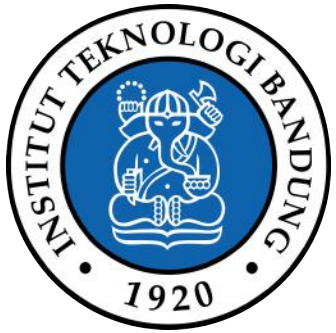
Subprogram (Python)

Tim Penyusun Materi Pengenalan Teknologi Informasi



Tujuan

- Mahasiswa dapat
  - Menjelaskan struktur fungsi/prosedur
  - Membuat algoritma fungsi/prosedur berdasarkan definisi yang diberikan
  - Menggunakan/memanggil fungsi/prosedur dalam program utama

Kode yang berulang

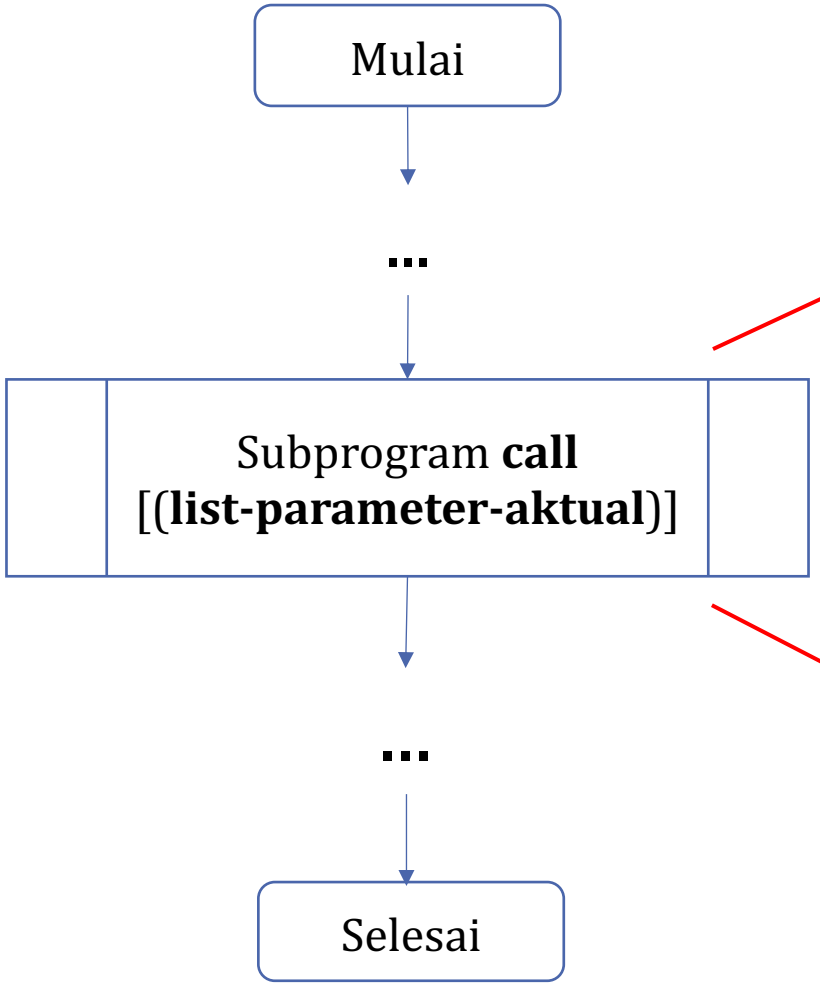
- Semakin besar program, semakin banyak **bagian kode yang berulang**
- Sangat tidak efisien jika bagian kode yang sama/serupa **diketik berulang-ulang**, (bahkan kalau di-*copy-paste*)
- Di samping itu, dalam banyak persoalan, ada berbagai **rumus/formula** yang berulang-ulang dipakai dalam satu program
- Bagaimana jika ada cara supaya bagian kode tersebut tidak perlu diketik berulang-ulang, tapi tetap dapat digunakan berkali-kali dalam program yang sama

