

## 4.2

# STRUKTUR KOD ARAHAN

Dalam kehidupan seharian, suatu masalah harus dianalisis, dikenal pasti dan dibahagikan kepada beberapa submasalah melalui proses leraian supaya masalah tersebut bukan sahaja kelihatan kurang rumit tetapi lebih mudah untuk diselesaikan. Bagi setiap submasalah, suatu algoritma khusus boleh dihasilkan. Gabungan kesemua algoritma yang dihasilkan akan mampu menyelesaikan masalah asal tersebut.

Pembangunan atur cara juga dilakukan dengan cara yang sama. Masalah pengaturcaraan dibahagikan kepada tugas utama dan beberapa tugas kecil (subtugas). Bagi setiap tugas kecil, satu algoritma dapat dibina.

Semasa fasa pengekodan, algoritma bagi tugas utama diterjemahkan kepada atur cara utama dan algoritma setiap tugas kecil diterjemahkan kepada subatur cara yang dipanggil *function* atau *procedure*. *Function* dan *procedure* akan diguna pakai dalam atur cara utama bagi menyelesaikan masalah tersebut.

4.2.1

### Fungsi Function dan Procedure dalam Atur Cara

#### (i) Function

*Function* sesuai digunakan bagi subtugas yang perlu memulangkan satu nilai selepas tugas itu diselesaikan. Sebagai contoh, dalam permainan yang melibatkan pembelian, situasi pengiraan dana yang sedia ada dan baki dana selepas pembelian diperlukan. *Function* sesuai digunakan bagi proses pengiraan kerana kod yang sama tidak perlu ditulis semula setiap kali pemain ingin melakukan pembelian. Rajah 4.42 dan Rajah 4.43 menunjukkan paparan simulasi aplikasi android MyStemVille bagi peringkat pembelian.

Apabila pemilihan dibuat dan pembelian disahkan, *function* yang berada di belakang item-item yang dipilih itu akan terlaksana. Pengiraan berlaku dan baki terakhir dikemaskinikan di ruang penjuru sebelah kiri. Perhatikan bahawa sebelum pembelian, baki sedia ada ialah 220 dan selepas pembelian disahkan, baki terbaharu ialah 20 iaitu proses pengiraan dilakukan oleh *function* dan nilai baki terakhir dipulangkan untuk dipaparkan.

### Tahukah Anda?

Python ialah bahasa pengaturcaraan sumber terbuka. Python telah mengalami banyak penambahbaikan dengan keluaran versi-versi baharu sejak mula digunakan pada tahun 1991.

### Standard Pembelajaran

Murid boleh:

- 4.2.1 Menerangkan fungsi struktur berikut dalam atur cara:  
 (i) *Function*  
 (ii) *Procedure*

Function dan procedure



goo.gl/LGsjnK





Rajah 4.42 Paparan simulasi permainan **MyStemVille**

### (ii) *Procedure*



**Procedure** sesuai digunakan dalam situasi di mana satu tugas perlu dilaksanakan berulang kali apabila ia diperlukan tanpa pemulangan nilai. Perhatikan bahawa *procedure* yang disimpan dalam butang profil akan terlaksana setiap kali butang ini diklik. *Procedure* ini akan memaparkan tetapan yang mengandungi profil pemain seperti yang ditunjukkan dalam Rajah 4.43.



Rajah 4.43 Paparan profil pemain

Penggunaan subatur cara seperti *function* dan *procedure* sesuai untuk pelaksanaan tugas yang khusus atau berulang kali dalam satu atur cara yang sama atau dalam atur cara yang berlainan. Ini menjadikan kod arahan seluruh atur cara menjadi lebih kemas, teratur, sistematik, bersifat modular dan lebih mudah dinyahepejat serta diselenggara.

Penggunaan *function* dan *procedure* dalam menghasilkan suatu atur cara merupakan satu pendekatan yang harus dikuasai oleh setiap murid. Kemahiran ini akan membolehkan murid memainkan peranannya sebagai *team player* dalam menghasilkan atur cara yang bertaraf komersial.

Dalam kebanyakan bahasa pengaturcaraan, *function* dan *procedure* mempunyai persamaan dan perbezaan yang jelas dari segi takrifan dan kegunaan. Dalam bahasa pengaturcaraan Python, *procedure* tidak wujud kerana *procedure* dianggap sebagai *implicit function*. *Procedure* Python tetap memulangkan nilai *None* walaupun memainkan peranan

yang sama seperti *procedure* dalam bahasa pengaturcaraan yang lain, iaitu tidak melakukan sebarang pengiraan dan hanya memaparkan maklumat. *None* ialah nilai lalai dalam Python. Takrifan dan kegunaan *function* serta *procedure* dalam buku ini akan dibuat berdasarkan bahasa pengaturcaraan Python. Rajah 4.44 menunjukkan persamaan dan perbezaan antara *function* dan *procedure* Python.

Perbezaan antara *function* dan *procedure* dalam Python.



[goo.gl/5jxz9z](http://goo.gl/5jxz9z)

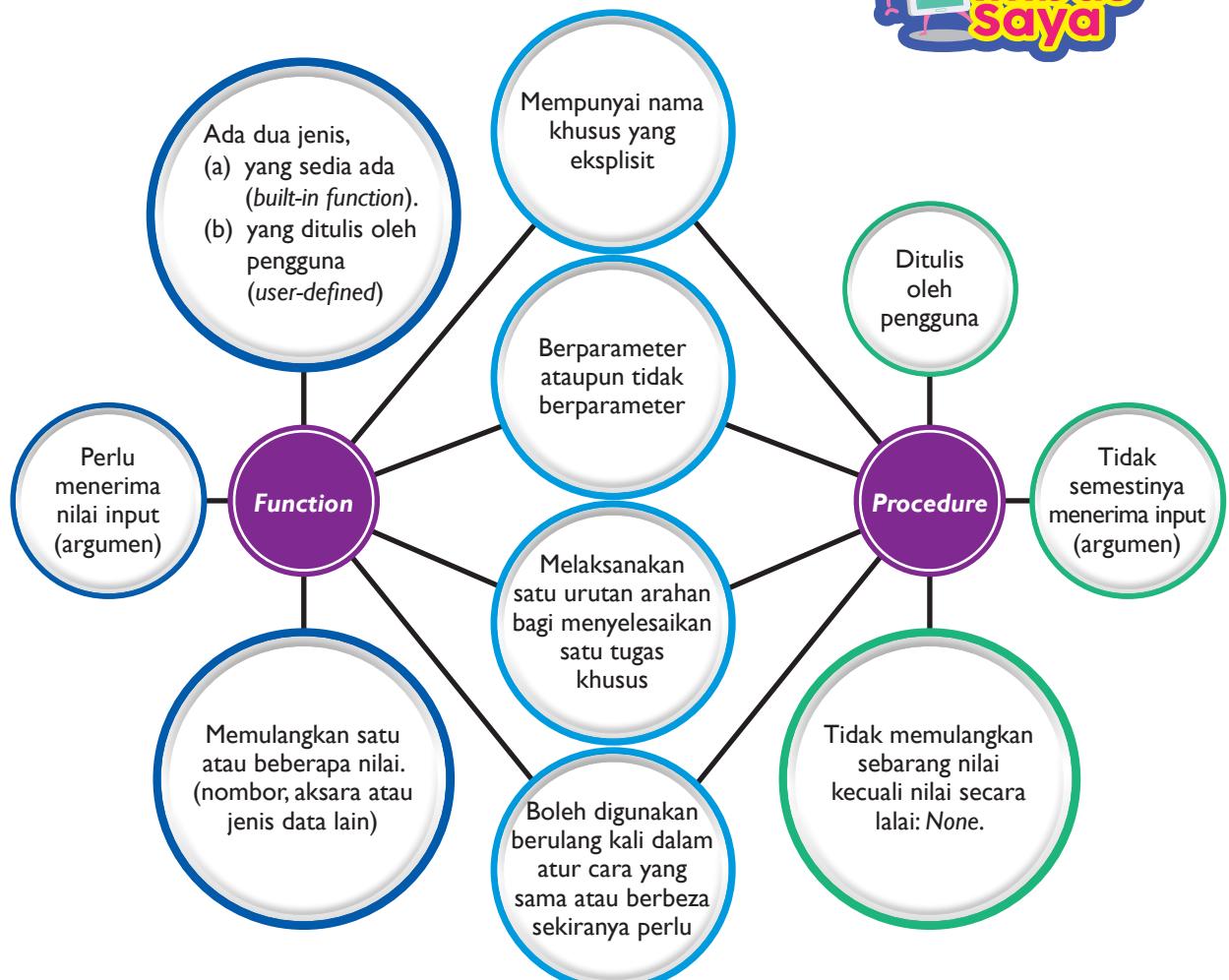
(Dipetik dari: <http://index-of.es/Python/Core.Python>.

Programming.2nd.  
Edition.Wesley.  
Chun.2006.pdf,  
muka surat 448)

## Tahukah Anda ?

**Parameter** ialah pemboleh ubah yang digunakan oleh sesuatu *function* atau *procedure*.

**Argumen** ialah data atau nilai sebenar dalam pemboleh ubah tersebut.



Rajah 4.44 Persamaan dan perbezaan antara *function* dan *procedure* Python

## Standard Pembelajaran

Murid boleh:

- 4.2.2 Memberi contoh penggunaan pernyataan *function*:
- (i) dalaman (*built-in*)
  - (ii) dihasilkan sendiri (*user-defined*)

### 4.2.2 Penggunaan Pernyataan *Function*

*Function* boleh dibahagikan kepada *built-in function* (fungsi dalaman) dan *user-defined function* (fungsi dihasilkan sendiri). Kedua-duanya mempunyai fungsi yang sama, iaitu melakukan suatu tugas yang khusus.

#### (i) *Function* dalaman (*built-in*)

*Built-in function* ialah fungsi yang sedia ada dan disimpan dalam *library* bahasa pengaturcaraan. Kod sumber untuk setiap *built-in function* tidak boleh dilihat oleh pengatur cara. Setiap *built-in function* mempunyai nama yang deskriptif, iaitu nama yang dapat mencerminkan tugas *built-in function* berkenaan. Nama *built-in function* tidak boleh ditukar. Dalam pengaturcaraan Python, fungsi input dan fungsi output ialah *built-in function* yang sering digunakan. Tugas yang dilakukan oleh *built-in function* yang dibekalkan adalah terhad dan ringkas sahaja.

Bagi kebanyakan bahasa pengaturcaraan, sebelum sesuatu *built-in function* boleh digunakan, nama fail *library* di mana *built-in function* berkenaan disimpan hendaklah dinyatakan terlebih dahulu di bahagian atas atur cara. Dalam pengaturcaraan Python, terdapat sekumpulan *built-in function* yang boleh digunakan secara terus tanpa menyatakan terlebih dahulu nama fail *library* di mana fungsi berkenaan disimpan.

Function  
dalaman bahasa  
pengaturcaraan  
Python



goo.gl/DWBx41

Imbas  
Saya

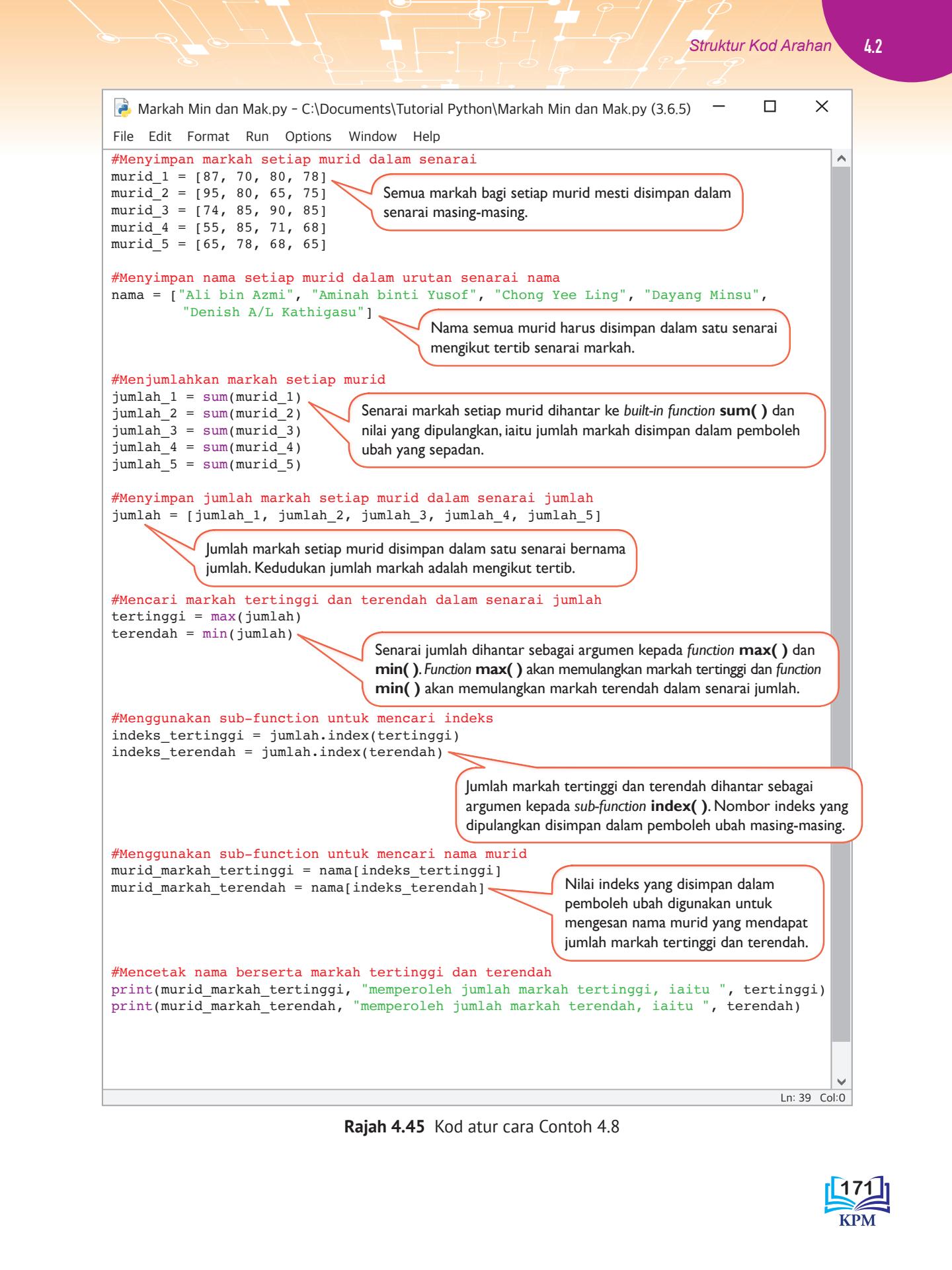
### Contoh 4.8

Penggunaan *built-in function* dalam bahasa pengaturcaraan Python.

