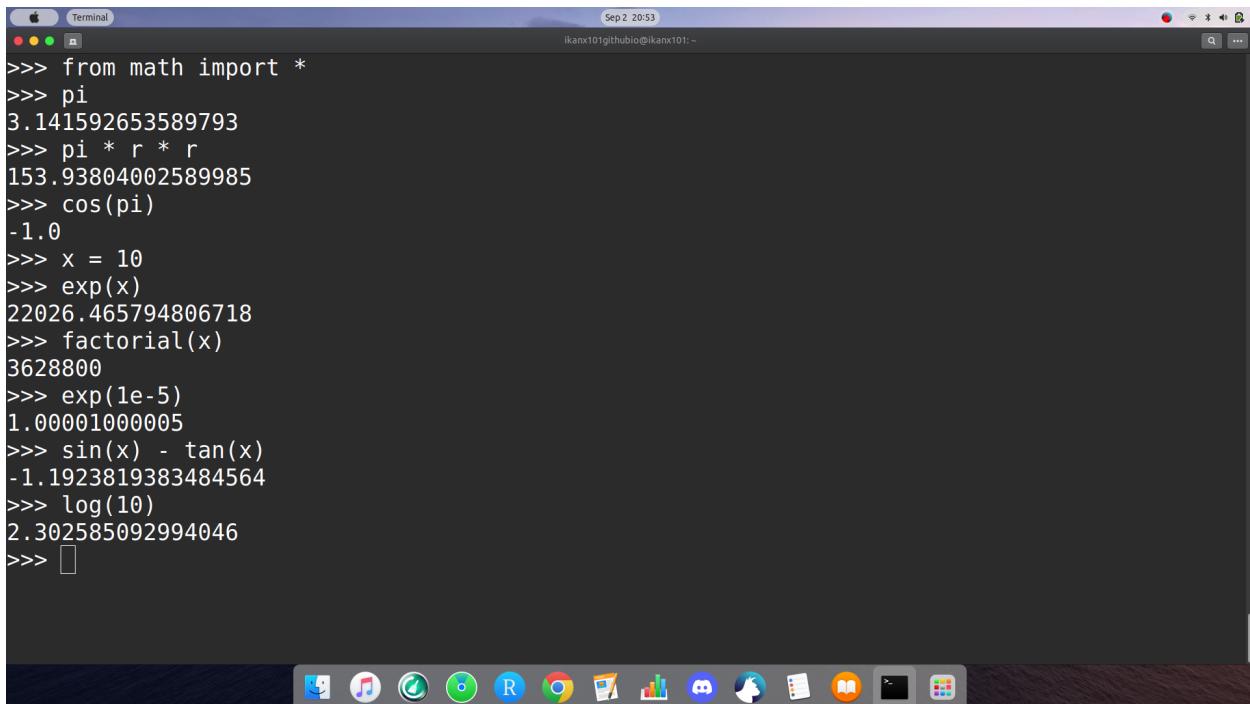
```
>>> complex(1,2)
(1+2j)
```

¹⁰https://id.wikipedia.org/wiki/Bilangan_kompleks

2.2.2 Advance Mathematical Operations

Selain operator matematika dasar, kita juga bisa melakukan operasi matematika yang lebih *advance* dengan memanfaatkan **modul** math. Kita bisa import ke dalam *environment*.

Contohnya:



```

>>> from math import *
>>> pi
3.141592653589793
>>> pi * r * r
153.93804002589985
>>> cos(pi)
-1.0
>>> x = 10
>>> exp(x)
22026.465794806718
>>> factorial(x)
3628800
>>> exp(1e-5)
1.00001000005
>>> sin(x) - tan(x)
-1.1923819383484564
>>> log(10)
2.302585092994046
>>> □

```

Figure 17: Advance Mathematical Operations

Terkait penggunaan *power* atau pangkat, ada aturan penulisan khusus di Python.