Subprogram

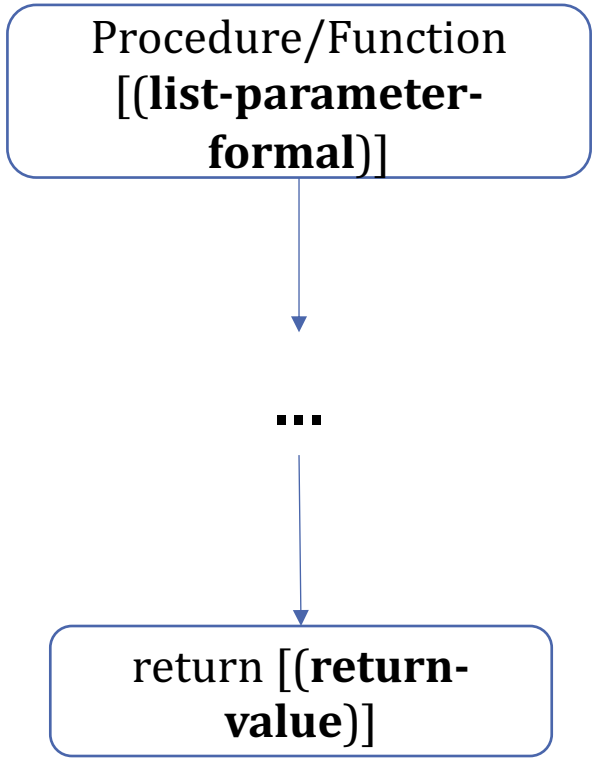
- *A set of instructions designed to perform a frequently used operation within a program*
- 2 (dua) jenis subprogram:
  - **Fungsi**: pemetaan suatu nilai domain (input) ke range (output)
    - Hasil dari fungsi dinyatakan dalam sebuah type data yang eksplisit
  - **Prosedur**: deretan instruksi yang jelas initial state dan final state-nya → mirip seperti program secara umum, namun dalam scope yang lebih kecil

Flowchart Symbol (Umum)

Pemanggilan subprogram dalam algoritma program utama



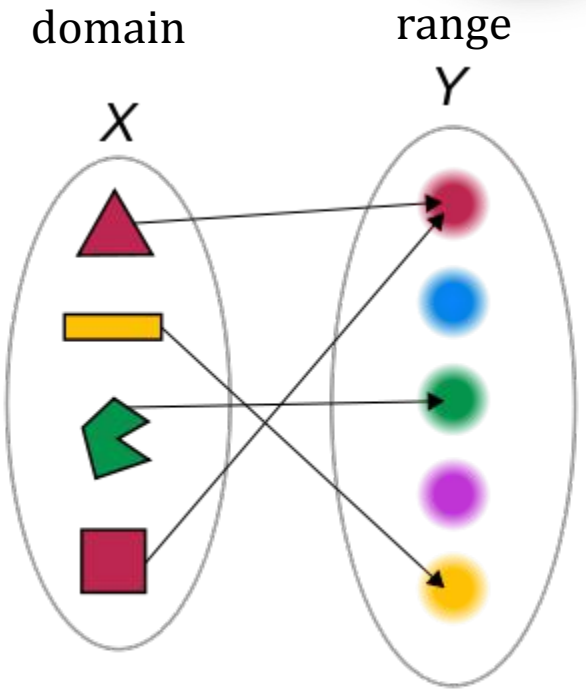
Flowchart terpisah untuk pendefinisian dan realisasi subprogram



Fungsi

Fungsi

- Konsep fungsi di pemrograman didasari oleh konsep pemetaan dan **fungsi** di matematika
- **Fungsi**: asosiasi (pemetaan) antara 2 himpunan nilai yaitu **domain** dan **range**
  - Setiap elemen pada himpunan domain dipetakan **tepat satu** ke sebuah elemen pada himpunan range
- Contoh: $f(X) = X^2$
  - fungsi untuk menghitung kuadrat dari suatu bilanga
  - Domain: bilangan bulat
  - Range: bilangan bulat (0 atau positif)



$f : X \rightarrow Y$

Fungsi dalam Pemrograman

- Memrogram fungsi pada dasarnya adalah: **merakit isi black box**
  - Berangkat dari keadaan awal → himpunan nilai yang terdefinisi sebagai **input** (domain)
  - Menghasilkan nilai-nilai yang mendefinisikan keadaan akhir → himpunan nilai yang terdefinisi sebagai **output** (range)
  - Tugas pemrogram fungsi adalah menentukan langkah-langkah untuk menghasilkan keadaan akhir berdasarkan keadaan awal
- Fungsi didefinisikan sebagai bagian terpisah dari program dan dipanggil dalam **program utama**