Contoh ini menunjukkan penggunaan *built-in function* `sum()`, `max()`, `min()` dan `index()` bagi mendapatkan nama murid yang memperoleh jumlah markah tertinggi dan terendah. Jadual 4.11 menunjukkan markah yang diperoleh lima murid bagi empat mata pelajaran.

Jadual 4.11

ID murid	Nama murid	BM	BI	Mate	Sains
0001	Ali bin Azmi	87	70	80	78
0002	Aminah binti Yusof	95	80	65	75
0003	Chong Yee Ling	74	85	90	85
0004	Dayang Minsu	55	85	71	68
0005	Denish A/L Kathigasu	65	78	68	65



Markah Min dan Mak.py - C:\Documents\Tutorial Python\Markah Min dan Mak.py (3.6.5) — X

File Edit Format Run Options Window Help

```

#Menyimpan markah setiap murid dalam senarai
murid_1 = [87, 70, 80, 78]
murid_2 = [95, 80, 65, 75]
murid_3 = [74, 85, 90, 85]
murid_4 = [55, 85, 71, 68]
murid_5 = [65, 78, 68, 65]

#Menyimpan nama setiap murid dalam urutan senarai nama
nama = ["Ali bin Azmi", "Aminah binti Yusof", "Chong Yee Ling", "Dayang Minsu",
         "Denish A/L Kathigasu"]

#Menjumlahkan markah setiap murid
jumlah_1 = sum(murid_1)
jumlah_2 = sum(murid_2)
jumlah_3 = sum(murid_3)
jumlah_4 = sum(murid_4)
jumlah_5 = sum(murid_5)

#Menyimpan jumlah markah setiap murid dalam senarai jumlah
jumlah = [jumlah_1, jumlah_2, jumlah_3, jumlah_4, jumlah_5]

#Mencari markah tertinggi dan terendah dalam senarai jumlah
tertinggi = max(jumlah)
terendah = min(jumlah)

#Menggunakan sub-function untuk mencari indeks
indeks_tertinggi = jumlah.index(tertinggi)
indeks_terendah = jumlah.index(terendah)

#Menggunakan sub-function untuk mencari nama murid
murid_markah_tertinggi = nama[indeks_tertinggi]
murid_markah_terendah = nama[indeks_terendah]

#Mencetak nama berserta markah tertinggi dan terendah
print(murid_markah_tertinggi, "memperoleh jumlah markah tertinggi, iaitu ", tertinggi)
print(murid_markah_terendah, "memperoleh jumlah markah terendah, iaitu ", terendah)

```

Semua markah bagi setiap murid mesti disimpan dalam senarai masing-masing.

Nama semua murid harus disimpan dalam satu senarai mengikut tertib senarai markah.

Senarai markah setiap murid dihantar ke *built-in function sum()* dan nilai yang dipulangkan, iaitu jumlah markah disimpan dalam pemboleh ubah yang sepadan.

Jumlah markah setiap murid disimpan dalam satu senarai bernama jumlah. Kedudukan jumlah markah adalah mengikut tertib.

Senarai jumlah dihantar sebagai argumen kepada *function max()* dan *min()*. *Function max()* akan memulangkan markah tertinggi dan *function min()* akan memulangkan markah terendah dalam senarai jumlah.

Jumlah markah tertinggi dan terendah dihantar sebagai argumen kepada *sub-function index()*. Nombor indeks yang dipulangkan disimpan dalam pemboleh ubah masing-masing.

Nilai indeks yang disimpan dalam pemboleh ubah digunakan untuk mengesahkan nama murid yang mendapat jumlah markah tertinggi dan terendah.

Rajah 4.45 Kod atur cara Contoh 4.8

Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

```
===== RESTART: C:\Documents\Tutorial Python\Markah Min dan Mak.py =====
Chong Yee Ling memperoleh jumlah markah tertinggi, iaitu 334
Denish A/L Kathigasu memperoleh jumlah markah terendah, iaitu 276
>>>
```

Ln: 7 Col:4

Rajah 4.46 Output bagi kod atur cara Contoh 4.8



### (ii) Function dihasilkan sendiri (*user-defined*)

*User-defined function* ialah satu set arahan yang ditulis oleh pengatur cara untuk melaksanakan suatu tugas khas yang akan berulang dalam atur cara utama. *User-defined function* yang ditulis juga perlu mempunyai nama yang deskriptif, iaitu nama yang dapat mencerminkan tugasnya. Lazimnya, *user-defined function* ditulis kerana ketiadaan *built-in function* yang dapat memenuhi kehendak pengatur cara. Seperti *built-in function*, *user-defined function* mampu menerima argumen dan memulangkan nilai kepada atur cara yang memanggilnya.

### Contoh 4.9

Penggunaan *user-defined function* yang mengira kuasa dua dalam bahasa pengaturcaraan Python.

kuasa\_dua.py - C:\Documents\Tutorial Python\kuasa\_dua.py (3.6.5)

File Edit Format Run Options Window Help

```
# Fungsi yang mengira kuasa dua
def kuasadua(x):
    return x*x

# Bahagian utama atur cara
# Minta pengguna memasukkan satu nombor
nom = int(input("Masukkan satu nombor integer: "))

# panggilan fungsi
nom_kuasa = kuasadua(nom)
print("Kuasa dua bagi", nom, "ialah", nom_kuasa)
```

User-defined function (kuasadua) ini:

- Menerima nilai argumen yang dihantar dan menyimpannya dalam parameter x
- Mengira nilai  $x^2$
- Memulangkan hasil kiraan kepada pernyataan kod yang meminta kiraan ini

Pernyataan kod ini memanggil function kuasadua dan menghantar nilai dalam pemboleh ubah nom kepadanya.

- Terima nilai yang dipulangkan oleh function kuasadua dan umpukan kepada pemboleh ubah nom\_kuasa.

Ln: 1 Col:0

Rajah 4.47 Kod atur cara Contoh 4.9

Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

```
===== RESTART: C:\Documents\Tutorial Python\kuasa_dua.py =====
Masukkan satu nombor integer: 2
Kuasa dua bagi 2 ialah 4
>>>
===== RESTART: C:\Documents\Tutorial Python\kuasa_dua.py =====
Masukkan satu nombor integer: 5
Kuasa dua bagi 5 ialah 25
>>>
```

Ln: 11 Col:4

Rajah 4.48 Output bagi kod atur cara Contoh 4.9



### Uji Minda 4.17

Nyatakan *built-in function* yang digunakan dalam bahagian utama atur cara dalam Contoh 4.9 yang membolehkan pengguna memasukkan nombor.

### Contoh 4.10

Penggunaan *user-defined function* yang membandingkan dua nombor dalam bahasa pengaturcaraan Python.

susun\_nombor.py - C:\Documents\Tutorial Python\susun\_nombor.py (3.6.5)

File Edit Format Run Options Window Help

```
""" User-defined function yang memulangkan nombor besar di kiri
dan nombor kecil di kanan """
def besar_kecil(x,y):
    if x > y:
        return x, y
    else:
        return y, x

# Bahagian utama atur cara
# Minta pengguna memasukkan dua nombor
a = int(input("Masukkan nombor integer yang pertama: "))
b = int(input("Masukkan nombor integer yang kedua: "))

# panggilan user-defined function
[besar, kecil] = besar_kecil(a,b)
print("Nombor",besar,"lebih besar daripada",kecil)
```

- Memanggil *function* besar\_kecil serta menghantar dua nilai ke pemboleh ubah x dan y.
- Nilai yang dipulangkan akan diumpukan kepada senarai [besar, kecil].
- Nilai-nilai dipaparkan.

User-defined function ini:

- Menerima dua nilai argumen dan simpan dalam parameter x dan y
- Membandingkan dua nilai dalam parameter dan terbalikkan tempat jika y > x
- Memulangkan kedua-dua nilai mengikut susunan nombor lebih besar di kiri dan nombor lebih kecil di kanan

Rajah 4.49 Kod atur cara Contoh 4.10

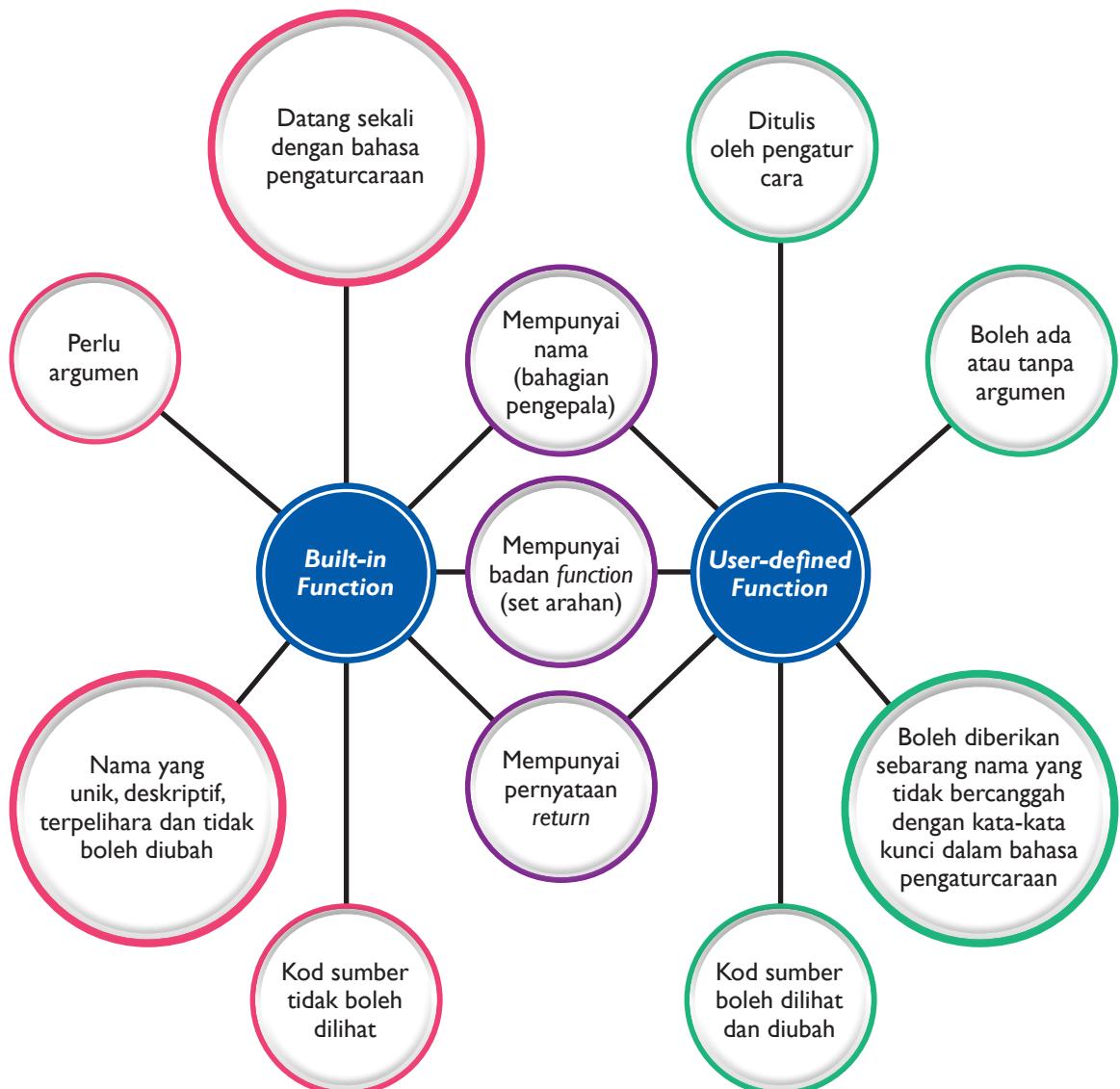
Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

```
===== RESTART: C:\Documents\Tutorial Python\susun_nombor.py =====
Masukkan nombor integer yang pertama: 6
Masukkan nombor integer yang kedua: 9
Nombor 9 lebih besar daripada 6
>>>
```

Ln: 8 Col:4

Rajah 4.50 Output bagi kod atur cara Contoh 4.10



Rajah 4.51 Persamaan dan perbezaan antara *built-in function* dan *user-defined function*