Misalkan $x^n \cdot y \cdot \sin^{2m}(z)$ dengan x, y, z suatu bilangan tertentu. Maka penulisannya adalah sebagai berikut:

```

>>> x = 2
>>> y = 1
>>> z = pi
>>> n = .5
>>> y = 2
>>> m = 3
>>> (x**n) * y * sin(z)**(2*m)
9.541356079099411e-96

```

Demikian pula dengan akar kuadrat. Misalkan kita ingin menghitung $\sqrt{z \cdot n + 3m}$ maka:

```
>>> sqrt(z*n + 3*m)
3.2512761074376466
```

2.2.3 Scientific Notation

Dalam beberapa kasus numerik, ada kalanya kita menggunakan beberapa *scientific notation*. Misalkan kita hendak mendefinisikan galat yang bisa diterima. Misalkan $\epsilon = 10^{-5}$. Maka format penulisannya di Python adalah sebagai berikut:

```
>>> eps = 1e-5
>>> eps
1e-05
```

2.3 Programs

Suatu program berisi dua komponen utama, yakni:

1. Pendefinisan data, dan
2. Sekumpulan perintah.

2.3.1 Pendefinisan Data

Suatu program bisa berisi satu atau lebih data. Pendefinisan data dilakukan dengan *assign* suatu *value* ke dalam suatu *variable*.

Python bersifat *case sensitive*, maka penamaan variabel menjadi penting¹¹.

2.3.1.1 Tipe Data di Python Sama halnya dengan **R**¹², ada empat tipe data, yakni:

1. **character**: merupakan tipe data berupa karakter atau string. Semua data bisa dilihat sebagai *character*. Oleh karena itu, secara hierarki tipe data ini ditempatkan di urutan paling atas. Namun, data tipe ini tidak bisa dilakukan operasi aritmatika.
2. **numeric**: merupakan tipe data angka mirip bilangan *real*.
3. **integer**: merupakan tipe data angka berupa bilangan bulat.
4. **logical**: merupakan tipe data *boolean*. Hanya berisi TRUE atau FALSE. Tipe data ini sangat berguna saat kita melakukan *conditional*, *looping*, atau membuat *regex* (*regular expression*).

¹¹Panduan penamaan variabel di **R** <https://ikanx101.com/blog/train-r-4/#tata-cara-memberikan-nama-object-atau-variabel>

¹²Tipe data di data sains <https://ikanx101.com/blog/train-r-4/#tipe-data-di-r>

Secara hierarki: `character > numeric > integer > logical`.

Untuk mengecek suatu variabel merupakan tipe data apa, kita bisa menggunakan perintah `type()`. Contohnya:

```
>>> x = 5
>>> type(x)
int
>>> y = "halo"
>>> type(y)
str
>>> z = True
>>> type(z)
bool
```

2.3.2 Assignment Variable

Assignment suatu variabel dilakukan dengan perintah `=`. Prinsip dasarnya adalah:

```
variable = expression
```

Saya telah menuliskan beberapa contoh di bagian-bagian sebelumnya.

2.3.3 Perintah Dasar *Input* dan *Output*

Input *Input statement* berguna untuk membaca masukan dari *user*. Bentuknya adalah *prompted* sehingga user harus memasukkan *value* yang diinginkan.

Misalkan:

```
>>> usia = input("Berapa usia Anda: ")
Berapa usia Anda:
```

Jika kita masukkan suatu *value*, maka *value* tersebut akan *assigned* ke variabel `usia`.

Output Untuk membuat *output* dari suatu variabel, kita bisa menggunakan perintah `print(variable)`.

Contohnya adalah sebagai berikut:

```
>>> print(y)
halo
>>> print("nilai x adalah ", x)
nilai x adalah 5
```

2.3.4 Contoh Membuat *Script*

Kita bisa membuat *script* di *software text editor* lalu menyimpannya dalam format .py untuk kemudian di-run di *interpreter*.

Saya akan berikan contoh sederhana program yang bertujuan untuk menghitung *body mass index* seseorang.

```
nama = input("Masukkan nama Anda: ")
berat = input("Masukkan berat badan Anda: ")
tinggi = input("Masukkan tinggi badan Anda: ")

berat = int(berat)
tinggi = int(tinggi)

bmi = berat / (tinggi/100)**2
bmi = round(bmi,2)

print("Yth. Bpk/Ibu ",nama," , BMI Anda adalah sebesar: ",bmi)
```

Saya akan simpan *script* di atas bernama bmi.py.