Fungsi dalam program

```
# Program Test
# Mengetes fungsi kuadrat

# KAMUS
#     A : integer
#     B : integer

# Fungsi Kuadrat
def Kuadrat ( X ) :
    # menghitung kuadrat X
    hasil = X * X
    return hasil

# ALGORITMA PROGRAM UTAMA
A = 5
B = Kuadrat (A) + 10
```

Function flow of control:

1) Salah satu baris pada kode program utama **memanggil** fungsi: B = **Kuadrat(A)** + 10 # A = 5
2) Program beralih ke kode fungsi **Kuadrat** mulai dari baris yang pertama sampai pada baris yang mendefinisikan hasil fungsi (return). Parameter input diasosiasikan dengan daftar parameter input pada fungsi.

```
def Kuadrat ( X ):
# menghitung kuadrat X
    hasil = X * X
    return hasil
```

Asosiasi input

```
def Kuadrat ( 5 ):
# menghitung kuadrat 5
    hasil = 5 * 5
    return hasil        #hasil=25
```

3) Program meninggalkan fungsi dengan menyimpan hasil perhitungan dan kembali pada baris terakhir program utama yang ditinggalkannya dan menggantikan hasil perhitungan berdasarkan hasil fungsi: B = **25** + 10 # B = 35
4) Program melanjutkan ke instruksi berikutnya.

Contoh (1)

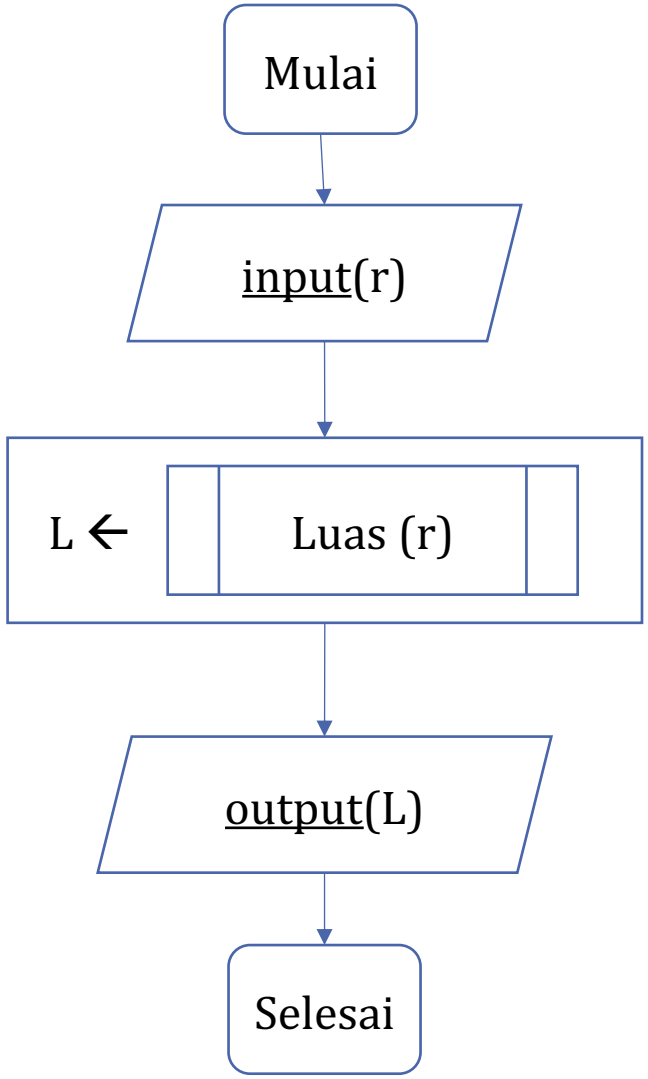
- Buatlah program yang menerima masukan buah nilai jari-jari lingkaran (bilangan riil), misalnya r, dan menuliskan luas lingkaran ke layar