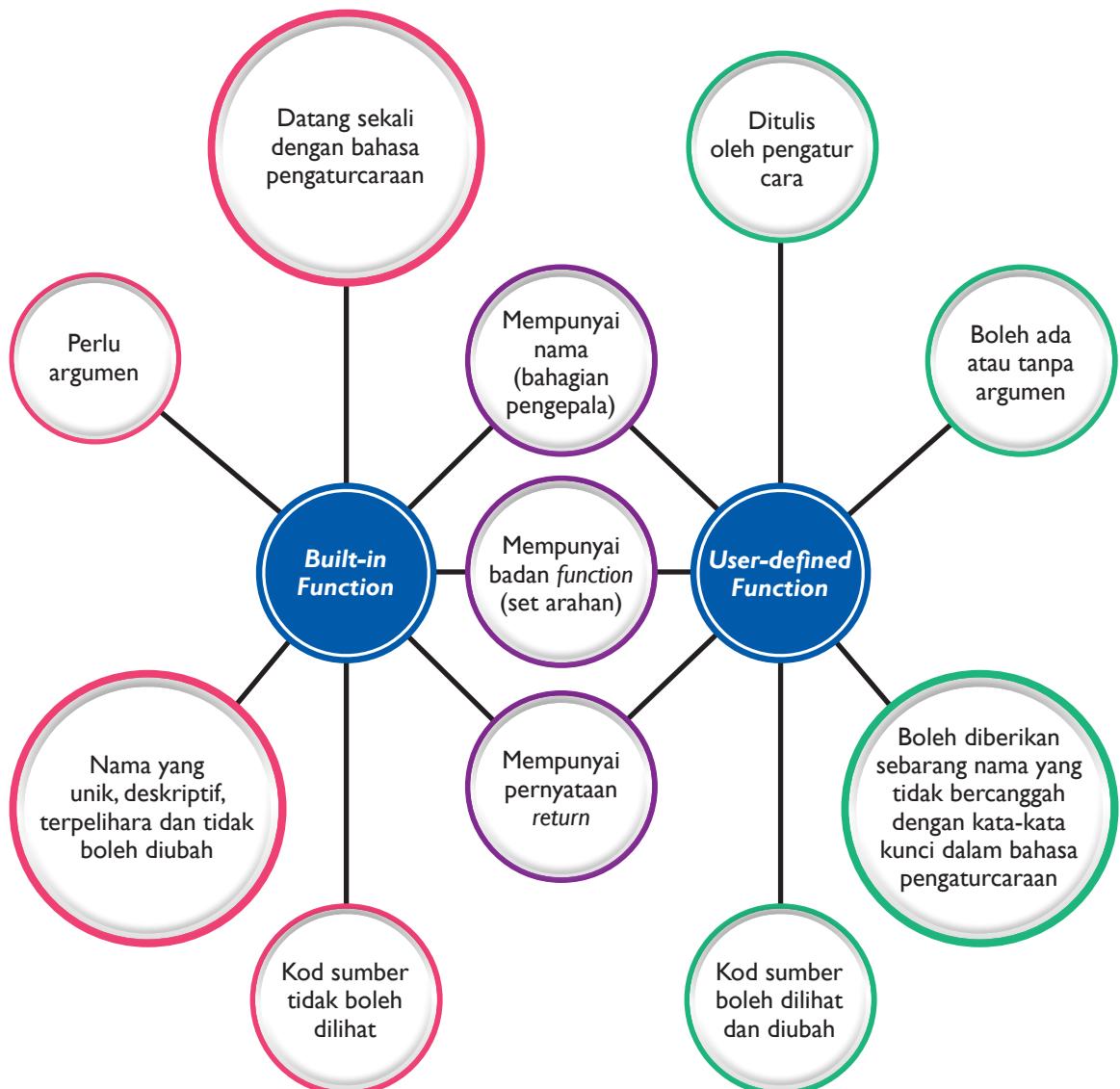
Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

```
===== RESTART: C:\Documents\Tutorial Python\susun_nombor.py =====
Masukkan nombor integer yang pertama: 6
Masukkan nombor integer yang kedua: 9
Nombor 9 lebih besar daripada 6
>>>
```

Ln: 8 Col:4

Rajah 4.50 Output bagi kod atur cara Contoh 4.10



Rajah 4.51 Persamaan dan perbezaan antara *built-in function* dan *user-defined function*



## Aktiviti 4.8



### Aktiviti Kumpulan

#### Mengenal Pasti *Function* dan *Procedure*



K-21

Kaedah  
Think-Pair-Share

- Kaji kod atur cara berikut, kenal pasti semua *function* dan *procedure*.

```

Harga Tempahan Kek.py - C:\Documents\Tutorial Python\Harga Temp...
File Edit Format Run Options Window Help
jenis_kek = ["keju", "mentega", "pelangi", "kopi"]
harga_kek = [40,35,35,30]
jumlah = [0,1,2,3]

a = int(input("Masukkan tempahan untuk kek keju: "))
b = int(input("Masukkan tempahan untuk kek mentega: "))
c = int(input("Masukkan tempahan untuk kek pelangi: "))
d = int(input("Masukkan tempahan untuk kek kopi: "))

tempahan = [a,b,c,d]

def jumlah_harga():
    for i in range(4):
        jumlah[i] = harga_kek[i] * tempahan[i]
    return (jumlah)

def cetak():
    print("\n\nTempahan anda ialah:")
    print(a, "kek", jenis_kek[0])
    print(b, "kek", jenis_kek[1])
    print(c, "kek", jenis_kek[2])
    print(d, "kek", jenis_kek[3])
    print("\nJumlah harga untuk tempahan ialah RM", sum(jumlah))

jumlah_harga()
cetak()

```

[goo.gl/wMqamQ](http://goo.gl/wMqamQ)

- Kenal pasti perkara berikut dan tulis pada sehelai kertas.
  - Jenis dan fungsi *built-in function*
  - User-defined function* dan *procedure* serta tugasnya
  - Output atur cara
- Kongsikan idea bersama-sama ahli kumpulan anda. Bincangkan cara-cara untuk menambah baik hasil dapatan anda.
- Sediakan satu persembahan dan bentangkan hasil dapatan kumpulan anda di dalam kelas.

## Standard Pembelajaran

Murid boleh:

- 4.2.3 Menulis pernyataan *function* dan *procedure*

## Tahukah Anda ?

Parameter digunakan untuk:

1. Menerima dan menyimpan data yang dihantar kepadanya semasa dipanggil
2. Memulangkan data kepada pernyataan arahan, *function*, *procedure* atau atur cara lain yang memanggilnya

4.2.3

## Menulis Pernyataan *Function* dan *Procedure*

### »» Function

Penulisan *user-defined function* bermula dengan mentakrifkan *function* tersebut di permulaan atur cara. Selepas itu, pengguna boleh memanggil fungsi tersebut setiap kali tugasan khas *function* perlu dilaksanakan dalam atur cara. Berikut menunjukkan sintaks pentakrifian suatu *user-defined function*.

```
def nama_function (parameter):  
    badan function  
    return (nilai)
```

Kata kunci **def** digunakan untuk mentakrifkan *function*. Nama sesuatu *function* harus mencerminkan tugas khas yang akan dilakukannya. Parameter ialah boleh ubah untuk menerima argumen (nilai) yang dihantar kepada *function* semasa dipanggil. Parameter dimasukkan dalam tanda kurungan ( ). Pentakrifian nama *function* dan parameter mesti diakhiri dengan tanda titik bertindih.

Badan *function* dimulakan sebaris selepas nama *function* dan perlu inden dari jidar kiri. Badan *function* terdiri daripada set arahan (*instructions*) bagi tugasnya. Perkataan **return** digunakan untuk memulangkan nilai atau output yang dihasilkan oleh *function* tersebut.

Dalam kebanyakan bahasa pengaturcaraan, pernyataan **return** dalam *function* memulangkan satu nilai sahaja. Nilai ini boleh merupakan nilai tunggal atau satu ungkapan yang menghasilkan satu nilai tunggal. Walau bagaimanapun, dalam bahasa pengaturcaraan Python, nilai yang dipulangkan dalam pernyataan **return** *function* boleh berupa satu nilai tunggal atau sekumpulan nilai yang diasingkan dengan tanda koma. Sebagai contoh,

### Situasi 1:

```
def hasilTambah (x,y):  
    jumlah = x + y  
    return (jumlah)
```

Pernyataan **return** memulangkan satu nilai tunggal.

### Situasi 2:

```
def hasilTambah (x,y):  
    return (x + y)
```

Pernyataan **return** memulangkan satu nilai tunggal dari ungkapan yang digunakan.

### Situasi 3:

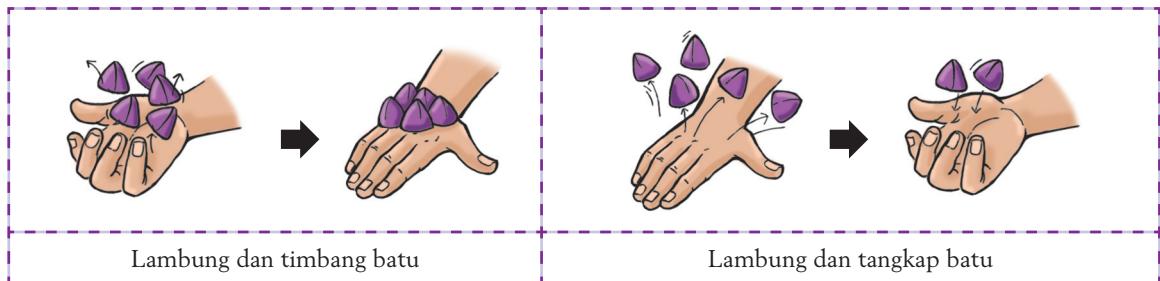
```
def susun_naih (x, y):  
    if x > y:  
        return (y, x)  
    else:  
        return (x, y)
```

Pernyataan **return** memulangkan lebih daripada satu nilai. Pernyataan **return** ini memulangkan dua nilai dalam susunan menaik.

**Contoh 4.11**

Penggunaan *user-defined function* bagi simulasi permainan batu seremban.

Batu seremban ialah sejenis permainan tradisional yang dimainkan di Malaysia. Dalam permainan ini, seorang pemain bermula dengan peringkat yang mudah sehingga ke peringkat terakhir yang disebut buah lapan. Kemudian, barulah pemain melakukan aktiviti timbang dan tangkap untuk mendapatkan mata.



**Rajah 4.52** Cara lambung dan timbang dan lambung dan tangkap batu

Dalam Contoh 4.11, simulasi aktiviti timbang dan tangkap dalam permainan boleh diwujudkan dan nama *user-defined function* yang akan dibina dinamakan `bilangan_batu()`. *User-defined function* ini akan menerima satu nilai yang mewakili bilangan batu yang akan digunakan dalam permainan, iaitu lima. Nilai ini akan dihantar ke *sub-function* bagi *built-in function* `random()`, iaitu `random.randint()`. Nilai yang dipulangkan kali pertama adalah bilangan batu yang berjaya ditimbang. Selepas itu, nilai ini dihantar ke *function random.randint()* semula dan nilai yang dipulangkan kali kedua mewakili bilangan batu yang berjaya ditangkap.

```
Batu Seremban.py - C:\Documents\Tutorial Python\Batu Seremban.py (3.6.5)
File Edit Format Run Options Window Help
import random

# User-defined function untuk simulasi menimbang batu
def bilangan_batu(x):
    hasil =(random.randint(0,x))
    return hasil

# Membuat panggilan fungsi.
# Hantar nilai bilangan batu yang digunakan ke dalam bilangan_batu()
terus = "Y"
while terus == "Y":
    hasil_timbang = bilangan_batu(5)
    print("Hasil timbang anda ialah " + str(hasil_timbang) + " batu.")

# Menangkap batu. Membuat panggilan fungsi.
# Hantar hasil timbang ke dalam bilangan_batu()
if hasil_timbang > 0:
    tangkap = bilangan_batu(hasil_timbang)
    print("Hasil tangkapan anda ialah " + str(tangkap))
terus = input("\nTeruskan [Y] atau Berhenti [T]? Tekan [Y|T] ")
print("")
```

Gelung while digunakan supaya aktiviti timbang dan tangkap dapat diteruskan sehingga pemain menamatkan permainan.

Sekiranya batu berjaya ditimbang, barulah *user-defined function bilangan\_batu()* dipanggil semula. Kali ini argumen yang dihantar ialah `hasil_timbang` dan nilai yang dipulangkan ialah bilangan batu yang berjaya ditangkap.

**Rajah 4.53** Kod atur cara Contoh 4.11

Panggilan *function* `bilangan_batu` diletakkan dalam gelung ulangan *while* supaya aktiviti timbang dan tangkap batu dapat dilakukan secara berulangan.



Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

```
===== RESTART: C:\Documents\Tutorial Python\Batu Seremban.py =====
Hasil timbangan anda ialah 1 batu.
Hasil tangkapan anda ialah 1

Teruskan [Y] atau Berhenti [T]? Tekan [Y|T] Y

Hasil timbangan anda ialah 2 batu.
Hasil tangkapan anda ialah 1

Teruskan [Y] atau Berhenti [T]? Tekan [Y|T] T
```

>>>

Ln: 25 Col:4

Rajah 4.54 Output bagi kod atur cara Contoh 4.11



## Aktiviti 4.9



Aktiviti Kumpulan

### Menulis Pernyataan *Function*



Kaedah Brainstorming Activity

Corak perbelanjaan murid sekolah di pesta buku:

Nama murid	RM		
	Buku cerita	Buku latihan	Buku rujukan
Aini	12	15	33
Aru	10	12	15
Lee	15	18	0

Anda ialah ahli jawatankuasa penganjur pesta buku. Anda ditugaskan untuk mendapatkan data-data mengenai:

- Jumlah bilangan buku yang dibeli oleh murid sekolah;
  - secara keseluruhan.
  - mengikut jenis bahan.
- Jenis bahan bacaan yang paling disukai oleh murid sekolah.