Jika saya ingin memanggilnya di *terminal* tanpa harus membuka *script*, saya cukup ketikkan:

```
python bmi.py
```

Berikut adalah tampilannya:

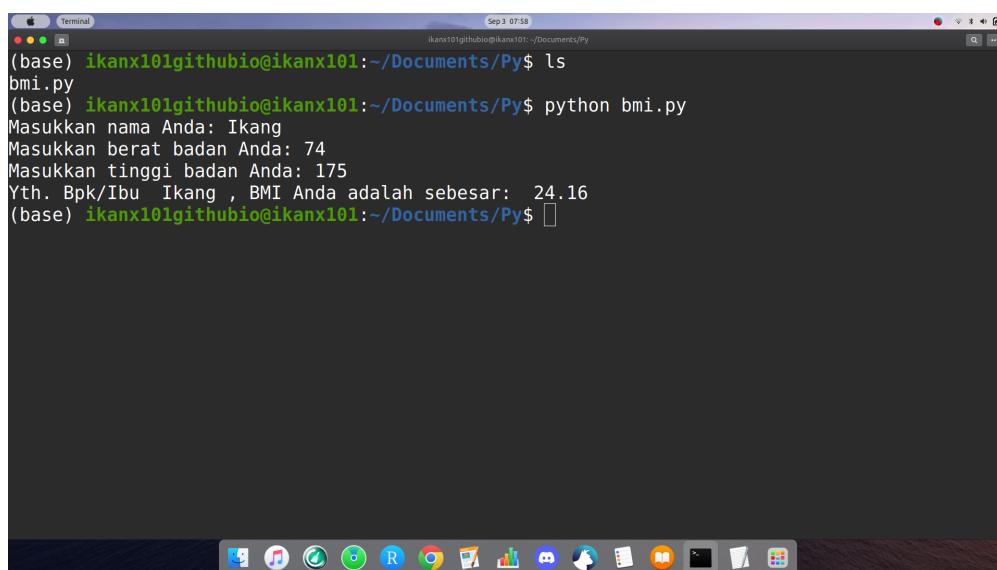


Figure 18: Run .py di Terminal Linux

2.4 Contoh Program Sederhana

Dari buku, ada beberapa program sederhana yang dicontohkan, yakni:

1. *Converter* temperatur Celcius ke Farenheit.
2. Penghitung *Euclidean distance* antara dua titik koordinat (x_1, y_1) dan (x_2, y_2) .

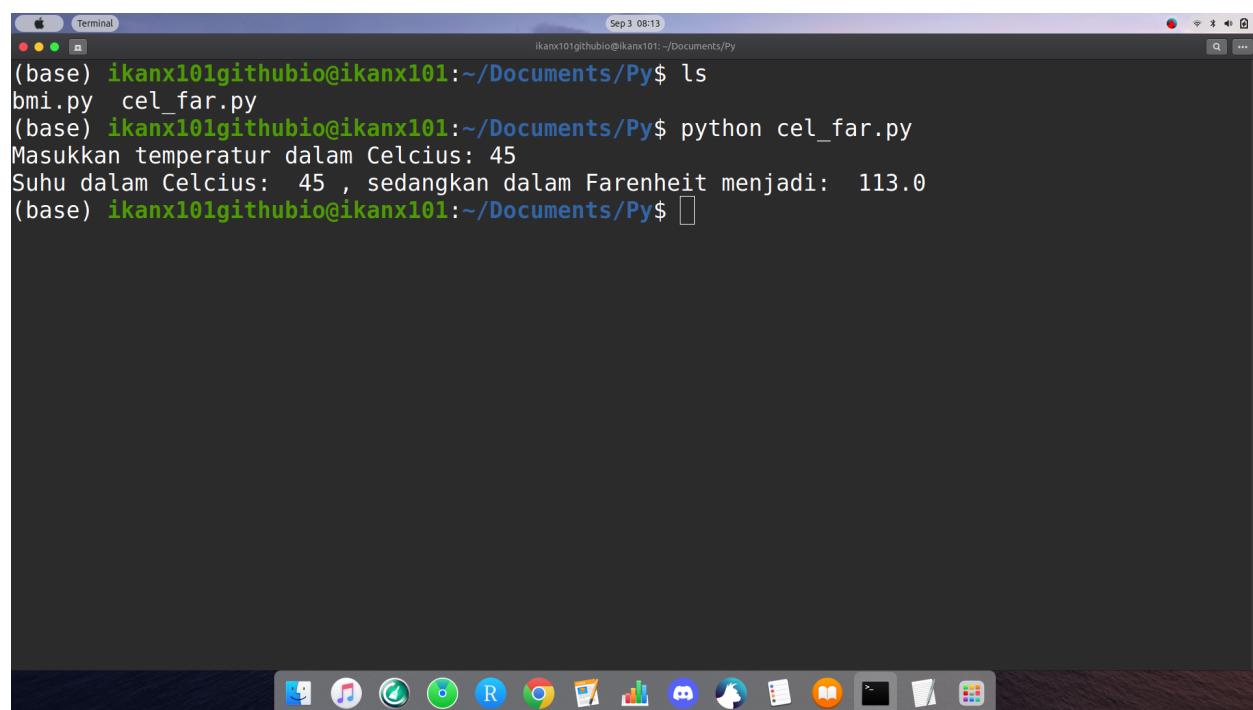
Untuk memudahkan penjelasan, saya akan gunakan kembali **framework problem - output** pada Figure 7.