- Anda diberi masa untuk:
  - Mengkaji situasi di atas dan meleraikan masalah
  - Menyatakan tugas khas yang telah dikenal pasti
  - Menulis pernyataan *function* bagi melakukan tugas khas yang telah dikenal pasti
- Selepas tamat tempoh yang diberikan, anda dikehendaki berkumpul dalam kumpulan. Kongsi dan bincangkan hasil dapatan masing-masing bersama-sama rakan sekumpulan.
- Gunakan bahasa pengaturcaraan Python untuk mengekod dan menguji pernyataan *function* yang dihasilkan.



## » Procedure

Sintaks untuk menulis *procedure* hampir serupa dengan sintaks pentakrifan *user-defined function*, cuma *procedure* tidak memulangkan sebarang nilai (*output*) kembali kepada atur cara atau pernyataan kod yang memanggilnya. Badan *procedure* hanya terdiri daripada set arahan bagi tugas khas.

```
def nama_procedure (parameter):
    badan procedure
```

Kata kunci **def** digunakan untuk mentakrifkan *procedure*. Nama sesuatu *procedure* harus mencerminkan tugas khas yang akan dilakukannya. Jika mempunyai input atau parameter, maka parameter perlu dinyatakan dalam tanda kurungan ( ). Jika tidak, tanda kurungan dibiarkan kosong. Pentakrifan nama *procedure* dengan atau tanpa parameter mesti diakhiri dengan tanda titik bertindih.

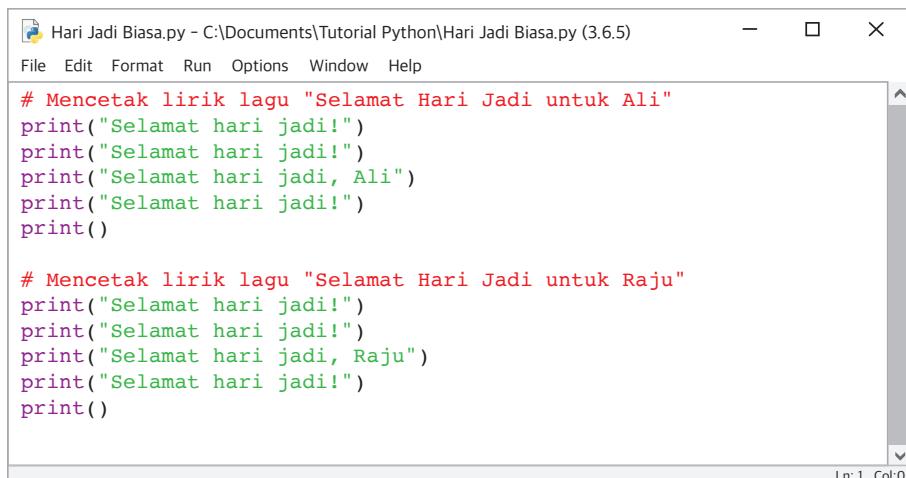
Badan *procedure* dimulakan sebaris selepas nama *procedure* dan perlu inden dari jidar kiri. Badan *procedure* terdiri daripada set arahan bagi melaksanakan tugasnya.

Setelah *procedure* siap dilaksanakan, pelaksanaan atur cara akan kembali semula ke baris selepas pemanggilan *procedure*.

### Contoh 4.12

Penggunaan *procedure* untuk mencetak lirik lagu menggunakan bahasa pengaturcaraan Python.

Aimah ialah seorang guru tadika dan kelasnya mempunyai 15 orang murid. Setiap bulan, Aimah akan menyediakan lirik lagu "Selamat Hari Jadi" untuk murid-muridnya. Katakan dua orang murid menyambut hari jadi pada bulan ini. Berikut menunjukkan satu segmen kod atur cara yang akan mencetak lirik lagu "Selamat Hari Jadi" yang bernama tanpa menggunakan *procedure*.



```
Hari Jadi Biasa.py - C:\Documents\Tutorial Python\Hari Jadi Biasa.py (3.6.5)
File Edit Format Run Options Window Help
# Mencetak lirik lagu "Selamat Hari Jadi untuk Ali"
print("Selamat hari jadi!")
print("Selamat hari jadi!")
print("Selamat hari jadi, Ali")
print("Selamat hari jadi!")
print()

# Mencetak lirik lagu "Selamat Hari Jadi untuk Raju"
print("Selamat hari jadi!")
print("Selamat hari jadi!")
print("Selamat hari jadi, Raju")
print("Selamat hari jadi!")
print()

Ln: 1 Col:0
```

Rajah 4.55 Kod atur cara Contoh 4.12

### Procedure

dalam bahasa pengaturcaraan Python



[pynewbs.com/7a/a](http://pynewbs.com/7a/a)





```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:\Documents\Tutorial Python\Hari Jadi Biasa.py =====
Selamat hari jadi!
Selamat hari jadi!
Selamat hari jadi, Ali
Selamat hari jadi!

Selamat hari jadi!
Selamat hari jadi!
Selamat hari jadi, Raju
Selamat hari jadi!

>>>
Ln: 15 Col:4
```

**Rajah 4.56** Output bagi kod atur cara Contoh 4.12

Aimah mendapati lirik lagu tersebut adalah sama dan berulangan. Bayangkan jika Aimah ingin mencetak lirik lagu untuk semua muridnya.

Penggunaan *procedure* lirik( ) dengan parameter boleh digunakan supaya seluruh lirik tidak perlu ditaip berulang kali. Ini telah menjimatkan masa menaip, kejadian ralat berkurangan dan segmen kod kelihatan lebih kemas. Berikut menunjukkan segmen kod yang telah ditambah baik dengan penggunaan *procedure* untuk mencetak lirik lagu “Selamat Hari Jadi”.

```
Hari Jadi Procedure.py - C:\Documents\Tutorial Python\Hari Jadi Procedure.py (3.6.5)
File Edit Format Run Options Window Help
# Menggunakan procedure untuk lirik lagu
def lirik(nama):
    print("Selamat hari jadi!")
    print("Selamat hari jadi!")
    print("Selamat hari jadi",nama)
    print("Selamat hari jadi!")
    print()

# Mencetak lirik lagu untuk pelajar
lirik("Ali")
lirik("Raju")
Ln: 12 Col:0
```

**Rajah 4.57** Kod atur cara Contoh 4.12 yang menggunakan *procedure*

Setelah penggunaan *procedure*, hanya dua baris kod sahaja diperlukan oleh Aimah bagi mencetak lirik lagu untuk murid-muridnya. Ini memudahkan pencetakan lirik lagu yang seterusnya untuk murid-murid yang lain.



## Aktiviti 4.10



### Aktiviti Kumpulan



Kaedah  
Gallery Walk

#### Menulis Pernyataan *Procedure*



Kaedah  
Gallery Walk

- Kaji segmen kod berikut. Tulis cadangan anda untuk meringkaskan kod di bawah dengan menggunakan *procedure*.

```
Lagu Chan Mali Chan.py - C:\Documents\Tutorial Python\Lagu Chan Mali...
File Edit Format Run Options Window Help
# Lirik Lagu Chan Mali Chan

print("Lirik Lagu Chan Mali Chan\n")

print("Di mana dia anak kambing saya?")
print("Anak kambing saya yang makan daun talas")
print("Di mana dia buah hati saya?")
print("Buah hati saya bagai telur dikupas\n")

print("Chan mali chan, chan mali chan")
print("Chan mali chan, ketipung payung")
print("Chan mali chan, chan mali chan")
print("Chan mali chan, ketipung payung\n")

print("Di mana dia anak kambing saya?")
print("Anak kambing saya yang makan daun talas")
print("Di mana dia buah hati saya?")
print("Buah hati saya bagai telur dikupas\n")

print("Chan mali chan, chan mali chan")
print("Chan mali chan, ketipung payung")
print("Chan mali chan, chan mali chan")
print("Chan mali chan, ketipung payung\n")
```

- □ ×

[goo.gl/DzbwE3](http://goo.gl/DzbwE3)



- Pamerkan hasil kerja anda pada dinding kelas untuk dilihat oleh kumpulan lain.
- Murid-murid digalakkan untuk menulis komen tentang hasil kerja kumpulan lain dan menampatkannya di atas hasil kerja tersebut.
- Guru akan memilih hasil kerja yang terbaik.
- Gunakan *procedure* yang serupa, sediakan satu segmen kod Python untuk lagu negeri anda. Anda digalakkan untuk menguji segmen kod anda menggunakan perisian aplikasi Python.



## Standard Pembelajaran

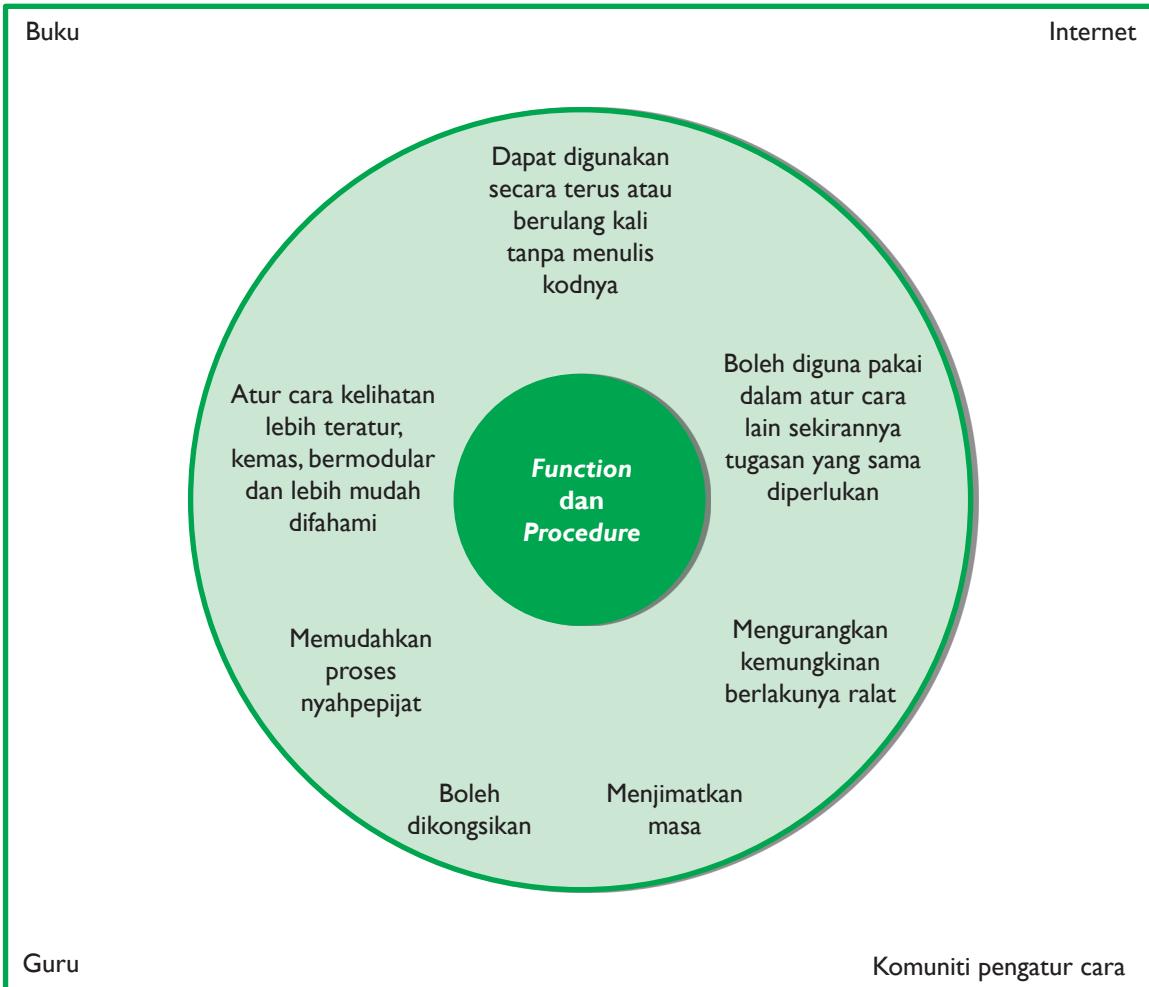
Murid boleh:

- 4.2.4 Menghasilkan atur cara yang melibatkan:
- (i) *function*
  - (ii) *procedure*

4.2.4

## Menulis Atur Cara yang Melibatkan Function dan Procedure

Anda telah mempelajari cara menulis pernyataan *function* dan *procedure* di subtopik 4.2.3. Penggunaan *function* dan *procedure* semasa penulisan kod untuk satu atur cara amat penting dan wajib. Rajah 4.58 menunjukkan kepentingan penggunaan *function* dan *procedure*.



Rajah 4.58 Kepentingan *function* dan *procedure* dalam penghasilan atur cara

**Contoh 4.13**

Menghasilkan satu atur cara kalkulator yang melibatkan gabungan *function* dan *procedure*.

Guru meminta anda menghasilkan satu atur cara kalkulator yang dapat melakukan empat operasi asas, iaitu tambah, tolak, darab dan bahagi terhadap dua nombor integer yang dimasukkan oleh pengguna atur cara.

Bagi menyelesaikan masalah ini, anda perlu mengaplikasikan konsep pemikiran komputasional dalam fasa-fasa pembangunan atur cara. Output yang dikehendaki adalah seperti berikut:

**Menu Kalkulator**

1. Tambah
2. Tolak
3. Darab
4. Bahagi
5. Tamat

Pilihan anda [1 hingga 5]: 3

Masukkan nombor pertama : 4

Masukkan nombor kedua : 5

Output:  $4 \times 5 = 20$

Terima kasih kerana menggunakan saya.

**Fasa Analisis Masalah**

1. Mengenal pasti masalah: Membina atur cara yang boleh melaksanakan empat operasi.
2. Mengenal pasti input, proses dan output:
  - (a) Input: Pilihan operasi dari menu kalkulator dan memasukkan dua nombor.
  - (b) Proses: Lakukan operasi yang dipilih.
  - (c) Output: Paparkan jawapan bagi dua nombor yang dimasukkan berdasarkan operasi yang dipilih.

**Fasa Reka Bentuk Atur Cara**

1. Membangunkan algoritma bagi tugas-tugas kecil (*sub-task*).
2. Menulis pseudokod.
3. Melakar carta alir.
4. Mereka bentuk antara muka pengguna untuk input dan output data.

**Imbas Kembali**

Fasa-fasa pembangunan atur cara.

Analisis Masalah

Reka Bentuk Atur Cara

Pengekodan

Pengujian dan Penyahpejitan

Dokumentasi

**Uji Minda 4.18**

Apakah teknik-teknik pemikiran komputasional yang digunakan semasa fasa analisis masalah?

## Pseudokod

```
1 Mula
2 Isytihar pemboleh ubah aktif, pilihan, nombor1, nombor2, hasil
3 Setkan aktif = 1
4 Selagi aktif == 1
    4.1 Papar menu berikut:
        Kalkulator Bermenu
        1. Tambah
        2. Tolak
        3. Darab
        4. Bahagi
        5. Tamat
    4.2 Papar mesej "Pilihan anda [1 hingga 5]:"
    4.3 Setkan pilihan = nombor yang dimasukkan
    4.4 Jika pilihan == 1
        4.4.1 Papar mesej "Masukkan nombor pertama:"
        4.4.2 Setkan pertama = nombor yang dimasukkan
        4.4.3 Papar mesej "Masukkan nombor kedua:"
        4.4.4 Setkan kedua = nombor yang dimasukkan
        4.4.5 Kira hasil = pertama + kedua
        4.4.6 Papar "Output: nilai pertama + nilai kedua = hasil"
    4.5 Jika pilihan == 2
        4.5.1 Papar mesej "Masukkan nombor pertama:"
        4.5.2 Setkan pertama = nombor yang dimasukkan
        4.5.3 Papar mesej "Masukkan nombor kedua:"
        4.5.4 Setkan kedua = nombor yang dimasukkan
        4.5.5 Kira hasil = pertama - kedua
        4.5.6 Papar "Output: nilai pertama - nilai kedua = hasil"
    4.6 Jika pilihan == 3
        4.6.1 Papar mesej "Masukkan nombor pertama:"
        4.6.2 Setkan pertama = nombor yang dimasukkan
        4.6.3 Papar mesej "Masukkan nombor kedua:"
        4.6.4 Setkan kedua = nombor yang dimasukkan
        4.6.5 Kira hasil = pertama * kedua
        4.6.6 Papar "Output: nilai pertama x nilai kedua = hasil"
    4.7 Jika pilihan == 4
        4.7.1 Papar mesej "Masukkan nombor pertama:"
        4.7.2 Setkan pertama = nombor yang dimasukkan
        4.7.3 Papar mesej "Masukkan nombor kedua:"
        4.7.4 Setkan kedua = nombor yang dimasukkan
        4.7.5 Kira hasil = pertama / kedua
        4.7.6 Papar "Output: nilai pertama / nilai kedua = hasil"
    4.8 Jika pilihan == 5
        4.8.1 Setkan aktif = 0
5 Papar mesej "Terima kasih kerana menggunakan saya."
6 Tamat
```

### Petunjuk:

Biru – Set pernyataan ini diasingkan dan dijadikan satu subpseudokod dptDuaNombor.

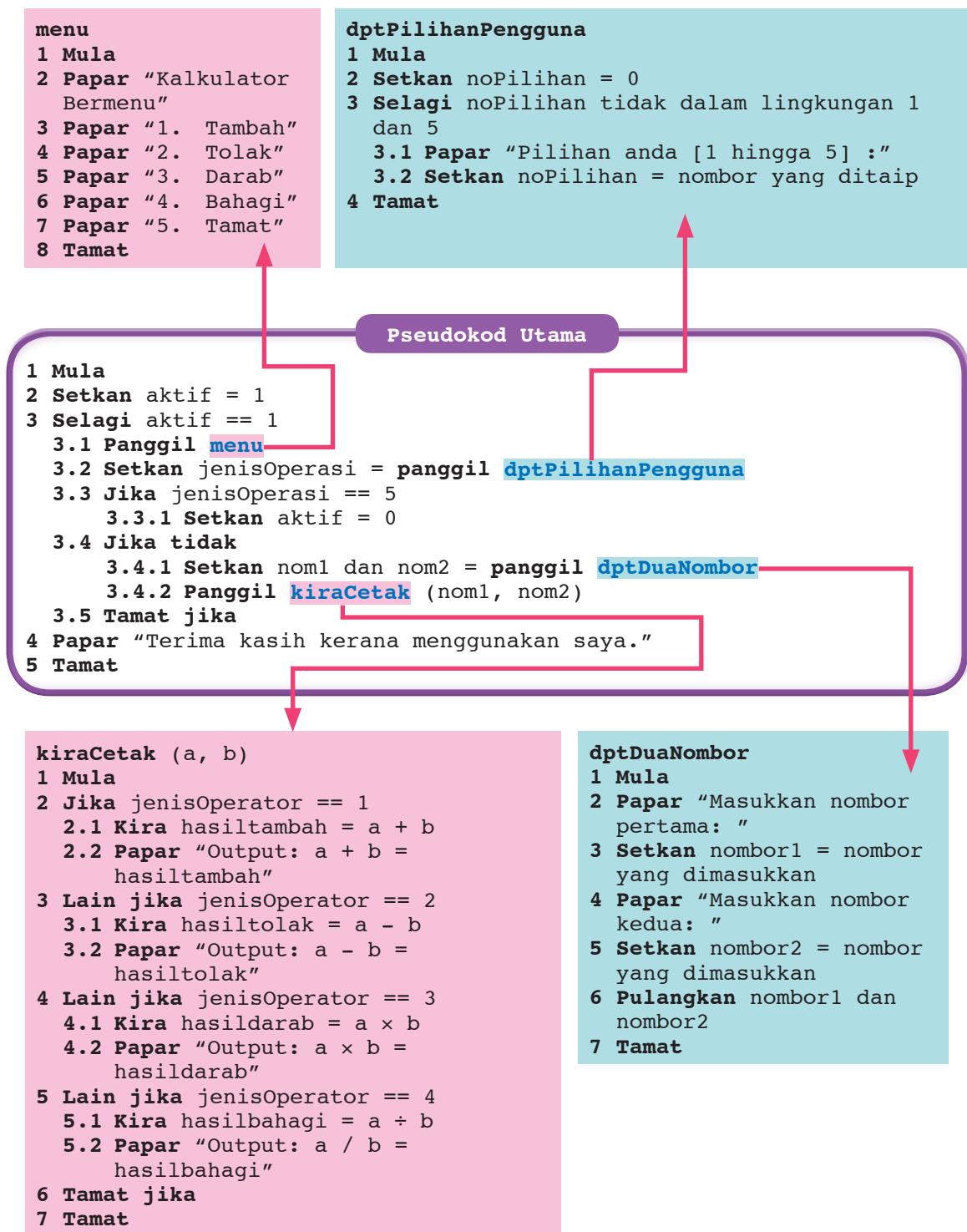
Ungu – Set pernyataan ini diasingkan dan dijadikan satu subpseudokod kiraCetak.

Hijau – Set pernyataan ini diasingkan dan dijadikan satu subpseudokod menu.

Kelabu – Set pernyataan ini diasingkan dan dijadikan satu subpseudokod dptPilihanPengguna.

## Pseudokod yang Dimurnikan Melalui Pengaplikasian Pemikiran Komputasional

Pseudokod berikut adalah lebih tersusun, padat, kemas dan bermódular.





## Yi Minda 4.19

Apakah teknik pemikiran komputasional yang digunakan semasa fasa pengekodan?

### Fasa Pengekodan

Semasa fasa pengekodan, anda perlu mengenal pasti penulisan sintaks-sintaks bagi *function*, *procedure*, input, proses dan output. Selain itu, anda juga perlu mengenal pasti jenis data input yang harus digunakan.



[goo.gl/onbYbT](http://goo.gl/onbYbT)

```

Kalkulator Bermenu_2.py - C:\Documents\Tutorial Python\Kalkulator Bermenu_2.py (3.6.5) - □ ×
File Edit Format Run Options Window Help

# Procedure menu()
def menu():
    print("Kalkulator Bermenu")
    print("1. Tambah")
    print("2. Tolak")
    print("3. Darab")
    print("4. Bahagi")
    print("5. Tamat")

# Function dptPilihanPengguna()
def dptPilihanPengguna():
    noPilihan = 0
    while (noPilihan < 1) or (noPilihan > 5):
        noPilihan = int(input("Pilihan anda [1 hingga 5]: "))
    return noPilihan

# Function dptDuaNombor()
def dptDuaNombor():
    nombor1 = int(input("Masukkan nombor pertama : "))
    nombor2 = int(input("Masukkan nombor kedua : "))
    return nombor1, nombor2

# Procedure kiraCetak()
def kiraCetak(jenisOperator, a, b):
    if jenisOperator == 1:
        print("Output: " + str(a) + " + " + str(b) + " = " + str(a + b) + "\n")
    elif jenisOperator == 2:
        print("Output: " + str(a) + " - " + str(b) + " = " + str(a - b) + "\n")
    elif jenisOperator == 3:
        print("Output: " + str(a) + " * " + str(b) + " = " + str(a * b) + "\n")
    elif jenisOperator == 4:
        print("Output: " + str(a) + " / " + str(b) + " = " + str(a / b) + "\n")

# main -----
aktif = 1
while aktif == 1:
    menu()
    jenisOperasi = dptPilihanPengguna()
    if jenisOperasi == 5:
        aktif = 0
    else:
        [nom1, nom2] = dptDuaNombor()
        kiraCetak(jenisOperasi, nom1, nom2)
print("Terima kasih kerana menggunakan saya.")

#

```

Ln: 47 Col:0

Rajah 4.59 Kod atur cara Contoh 4.13

Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

```
== RESTART: C:\Documents\Tutorial Python\Kalkulator
Bermenu_2.py ==
Kalkulator Bermenu
1. Tambah
2. Tolak
3. Darab
4. Bahagi
5. Tamat
Pilihan anda [1 hingga 5]: 1
Masukkan nombor pertama : 4
Masukkan nombor kedua : 7
Output: 4 + 7 = 11

Kalkulator Bermenu
1. Tambah
2. Tolak
3. Darab
4. Bahagi
5. Tamat
Pilihan anda [1 hingga 5]: 2
Masukkan nombor pertama : 6
Masukkan nombor kedua : 2
Output: 6 - 2 = 4

Kalkulator Bermenu
1. Tambah
2. Tolak
3. Darab
4. Bahagi
5. Tamat
Pilihan anda [1 hingga 5]: 3
Masukkan nombor pertama : 3
Masukkan nombor kedua : 9
Output: 3 * 9 = 27

Kalkulator Bermenu
1. Tambah
2. Tolak
3. Darab
4. Bahagi
5. Tamat
Pilihan anda [1 hingga 5]: 4
Masukkan nombor pertama : 3
Masukkan nombor kedua : 5
Output: 3 / 5 = 0.6

Kalkulator Bermenu
1. Tambah
2. Tolak
3. Darab
4. Bahagi
5. Tamat
Pilihan anda [1 hingga 5]: 5
Terima kasih kerana menggunakan saya.
>>>
```

Ln: 59 Col:4

Rajah 4.60 Output kod atur cara Contoh 4.13

Imbas QR code ini untuk menonton video carta alir bagi atur cara Kalkulator Bermenu.



[goo.gl/otCmBc](http://goo.gl/otCmBc)



### Uji Minda 4.20

- Apakah teknik pemikiran komputasional yang digunakan untuk mengenal pasti corak yang berulangan di dalam pseudokod?
- Apakah teknik pemikiran komputasional yang digunakan untuk mengenal pasti persamaan antara empat pernyataan dalam subpseudokod KiraCetak?



## Aktiviti

4.11



Aktiviti Kumpulan

### Menambahkan Operasi Kuasa dan Punca Kuasa untuk Kalkulator Bermenu

1. Perhatikan Contoh 4.13.
2. Anda perlu menambah operasi kuasa dan punca kuasa n bagi suatu nombor bulat di mana n ialah suatu integer positif.
3. Berdasarkan pseudokod di halaman 184, tambahkan operasi kuasa dan punca kuasa. Tuliskan pseudokod baharu anda di atas kertas.
4. Dalam kumpulan, kongsikan hasil pseudokod masing-masing dan bincangkan bersama-sama untuk menghasilkan satu pseudokod yang terbaik.
5. Muat turun fail **Kalkulator Bermenu\_2.py** dari [goo.gl/onbYbT](http://goo.gl/onbYbT). Kemudian, lancarkan perisian aplikasi Python untuk menambahkan kod yang diperlukan bagi menambahkan operasi kuasa dan punca kuasa. Simpan fail baharu ini dengan nama **Kalkulator Bermenu Baharu.py**.