2.4.1 *Converter* Celcius ke *Farenheit*

Maka bentuk programnya adalah sebagai berikut:

```
celcius = input("Masukkan temperatur dalam Celcius: ")  
celcius = int(celcius)  
far = celcius * (9.0/5.0) + 32.0  
  
print("Suhu dalam Celcius: ",celcius,", sedangkan dalam Farenheit menjadi: ",far)
```

Saya simpan skrip di atas dengan nama `cel_far.py`. Jika saya *run* di *terminal*:



The screenshot shows a Mac OS X desktop with a terminal window open. The terminal window title is "Terminal". The command prompt shows the user's name and the path: "(base) ikanx101githubio@ikanx101:~/Documents/Py\$". The user runs the command "ls" to list files, showing "bmi.py" and "cel_far.py". Then, the user runs the script with "python cel_far.py". The terminal displays the user's input "Masukkan temperatur dalam Celcius: 45" and the output "Suhu dalam Celcius: 45 , sedangkan dalam Farenheit menjadi: 113.0". The terminal window has a dark background and a light-colored text area. At the bottom, there is a dock with various application icons.

Figure 19: Run Converter Celcius Farenheit di Terminal Linux

2.4.2 Penghitung *Euclidean Distance*

Dari *statement* di atas, kita masukkan ke dalam *framework* sebagai berikut:

- **Masalah**

- Menghitung jarak antara dua titik koordinat (x_1, y_1) dan (x_2, y_2) .

- **Tujuan**

- Menghitung jarak antara dua titik koordinat (x_1, y_1) dan (x_2, y_2) .

- **Input**

- Titik (x_1, y_1) dan (x_2, y_2) .

- **Proses**

- Untuk menghitung jarak, kita akan gunakan teorema Phytagoras $jarak = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

- **Output**

- Hasil akhir yang diharapkan adalah nilai *jarak*.

Misalkan saya memiliki titik $(1, 4)$ dan $(5, 10)$ sebagai berikut:

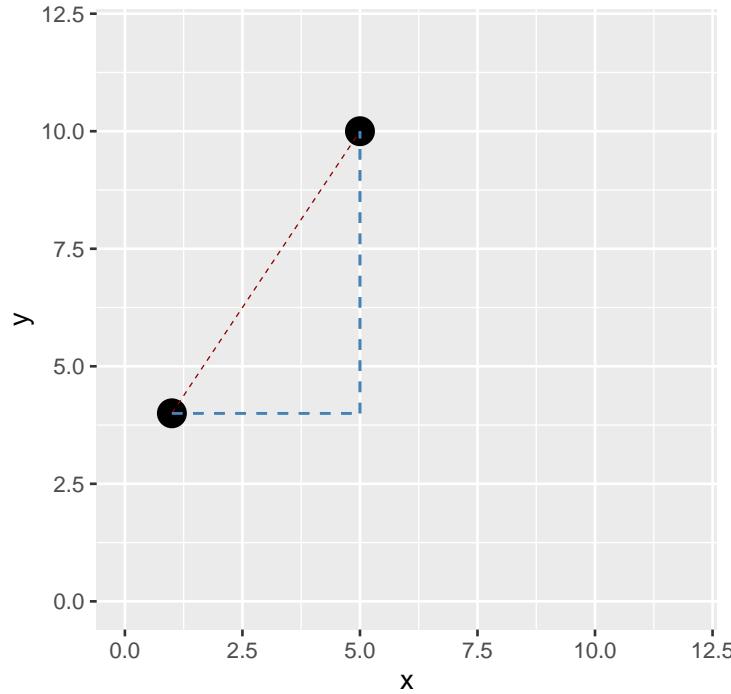


Figure 20: Ilustrasi Perhitungan Jarak

Jarak kedua titik bisa dipandang sebagai sisi miring dari segitiga siku-siku yang dibentuk.