## Praktis Amali

4.7

### Membangunkan Satu Atur Cara yang Mengandungi *Function* dan *Procedure*



```
*****
Menu Mengira Isi padu
*****
1. Kuboid
2. Silinder
3. Kon
4. Sfera
*****
Masukkan pilihan anda: [1 - 4] : _____
```

Sekiranya pilihan ialah 2, maka paparan berikut akan muncul:  
 Masukkan jejari : 3  
 Masukkan tinggi : 21  
 Isi padu ialah 593.838

#### Imbas Kembali

Isi padu kuboid = panjang × lebar × tinggi  
 Isi padu silinder = luas tapak × tinggi;  
 $luas\ tapak = \pi \times \text{jejari} \times \text{jejari}$   
 $\text{Isi}\ padu\ kon = 1/3 \times \text{luas}\ tapak \times \text{tinggi};$   
 $\text{luas}\ tapak = \pi \times \text{jejari} \times \text{jejari}$   
 $\text{Isi}\ padu\ sfera = 4/3 \times \pi \times \text{jejari} \times \text{jejari} \times \text{jejari};$   
 $\pi$  ialah  $22/7$  atau  $3.142$

Sekiranya pilihan ialah 4, maka paparan berikut akan muncul:  
 Masukkan jejari : 3  
 Isi padu ialah 113.112

Bangunkan satu atur cara yang mengira isi padu kuboid, silinder, kon dan sfera dengan menggunakan langkah-langkah di bawah:

1. Tuliskan satu *procedure* untuk mencetak Menu Mengira Isi padu.
2. Bagi setiap bentuk geometri, tuliskan satu *function* khusus untuk mengira isi padunya.
3. Tuliskan satu atur cara utama yang mengabungkan *procedure* dan semua *function* yang telah dibina untuk membolehkan pengguna mengira isi padu bentuk geometri pilihannya.
4. Cadangkan cara bagaimana anda memurnikan atur cara yang dibina dalam langkah 3 agar atur cara itu boleh digunakan berkali-kali sehingga pengguna memilih untuk menamatkannya.

4.2.5

## Menguji Atur Cara dan Membaiki Ralat

Menguji dan membaiki ralat ialah satu fasa penting dalam pembangunan atur cara. Atur cara yang baik harus bebas ralat. Ralat pengaturcaraan dinamakan pepijat (*bugs*) dan proses menjelaki ralat ini disebut sebagai nyahpepijat (*debugging*). Ralat dalam pengaturcaraan boleh dibahagikan kepada tiga kategori, iaitu ralat sintaks (*syntax error*), ralat masa larian (*runtime error*) dan ralat logik (*logic error*).

### Standard Pembelajaran

Murid boleh:

- 4.2.5 Menguji atur cara dan membaiki ralat pada atur cara yang dihasilkan.



### Uji Minda 4.21

Nyatakan kaedah-kaedah yang telah anda gunakan semasa di Tingkatan 1 dan Tingkatan 2 untuk menguji atur cara.



## Ralat sintaks

Setiap bahasa pengaturcaraan mempunyai set tatabahasa, hukum dan peraturan yang mengawal cara penggunaan bahasa tersebut. Kesemua ini dikenali sebagai sintaks kepada bahasa pengaturcaraan. Sintaks yang difahami oleh penterjemah C dan Python adalah berlainan.

Jadual 4.12 menunjukkan format penulisan sintaks bagi struktur pilihan kedua-dua bahasa C dan Python yang berlainan. Struktur pilihan bahasa Python mempunyai kata kunci *elif* yang hanya boleh difahami oleh penterjemah Python dan setiap kata kuncinya diakhiri dengan tanda ‘:’. Selain itu, bahasa Python tidak menggunakan simbol { } walaupun blok arahan mengandungi lebih daripada satu pernyataan dan setiap arahan tidak diakhiri dengan tanda ‘:’. Hanya setiap baris yang bermula dengan kata kunci diakhiri dengan tanda ‘:’.

**Jadual 4.12** Sintaks bagi keratan arur cara dalam bahasa C dan Python

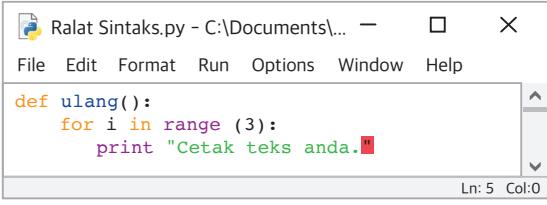
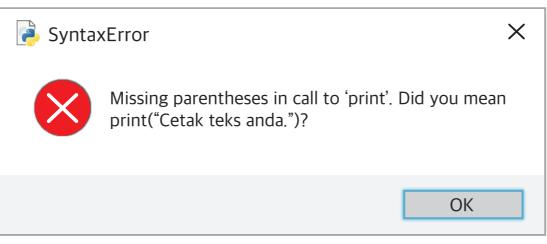
Bahasa C	Bahasa Python
if (a== b) { pernyataan1; pernyataan2; } else if (a > b) pernyataan3; else pernyataan4;	if (a== b) : pernyataan1 pernyataan2 elif (a > b) : pernyataan3 else : pernyataan4

Sekiranya satu baris kod arahan yang ditulis itu tidak menepati laras bahasa pengaturcaraan yang digunakan, maka baris kod arahan itu bukan sahaja tidak dapat difahami oleh penterjemah bahasa tersebut bahkan tidak dapat dilaksanakan. Ralat ini dikenali sebagai **ralat sintaks**. Ralat ini sering berlaku kerana kecuaian pengatur cara semasa menaip kod arahan. Jujukan pelaksanaan (*execution*) arahan dalam *function* atau *procedure* akan terhenti pada baris kod arahan yang mempunyai ralat dan mesej ralat akan dipaparkan.

### Contoh 4.14 Pengesahan ralat sintaks dalam *procedure* oleh penterjemah Python.

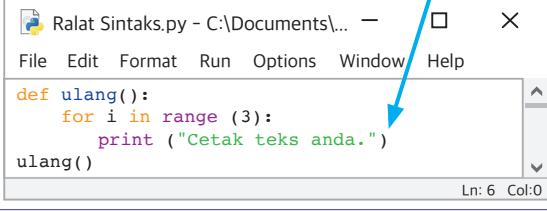
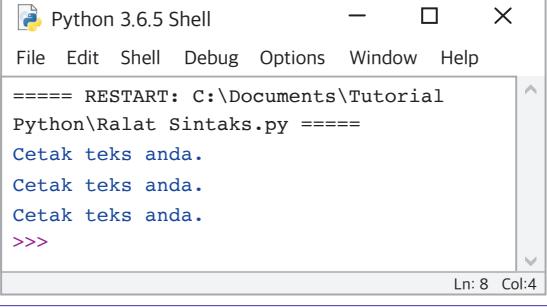
Jadual 4.13 menunjukkan kod *procedure* ulang( ) yang mempunyai ralat sintaks. Pelaksanaan arahan dalam *procedure* ini akan terhenti pada penghujung baris yang ditandakan dengan ” oleh penterjemah Python. Tetingkap *SyntaxError* akan dipaparkan dengan penjelasan ralat yang berlaku dan cadangan membaikinya.

**Jadual 4.13**

Paparan kod <i>procedure</i> yang mempunyai ralat sintaks	Paparan mesej ralat sintaks
 Ralat Sintaks.py - C:\Documents\... — □ × File Edit Format Run Options Window Help <pre>def ulang():     for i in range (3):         print "Cetak teks anda." Ln: 5 Col:0</pre>	 SyntaxError Missing parentheses in call to 'print'. Did you mean print("Cetak teks anda.")? OK

Selepas ralat sintaks ini dibaiki, pelaksanaan arahan dalam *procedure* ini dapat disempurnakan oleh penterjemah Python. Kod arahan yang telah dibaiki dan outputnya seperti yang ditunjukkan dalam Jadual 4.14.

**Jadual 4.14**

Paparan kod <i>procedure</i> ralat sintaks yang telah diperbaiki	Paparan output bagi kod <i>procedure</i>
<p>Masukkan tanda kurungan.</p>  <pre>Ralat Sintaks.py - C:\Documents\... ━ ─ ┗ × File Edit Format Run Options Window Help def ulang():     for i in range (3):         print ("Cetak teks anda.") ulang()  Ln: 6 Col:0</pre>	 <pre>Python 3.6.5 Shell ━ ─ ┗ × File Edit Shell Debug Options Window Help ===== RESTART: C:\Documents\Tutorial Python\Ralat Sintaks.py ===== Cetak teks anda. Cetak teks anda. Cetak teks anda. &gt;&gt;&gt;  Ln: 8 Col:4</pre>

### Contoh 4.15 Cara menguji suatu *function*.

Ramli dan Kien Tien telah ditugaskan untuk menguji satu *function*. Jadual 4.15 menunjukkan cara yang digunakan oleh mereka. Terdapat beberapa cara untuk menguji *function*. Cara yang digunakan ini bergantung kepada tahap penguasaan bahasa pengaturcaraan yang dipelajari dan kreativiti pengatur cara. Dalam buku ini, dua cara yang asas akan dibincangkan.

**Jadual 4.15**

Ramli	Kien Tien	Output
<pre>def kuasadua(x):     return x**2 hasil=kuasadua(4) print(hasil)</pre>	<pre>def kuasadua(x):     return x**2 print(kuasadua(4))</pre>	16

Ramli menggunakan cara yang mengumpulkan nilai yang dipulangkan oleh *function* kuasadua(x) kepada satu pemboleh ubah. Selepas itu, Ramli menggunakan *built-in function* print() untuk mencetak nilai pemboleh ubah tersebut.

Kien Tien pula menggunakan *built-in function* print() untuk terus mencetak nilai yang dipulangkan oleh *function* kuasadua(x).

Kedua-dua kaedah yang digunakan memberikan output yang betul.

### Uji Minda 4.22

Berdasarkan contoh 4.15, jawab soalan-soalan berikut:

- Dapatkan anda mengenal pasti cara penulisan murid yang manakah lebih bagus?
- Apakah kekurangan bagi cara yang bukan pilihan anda?
- Bincangkan kelebihan dan kekurangan cara yang digunakan oleh Ramli dan Kien Tien untuk menguji *function*.

## Contoh 4.16

Menyemak kod arahan dan membaiki ralat.

Atur cara berikut ditulis oleh seorang murid untuk mengira isi padu sebuah piramid bertapak segi empat sama. Pelaksanaan atur cara ini tidak menghasilkan output yang sepatutnya. Jadual 4.16 menunjukkan kod atur cara dan paparan output yang diperoleh.

Jadual 4.16 Kod atur cara dan paparan output yang diperoleh

Kod atur cara
<pre># Mengira isi padu piramid bertapak segi empat sama def kira_Isipadu_piramid(a,b):     isipadu_piramid = (1/3) * (sisi * sisi) * tinggi     return(isipadu_piramid)  # Atur cara utama print("Kira Isi Padu Piramid") sisi = int(input("Masukkan ukuran sisi tapak piramid:")) tinggi = int(input("Masukkan tinggi piramid:"))  # Pemanggilan function dan pemulangan nilai print("Isi Padu Piramid =",kira_Isipadu_piramid(a,b))</pre>
Paparan output dan mesej ralat
<pre>Kira Isi Padu Piramid Masukkan ukuran sisi tapak piramid: 4 Masukkan tinggi piramid: 3 Traceback (most recent call last):   File "C:/Documents/Tutorial Python/kira_Isipadu_piramid.py", line 12, in &lt;module&gt;     print("Isi Padu Piramid =",kira_Isipadu_piramid(a,b)) NameError: name 'a' is not defined &gt;&gt;&gt;</pre>

Semakan ke atas kod atur cara harus dibuat secara tertib.

### Tahukah Anda ?

Apabila *function* dan *procedure* digunakan dalam pembangunan sesuatu atur cara, pendekatan yang diguna pakai dalam proses nyahpepijatan (*debugging*) adalah berlainan sedikit. Teknik *step through* tidak digunakan sepenuhnya kerana *function* dan *procedure* adalah subatur cara yang bersifat modular yang tidak bersandarkan kepada atur cara utama. Oleh demikian, setiap subatur cara ini perlu diuji dan dibaiki ralatnya sehingga bebas ralat sebelum subatur cara ini boleh diguna pakai dalam atur cara utama.

**Langkah 1****Pengujian kod *function* kira\_Isipadu\_piramid()**

*Function* kira\_Isipadu\_piramid( ) harus diuji dahulu dengan menggunakan *function* print( ). Panggilan *function* kira\_Isipadu\_piramid( ) boleh dibuat dalam *function* print( ) dengan meletakkan dua argumen (nilai) semasa panggilan *function* dilakukan. Kod *function* diuji dan didapati mengandungi ralat seperti yang ditunjukkan dalam Jadual 4.17.

**Jadual 4.17** Kod *function* yang mempunyai ralat dan paparan output yang diperoleh

Pengujian Kod <i>function</i>
<pre># Mengira isi padu piramid bertapak segi empat sama def kira_Isipadu_piramid(a,b):     isipadu_piramid = (1/3) * (sisi * sisi) * tinggi     return(isipadu_piramid)  # Atur cara utama print("Kira Isi Padu Piramid") sisi = int(input("Masukkan ukuran sisi tapak piramid:")) tinggi = int(input("Masukkan tinggi piramid:"))  # Pemanggilan function dan pemulangan nilai print("Isi Padu Piramid =",kira_Isipadu_piramid(a,b))</pre>
<b>Uji <i>function</i> kira_Isipadu_piramid( ) dengan <i>function</i> print( )</b> <code>print(kira_Isipadu_piramid(2,3))</code>
<b>Mesej Ralat</b> <pre>Traceback (most recent call last):   File "C:/Documents/Tutorial Python/function kira_Isipadu_piramid.py", line 6, in &lt;module&gt;     print(kira_Isipadu_piramid(2,3))   File "C:/Documents/Tutorial Python/function kira_Isipadu_piramid.py", line 3, in kira_Isipadu_piramid     isipadu_piramid = (1/3) * (sisi * sisi) * tinggi NameError: name 'sisi' is not defined</pre>

Merujuk kepada mesej ralat di atas, didapati pemboleh ubah bernama sisi tidak ditakrifkan.