Secara *simple*, *script*-nya adalah sebagai berikut:

```
x1 = input("Masukkan titik koordinat x1: ")
y1 = input("Masukkan titik koordinat y1: ")
x2 = input("Masukkan titik koordinat x2: ")
y2 = input("Masukkan titik koordinat y2: ")

x1 = int(x1)
x2 = int(x2)
y1 = int(y1)
y2 = int(y2)

del_x = (x1-x2)**2
del_y = (y1-y2)**2

import math
jarak = math.sqrt(del_x + del_y)

print("Jarak antara kedua titik adalah: ", jarak)
```

Saya simpan sebagai *file* bernama *jarak.py*. Jika saya *run* di *terminal*, berikut hasilnya:

```
Terminal Sep 3 08:59
ikanx101githubio@ikanx101:~/Documents/Py$ ls
bmi.py cel_far.py jarak.py
(base) ikanx101githubio@ikanx101:~/Documents/Py$ python jarak.py
Masukkan titik koordinat x1: 1
Masukkan titik koordinat y1: 4
Masukkan titik koordinat x2: 5
Masukkan titik koordinat y2: 10
Jarak antara kedua titik adalah: 7.211102550927978
(base) ikanx101githubio@ikanx101:~/Documents/Py$
```

Figure 21: Run Program Jarak di Terminal Linux

2.5 Struktur Umum Program Python

Secara umum, berikut adalah urutan isi program Python:

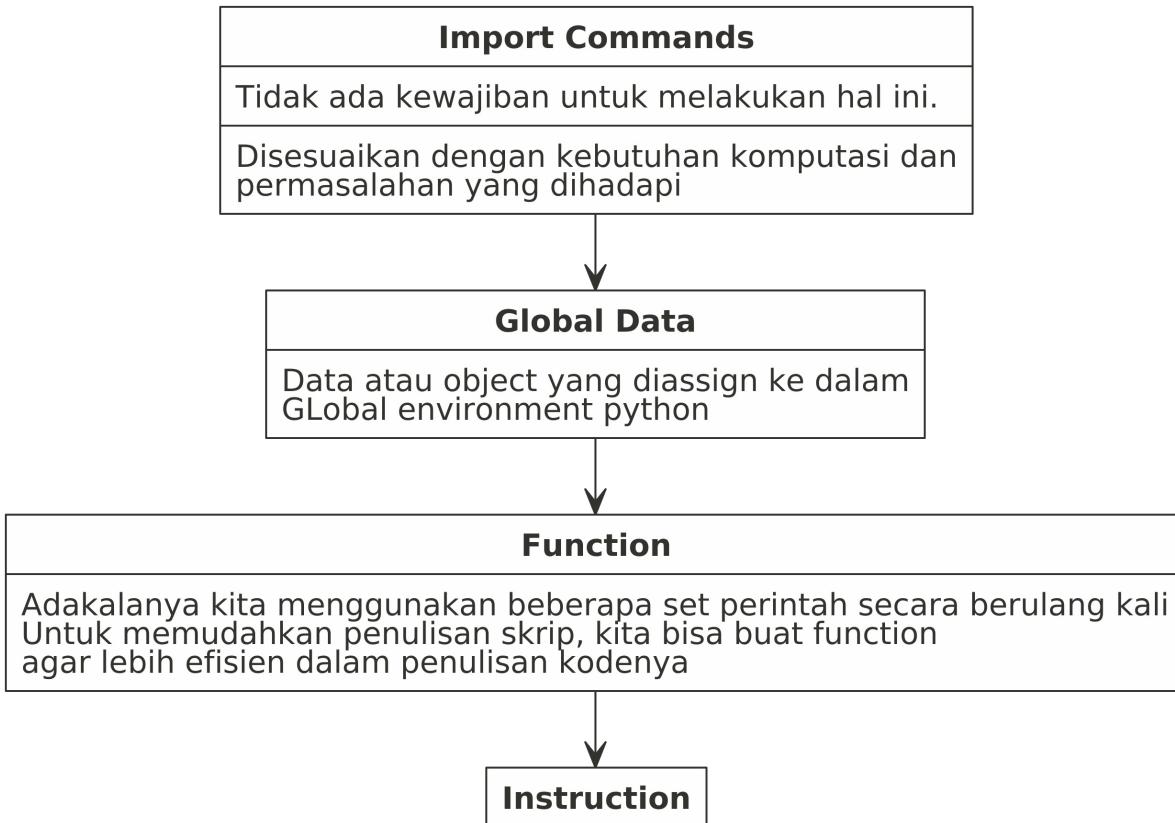


Figure 22: Struktur Program Python

2.6 Pendefinisian *Function*

Pada Python, kita bisa membuat *custom function* sendiri. Gunanya adalah agar *set* perintah yang biasa kita gunakan berulang bisa menjadi lebih singkat penulisannya.

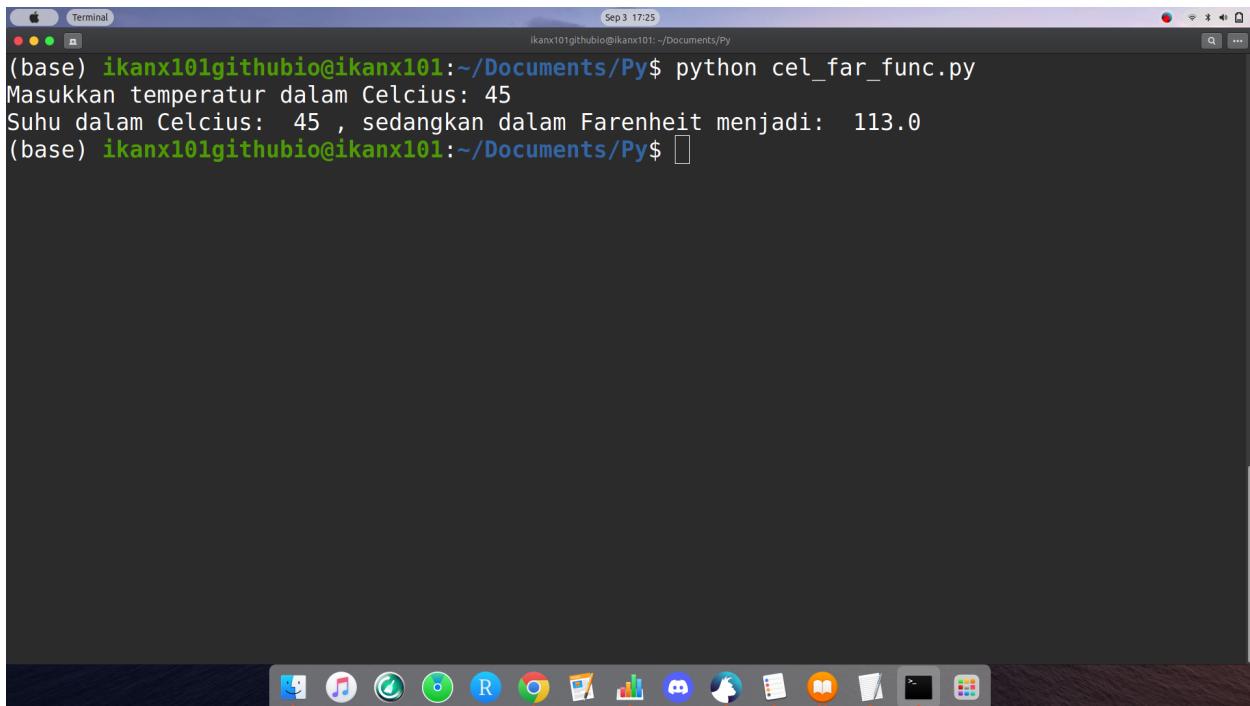
Struktur menuliskannya adalah sebagai berikut:

```
def function_name ( [parameters] ) :  
    [ local declarations ]  
    [ executable language statements ]
```

Kita buat contoh saja dari permasalahan di bagian sebelumnya, yakni konversi suhu. Saya akan membuat satu *function* untuk itu.

```
def convert (c):  
    f = c * (9.0/5.0) + 32.0  
    return(f)  
  
celcius = input("Masukkan temperatur dalam Celcius: ")  
celcius = int(celcius)  
print("Suhu dalam Celcius: ",celcius,",  
      sedangkan dalam Farenheit menjadi: ",convert(celcius))
```

Saya simpan dalam file bernama `cel_far_func.py`. Berikut jika saya *run* dalam *terminal*:



The screenshot shows a macOS Terminal window titled "Terminal". The command `python cel_far_func.py` is run, prompting for input of temperature in Celcius (45), and then outputting the converted temperature in Fahrenheit (113.0).

```
Sep 3 17:25
ikanx101githubio@ikanx101:~/Documents/Py$ python cel_far_func.py
Masukkan temperatur dalam Celcius: 45
Suhu dalam Celcius: 45 , sedangkan dalam Farenheit menjadi: 113.0
ikanx101githubio@ikanx101:~/Documents/Py$
```

Figure 23: Run Program Celcius Farenheit dalam Function

== End ==