**Uji Minda 4.23**

1. Berdasarkan penelitian anda, mengapakah hanya dua argumen (nilai) digunakan semasa panggilan *function* kira\_Isipadu\_piramid dibuat?
2. Pada pendapat anda, mengapakah nilai 2 dan 3 telah digunakan sebagai argumen?



## Uji Minda 4.24

Bagaimakah anda dapat mengesahkan bahawa ketiga-tiga pernyataan dalam atur cara utama ini bebas ralat?

### Langkah 2

### Baiki ralat yang dijumpai

Gantikan pemboleh ubah a dan b dalam parameter *function* masing-masing dengan pemboleh ubah sisi dan tinggi. Uji semula *function* `kira_Isipadu_piramid()`.

Jadual 4.18 Baiki ralat yang dijumpai

#### Baiki ralat yang dijumpai dalam kod *function*

```
# Mengira isi padu piramid bertapak segi empat sama
def kira_Isipadu_piramid(sisi,tinggi):
    isipadu_piramid = (1/3) * (sisi * sisi) * tinggi
    return(isipadu_piramid)

# Atur cara utama
print("Kira Isi Padu Piramid")
sisi = int(input("Masukkan ukuran sisi tapak piramid: "))
tinggi = int(input("Masukkan tinggi piramid:"))

# Pemanggilan function dan pemulangan nilai
print("Isi Padu Piramid =",kira_Isipadu_piramid(a,b))
```

Gantikan pemboleh ubah a dan b dalam parameter *function* masing-masing kepada sisi dan tinggi.

**Uji semula *function* `kira_Isipadu_piramid()` dengan *function* `print()`**  
`print(kira_Isipadu_piramid(2,3))`

**Output**  
4.0  
>>>

Hasil pengujian menunjukkan *function* `kira_Isipadu_piramid()` kini telah bebas ralat.

### Langkah 3

### Uji atur cara utama

Tiga pernyataan dalam atur cara utama diuji untuk mencari ralat lain.

```
# Atur cara utama
print("Kira Isi Padu Piramid")
sisi = int(input("Masukkan ukuran sisi tapak piramid: "))
tinggi = int(input("Masukkan tinggi piramid: "))
```

Tiada ralat dijumpai.

### Langkah 4

### Uji pernyataan yang memanggil *function*

Semasa panggilan *function* `kira_Isipadu_piramid()`, dua argumen diperlukan. Dua argumen ini ialah nilai yang disimpan dalam pemboleh ubah sisi dan tinggi. Maka a dan b masing-masing harus digantikan dengan pemboleh ubah sisi dan tinggi.

```
print("Isi Padu Piramid =",kira_Isipadu_piramid(sisi,tinggi))
```

Masukkan argumen (nilai sebenar) yang betul semasa memanggil *function*.

## Ralat masa larian

Ralat masa larian akan timbul atau kelihatan apabila *user-defined function*, *procedure* atau atur cara sedang dilaksanakan (*executed*). Ralat jenis ini akan menyebabkan pelaksanaan kod arahan terhenti secara tiba-tiba dan mesej ralat akan dipaparkan.



Kesilapan umum ialah punca utama kejadian ralat masa larian.

### Contoh 4.17

Pengesanan ralat masa larian dalam *function*.

Contoh ini menunjukkan cara pengesanan ralat masa larian dalam *user-defined function* *kira\_purata*. Semak kod atur cara yang berikut. Penulisan sintaks adalah betul. Akan tetapi, semasa pelaksanaan kod arahan tersebut atur cara terhenti secara tiba-tiba apabila input pertama yang dimasukkan bukan satu angka.

**Jadual 4.19** Kod atur cara dan paparan output yang diperoleh

Kod atur cara
<pre># Function mengira purata def kira_purata(x,y):     purata = jumlah / bilangan     return(purata)  # Atur cara utama senaraiNo = [] cont = True while cont:     try:         nom = float(input("Masukkan nombor [X utk berhenti]: "))         senaraiNo.append(nom)     except ValueError:         cont = False  bilangan = len(senaraiNo) jumlah = sum(senaraiNo) print("Purata = ",str(kira_purata(jumlah,bilangan))) print("Tamat.")</pre>
Mesej ralat terpapar secara tiba-tiba semasa pelaksanaan
<pre>RESTART: C:\Documents\Tutorial Python\Ralat Masa Larian.py Masukkan nombor [X utk berhenti]: 0 Masukkan nombor [X utk berhenti]: X Purata = 0.0 Tamat. &gt;&gt;&gt; RESTART: C:\Documents\Tutorial Python\Ralat Masa Larian.py Masukkan nombor [X utk berhenti]: X Traceback (most recent call last):   File "C:\Documents\Tutorial Python\Ralat Masa Larian.py", line 18, in &lt;module&gt;     print("Purata = ",str(kira_purata(jumlah,bilangan)))   File "C:\Documents\Tutorial Python\Ralat Masa Larian.py", line 3, in kira_purata     purata = jumlah / bilangan ZeroDivisionError: division by zero &gt;&gt;&gt;</pre> <div style="border: 1px solid blue; padding: 5px; margin-left: 20px;"> <p>Gelung <i>while</i> ini akan meminta pengguna memasukkan nombor secara berulang kali. Nombor yang diterima oleh atur cara akan dimasukkan ke dalam <b>senaraiNo</b>. Sekiranya nilai bukan angka dimasukkan, gelung akan berhenti.</p> </div> <div style="border: 1px solid blue; padding: 5px; margin-left: 20px;"> <p>Dalam mesej ralat, baris yang menyebabkan ralat dan punca kejadian ralat akan ditunjukkan.</p> </div>

Punca atur cara terhenti kerana atur cara diminta melakukan pembahagian dengan sifar. Kejadian ini berlaku pada baris kod dalam *function kira\_purata(x,y)*.

```
purata = jumlah / bilangan
```

Ini disebabkan nilai bagi pemboleh ubah bilangan adalah sifar. Nilai ini datang dari baris kod dalam atur cara utama yang berikut.

```
bilangan = len(senaraiNo)
```

Jadual 4.20 menunjukkan kod atur cara yang telah dibaiki dan paparan output yang diperoleh.

**Jadual 4.20** Kod atur cara dan paparan output yang diperoleh

#### Kod atur cara yang dibaiki ralat

```
# Function mengira purata
def kira_purata(x,y):
    purata = jumlah / bilangan
    return(purata)

# Atur cara utama
senaraiNo = []
cont = True
while cont:
    try:
        nom = float(input("Masukkan nombor [X utk berhenti]: "))
        senaraiNo.append(nom)
    except ValueError:
        cont = False
bilangan = len(senaraiNo)

if bilangan > 0:
    jumlah = sum(senaraiNo)
    print("Purata = ",str(kira_purata(jumlah,bilangan)))

print("Tamat.")
```

Gunakan struktur kawalan pilihan untuk membenarkan pengiraan diteruskan hanya apabila bilangan lebih besar daripada sifar.

#### Paparan output

```
RESTART: C:\Documents\Tutorial Python\Ralat Masa Larian 2.py
Masukkan nombor [X utk berhenti]: 0
Masukkan nombor [X utk berhenti]: X
Purata = 0.0
Tamat.
>>>
RESTART: C:\Documents\Tutorial Python\Ralat Masa Larian 2.py
Masukkan nombor [X utk berhenti]: X
Tamat.
>>>
```

## Ralat logik

Ralat logik tidak akan menyebabkan pelaksanaan sesuatu atur cara terhenti secara tiba-tiba dengan atau tanpa memaparkan mesej ralat. Ralat logik akan menyebabkan atur cara melakukan sesuatu tindakan yang tidak dijangka ataupun memberikan output yang tidak sepatutnya. Ralat logik sukar dikesan.

### Contoh 4.18

Pengesanan ralat logik dalam *user-defined function*.

Contoh ini membincangkan cara mengesan ralat logik dalam atur cara yang menggunakan *user-defined function* tambah() dan semak() untuk mencari hasil tambah dua nombor bulat yang dijanakan secara rawak oleh komputer serta menyemak jawapan pengguna.

Atur cara di bawah telah dihasilkan. Apabila dilaksanakan, pelaksanaannya berjaya.

**Jadual 4.21** Kod atur cara dan paparan output yang diperoleh

Kod atur cara yang dibaiki ralat	
<pre># Atur cara ini menggunakan built-in function random import random  # User-defined function tambah() def tambah(a,b):     return(a+b)</pre>	<b>Langkah 1 Uji function tambah()</b> <pre>def tambah(a, b):     return(a + b)  print(tambah(2,3))</pre> <p>Output: 5 Keputusan: Function ini tiada ralat.</p>
<pre># User-defined function semak() def semak(jawapan, jumlah):     if jawapan == jumlah:         print("Betul!")     else:         print("Salah! Jawapannya ialah", jumlah)</pre>	<b>Langkah 2 Uji function semak()</b> <pre>def semak(jawapan, jumlah):     if jawapan == jumlah:         print("Betul!")     else:         print("Salah! Jawapannya ialah", jumlah)  print(semak(2,3))</pre> <p>Output: Betul! Keputusan: Function ini tiada ralat.</p>
<pre># Atur cara utama nombor1 = random.randint(1, 10) nombor2 = random.randint(1, 10) print("Selesaikan", nombor1, "+", nombor2, ".") jawapan = input()</pre>	<b>Langkah 3 Semak jenis data</b> <pre>jawapan = input("Jawapan anda:") print(type(jawapan))</pre> <p>Output: Jawapan anda: 5 &lt;class 'str'&gt; Keputusan: Input mempunyai jenis data str</p>
<pre># Pemanggilan function tambah() dan semak() jumlah = tambah(nombor1,nombor2) semak(jawapan, jumlah)</pre>	

Ralat hanya kelihatan dalam situasi tertentu seperti Jadual 4.22.

Jadual 4.22 Atur cara menghasilkan output yang tidak tepat

Katakan pelaksanaan atur cara kali pertama menghasilkan soalan: Selesaikan $1 + 8$ ?  Sekiranya jawapan yang dimasukkan ialah 10, maka respons atur cara adalah seperti yang dijangkakan.	Output  Selesaikan $1 + 8$ . 10 Salah! Jawapannya ialah 9
Katakan pelaksanaan atur cara kali kedua menghasilkan soalan: Selesaikan $2 + 3$ ?  Sekiranya jawapan yang dimasukkan ialah 5, maka respons atur cara adalah tidak seperti yang dijangkakan.	Output  Selesaikan $2 + 3$ . 5 Salah! Jawapannya ialah 5

Didapati respons yang diberikan oleh atur cara adalah tidak seperti yang dijangkakan. Oleh itu, setiap *function* harus diuji. Berdasarkan Jadual 4.21, Langkah 1 dan 2 telah mengesahkan kedua-dua *function* tambah( ) dan semak( ) adalah bebas ralat. Maka, tinggal kod dalam atur cara utama.

Pernyataan-pernyataan ini tidak melibatkan manipulasi data, maka tidak mungkin menyebabkan ralat.

```
nombor1 = random.randint(1, 10)
nombor2 = random.randint(1, 10)
print("Selesaikan", nombor1, "+", nombor2)
jawapan = input()
```

Berdasarkan keputusan Langkah 3, jenis data bagi boleh ubah jawapan ialah jenis str. Oleh sebab argumen (nilai) ini perlu dibandingkan dengan nilai jumlah yang berjenis int melalui operator ==, maka jenis data bagi boleh ubah jawapan harus diubah kepada int supaya pernyataan **if jawapan == jumlah:** dalam *function* semak() dapat dilakukan dengan betul, iaitu data integer berbanding dengan data string.

Berikut cara membaiki ralat yang ditemui dalam Langkah 3.

```
jawapan = int(input())
```

Nilai yang dimasukkan oleh pengguna ditukarkan kepada jenis data integer.



### Uji Minda 4.25

Pada pendapat anda, mengapa hanya set data ujian {2, 3} dipilih untuk digunakan dalam pengujian ini?



## Aktiviti 4.12

Aktiviti Pasangan

Menguji Atur Cara dan Membaiki Ralat yang Dijumpai

Kaedah  
Think-Pair-Share

Jadual 1 Kod atur cara 1 dan 2

Kod atur cara 1	Kod atur cara 2
<pre>def bilangturun(simbol):     bil_baris = 1     while bil_baris &lt; 10:         print(simbol * bil_baris)         bil_baris = bil_baris + 1  #Atur cara utama aksara=input("Masukkan simbol: ") bilangturun("aksara")</pre>	<pre>def banding(x,y):     if x &gt; y:         besar = x     else:         kecil = x     return (besar,kecil)  #Atur cara utama [besar, kecil] = banding(6,9) print("Nombor ",besar,"lebih besar daripada ",kecil)</pre>

Jadual 2 Contoh output atur cara 1 dan 2

Contoh output atur cara 1	Contoh output atur cara 2
Masukkan simbol: & & && &&& &&&& &&&&& &&&&&& &&&&&&&	Nombor 9 lebih besar daripada 6

- Perhatikan Jadual 1 dan Jadual 2, kemudian jawab soalan-soalan berikut dalam masa yang diperuntukkan.
  - Uji kod-kod atur cara dalam Jadual 1 dengan kaedah-kaedah yang telah anda pelajari. Tuliskan cara kerja anda pada sehelai kertas.
  - Kenal pasti jenis ralat yang dikesan.
  - Baiki ralat yang dikesan supaya output yang diperoleh adalah serupa dengan contoh output yang ditunjukkan.
- Selepas tempoh masa yang diperuntukkan tamat, kongsikan idea bersama-sama rakan anda. Lakukan perbincangan untuk menambah baik hasil dapatan anda.
- Sediakan satu persembahan dan bentangkan hasil dapatan kumpulan anda di dalam kelas.

## Standard Pembelajaran

Murid boleh:

- 4.2.6 Menghasilkan atur cara yang melibatkan gabungan struktur kod arahan bagi menyelesaikan masalah dalam kehidupan seharian

4.2.6

## Menghasilkan Atur Cara yang Melibatkan Gabungan Struktur Kod Arahan bagi Menyelesaikan Masalah

Atur cara yang mempunyai susunan struktur kod arahan yang sistematis, logik dan bermódular dapat memudahkan pengatur cara dan penyenggara atur cara mengenal pasti tugas setiap struktur atau modul. Atur cara yang bermódular dapat dihasilkan dengan penggunaan *function* dan *procedure*.

### Contoh 4.19

Penghasilan atur cara yang menggunakan *function* dan *procedure* bagi mengira peratus keuntungan dan kerugian.

Tarmini ingin membina satu atur cara bagi mengira peratus keuntungan dan kerugian barang yang dijual oleh abangnya. Dalam mata pelajaran Matematik, Tarmini telah mempelajari rumus seperti dalam Jadual 4.23.

Jadual 4.23

Untung = Harga Jualan – Harga Kos	Rugi = Harga Kos – Harga Jualan
$\% \text{ keuntungan} = \frac{\text{Untung}}{\text{Harga Kos}} \times 100$	$\% \text{ kerugian} = \frac{\text{Rugi}}{\text{Harga Kos}} \times 100$

Tarmini akan menggunakan rumus di atas dan mengaplikasikan fasa-fasa pembangunan atur cara yang telah dipelajari dalam Bab 1.

### Fasa Menganalisis Masalah

Gunakan teknik leraian untuk memecahkan masalah mengira peratus keuntungan dan kerugian kepada beberapa bahagian kecil seperti mendapatkan input bagi harga kos dan harga jualan, proses pengiraan bagi peratus keuntungan dan kerugian serta cara memaparkan output yang sepadan. Bahagian-bahagian kecil ini lebih mudah diuruskan.



### Fasa Reka Bentuk Atur Cara

Corak dan prinsip yang menghasilkan corak ini harus dikenal pasti. Analisis rumus matematik bagi pengiraan peratus keuntungan dan peratus kerugian diperlihatkan. Kedua-dua pengiraan ini menggunakan **item input** yang sama, iaitu **harga kos** dan **harga jualan**. Perbezaan antara harga jualan dan harga kos akan menentukan peratus keuntungan, peratus kerugian atau tiada kerugian.



Jadual 4.24

Untung = Harga Jualan – Harga Kos	Rugi = Harga Kos – Harga Jualan
$\% \text{ keuntungan} = \frac{\text{Untung}}{\text{Harga Kos}} \times 100$	$\% \text{ kerugian} = \frac{\text{Rugi}}{\text{Harga Kos}} \times 100$



Berdasarkan maklumat yang diperoleh, kini Tarmini perlu menyediakan algoritma bagi atur cara mengira peratus keuntungan dan kerugian. Kemudian, algoritma ini diterjemahkan ke dalam bentuk pseudokod atau carta alir.

## Algoritma

1. Dapatkan harga kos (**h1**) dan harga jualan (**h2**) daripada pengguna.
2. Bandingkan nilai harga kos dengan harga jualan.
3. Jika harga jualan sama dengan harga kos, papar mesej “Tiada keuntungan”.
4. Jika harga jualan lebih tinggi daripada harga kos, kira dan papar peratus keuntungan.
5. Jika harga kos lebih tinggi daripada harga jualan, kira dan papar peratus kerugian.

## Pseudokod

```

1 Mula
2 Papar "Masukkan harga kos RM: "
3 Setkan h1 = harga kos yang dimasukkan
4 Papar "Masukkan harga jualan RM: "
5 Setkan h2 = harga jualan yang dimasukkan
6 Jika h1 == h2
    6.1 Papar "Tiada keuntungan."
7 Jika h2 > h1
    7.1 Kira peratus keuntungan =  $\frac{h2 - h1}{h1} \times 100$ 
    7.2 Papar peratus keuntungan
8 Jika tidak
    8.1 Kira peratus kerugian =  $\frac{h1 - h2}{h1} \times 100$ 
    8.2 Papar peratus kerugian
9 Tamat jika
10 Tamat

```

Jika algoritma yang dihasilkan dikaji semula, anda akan mendapati bahawa ada proses yang berulangan kerana mempunyai corak yang sama. Sebagai contoh, proses memasukkan data mempunyai corak yang sama, iaitu meminta pengguna memasukkan data sebanyak dua kali:

**Papar** "Masukkan harga kos RM: "  
**h1** = nilai harga kos  
**Papar** "Masukkan harga jualan RM: "  
**h2** = nilai harga jualan



Carta alir bagi mengira peratus keuntungan dan kerugian.



[goo.gl/9QCKDy](http://goo.gl/9QCKDy)



Video carta alir bagi mengira peratus keuntungan dan kerugian dengan menggunakan *function* dan *procedure*.



[goo.gl/J4YoSM](http://goo.gl/J4YoSM)



Begitu juga dengan proses pengiraan, di mana pengiraan peratus keuntungan dan peratus kerugian mempunyai formula yang serupa dan kedua-dua formula tersebut menggunakan data yang sama.

$$\text{Kira peratus keuntungan} = \frac{h2 - h1}{h1} \times 100$$

Papar peratus keuntungan

$$\text{Kira peratus kerugian} = \frac{h1 - h2}{h1} \times 100$$

Papar peratus kerugian

Bagi menjadikan penyelesaian masalah lebih mudah dan bermodular, proses-proses yang berulangan ini, iaitu proses memasukkan data dan proses pengiraan serta pemaparan peratus keuntungan dan kerugian, perlu dijadikan *function* ataupun *procedure*. Berikut menunjukkan pseudokod setelah diubah suai.

### Pseudokod *function* dan *procedure*

*Function inputPengguna* untuk pengguna menginput harga kos dan harga jualan:



#### *Function inputPengguna(mesejInput)*

- 1 Mula
- 2 Papar mesejInput
- 3 Setkan harga = input pengguna
- 4 Pulangkan harga
- 5 Tamat

*Procedure kiraPeratus* untuk mengira dan memaparkan peratus keuntungan atau peratus kerugian:

#### *Procedure kiraPeratus(h1, h2)*

- 1 Mula
- 2 Kira peratus =  $\frac{h2 - h1}{h1} \times 100$
- 3 Jika peratus > 0  
3.1 Papar "Peratus keuntungan =" dan peratus
- 4 Jika tidak  
4.1 Papar "Peratus kerugian =" dan peratus (nilai mutlak)
- 5 Tamat jika
- 6 Tamat

### Pseudokod atur cara utama

```

1 Mula
2 h1 = panggil inputPengguna("Masukkan harga kos RM: ")
3 h2 = panggil inputPengguna ("Masukkan harga jualan RM:")
4 Jika h1 == h2
    4.1 Papar "Tiada keuntungan."
5 Jika tidak
    5.2 Panggil kiraPeratus(h1, h2)
6 Tamat jika
7 Tamat

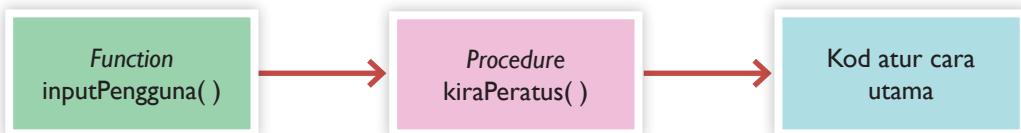
```

### Fasa Pengekodan

Semasa pengekodan, lazimnya *procedure* atau *function* dihasilkan dahulu dan diuji supaya bebas ralat sebelum proses penulisan kod atur cara utama dimulakan. Dalam fasa ini, algoritma diterjemahkan kepada kod arahan bahasa pengaturcaraan yang terpilih. Teknik-teknik pemikiran komputasional masih boleh diaplikasikan mengikut kesesuaian dengan sintaks bahasa pengaturcaraan yang digunakan.



Dalam Contoh 4.19, kod arahan *function* `inputPengguna()` dihasilkan dan diuji supaya bebas ralat sebelum *procedure* `kiraPeratus()` dihasilkan kerana output daripada *function* `inputPengguna()` akan menjadi input kepada *procedure* `kiraPeratus()`. Apabila kod *procedure* `kiraPeratus()` diuji dan didapati bebas ralat, bawarulah penulisan kod atur cara utama dimulakan. Proses ini ditunjukkan dalam peta alir di bawah:



Rajah 4.61 Proses penghasilan atur cara Contoh 4.19

### Langkah 1 Mengekod *function* `inputPengguna`

Anda harus menentukan jenis data input yang diterima oleh parameter, proses dan output bagi *function* ini sebelum mengekod.

```

inputPengguna.py - C:\Documents\Tutorial Python\inputPengguna.py (3.6.5)
File Edit Format Run Options Window Help
# Function inputPengguna
def inputPengguna(mesejInput):
    print(mesejInput)
    harga = float(input())
    return harga
  
```

Ln: 6 Col:0

Rajah 4.62 Kod *function* `inputPengguna`

Selepas siap mengekod *function* *inputPengguna*, kod arahan harus diuji supaya bebas ralat.

Jadual 4.25 Pengujian *inputPengguna()*

Pengujian <i>inputPengguna()</i>	Output
<pre>def inputPengguna(mesejInput):     print(mesejInput)     harga = float(input())     return harga  h1 = inputPengguna ("Masukkan harga kos RM ") print(h1)</pre>	Masukkan harga kos RM 5.45 5.45 Input pengguna

## Langkah 2 Mengekod *procedure* *kiraPeratus*

Argumen yang diterima oleh *procedure* ini ialah data jenis *float*. Hasil kiraan peratus mesti sentiasa suatu nilai yang positif. *Procedure* *kiraPeratus* juga harus diuji.

```
# Procedure kiraPeratus
def kiraPeratus(h1,h2):
    peratus =((h2-h1)/h1)*100
    peratus = round(peratus, 2) # dua tempat perpuluhan
    if peratus > 0:
        print("Keuntungan ialah", peratus, "%")
    else:
        print("Kerugian ialah", abs(peratus), "%") # positifkan nilai
```

Ln: 5 Col:19

Rajah 4.63 Kod *procedure* *kiraPeratus*

Jadual 4.26 Pengujian *kiraPeratus()*

Pengujian <i>kiraPeratus()</i>	Output
<pre># Procedure kiraPeratus def kiraPeratus(h1,h2):     peratus =((h2-h1)/h1)*100     peratus = round(peratus, 2)     if peratus &gt; 0:         print("Keuntungan ialah", peratus,         "%")     else:         print("Kerugian ialah", abs(peratus),         "%")  kiraPeratus(2.50, 3.00) kiraPeratus(1.50, 2.80) kiraPeratus(2.75, 2.55) kiraPeratus(2.50, 2.50)</pre>	Keuntungan ialah 20.0 % Keuntungan ialah 86.67 % Kerugian ialah 7.27 % Kerugian ialah 0.0 %  Jika h1 dan h2 adalah sama, output yang diperoleh tidak tepat.

Anda perlu menguji *procedure* kiraPeratus() dengan pelbagai situasi yang berlainan supaya pengujian yang dilakukan adalah lengkap. Didapati kod *procedure* kiraPeratus() tidak dapat menghasilkan output yang sepatutnya apabila nilai h1 sama dengan nilai h2. Maka, atur cara utama perlu menentukan bahawa argumen (nilai h1 dan h2) yang diberikan kepada *procedure* kiraPeratus() tidak mempunyai nilai yang sama.

### Langkah 3

### Mengekod atur cara utama

Dalam pengekodan atur cara utama, *function* dan *procedure* yang bebas ralat itu harus dimasukkan dan panggilan *function* dan *procedure* boleh dibuat apabila diperlukan.

```

# Function inputPengguna
def inputPengguna(mesejInput):
    print(mesejInput)
    harga = float(input())
    return harga

# Procedure kiraPeratus
def kiraPeratus(h1,h2):
    peratus =((h2-h1)/h1)*100
    peratus = round(peratus, 2) #dua tempat perpuluhan
    if peratus > 0:
        print("Keuntungan ialah", peratus, "%")
    else:
        print("Kerugian ialah", abs(peratus), "%") #positifkan nilai

# Atur cara utama
h1 = inputPengguna("Masukkan harga kos RM ")
h2 = inputPengguna("Masukkan harga jualan RM ")

if h1 == h2:
    print("Tiada keuntungan")
else:
    kiraPeratus(h1, h2)

```

Struktur kawalan ini memastikan nilai-nilai h1 dan h2 yang dihantar kepada *procedure* kiraPeratus() adalah berlainan.

**Rajah 4.64** Kod atur cara utama Contoh 4.19

Selepas atur cara yang lengkap siap dikodkan, kod perlu diuji semula. Rajah 4.65 menunjukkan tiga set data ujian berserta output sepadannya. Output setiap set data ujian ini adalah seperti yang dijangkakan.

```

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
== RESTART: C:\Documents\Tutorial Python\kira_PeratusUntungRugi.py ==
Masukkan harga kos RM
3.50
Masukkan harga jualan RM
3.50
Tiada keuntungan
>>>
== RESTART: C:\Documents\Tutorial Python\kira_PeratusUntungRugi.py ==
Masukkan harga kos RM
2.00
Masukkan harga jualan RM
3.00
Keuntungan ialah 50.0 %
>>>
== RESTART: C:\Documents\Tutorial Python\kira_PeratusUntungRugi.py ==
Masukkan harga kos RM
2.00
Masukkan harga jualan RM
1.50
Kerugian ialah 25.0 %
>>>

```

Ln: 24 Col:4

Rajah 4.65