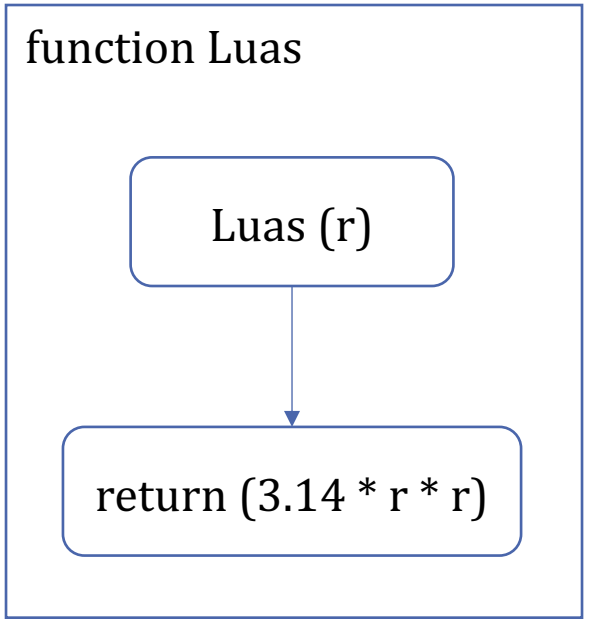
- Perhitungan luas lingkaran → dibuat menjadi fungsi

```
# Program LuasLingkaran
# Menghitung luas lingkaran berdasarkan jari-jari

# KAMUS
# r, L : float
# Definisi dan Realisasi Fungsi Luas
def Luas ( r ):
# menghasilkan luas lingkaran berdasarkan jari-jari r
    Luas := 3.14 * r * r
    return (Luas)

# PROGRAM UTAMA
r = float(input())
L = Luas(r)  # pemanggilan fungsi Luas
print(L)
```

Contoh (2)

- Flowchart





Kegunaan Fungsi

- Program dapat didekomposisi menjadi sub-sub bagian
  - Tiap sub bagian dapat didefinisikan sebagai fungsi yang tinggal dipanggil sebagai 1 baris atau ekspresi dalam program utama
- *Code reuse instead of code rewriting*
  - Jika task yang harus dikerjakan fungsi banyak dipakai di program, memrogram menjadi jauh lebih sederhana jika task tersebut dibuat dalam bentuk fungsi
  - Contoh: fungsi untuk menghitung akar kuadrat (**sqrt**) sangat berguna untuk berbagai jenis persoalan → bayangkan kalau setiap kali Anda harus menulis programnya ☺
- Setiap fungsi dapat dites secara mandiri dan tidak tergantung pada bagian program yang lain
  - Di Python: fungsi dapat dites dulu dalam interpreter (tidak harus membuat program utuh terlebih dahulu)
  - Jika program besar dan harus dikerjakan oleh lebih dari 1 programmer, hal ini memudahkan pembagian kerja

Tahap Penggunaan Fungsi

1. Mendefinisikan fungsi
   - Mendefinisikan nama, spesifikasi, domain (input), range (output)
2. Merealisasikan fungsi
   - Membuat program untuk menghasilkan output berdasarkan input
3. Menggunakan fungsi
   - Memanggil fungsi di program utama atau dalam fungsi lain

```
# Program Test
# Mengetes fungsi kuadrat

# KAMUS
#     A : integer
#     B : integer

# Fungsi Kuadrat
def Kuadrat ( X ) :
    # menghitung kuadrat X
    hasil = X * X
    return hasil

# ALGORITMA PROGRAM UTAMA
A = 5
B = Kuadrat (A) + 10
```

Mendefinisikan Fungsi

- Mendefinisikan fungsi dalam program berarti mendefinisikan di bagian blok **KAMUS**:
  - **Nama** dan **spesifikasi** fungsi
  - Himpunan nilai **domain**: type data **parameter input**
  - Himpunan nilai **range**: type data **output**
- Contoh: Fungsi kuadrat: $f(x) = x^2$
  - Nama fungsi: **kuadrat** → nama dibuat oleh programmer
  - Spesifikasi fungsi: "menghasilkan kuadrat dari x"
  - Type data input: **int**
  - Type data output: **int**

Mendefinisikan Fungsi dalam Python

- **Nama fungsi** didefinisikan setelah keyword **def**
- **Spesifikasi fungsi** dituliskan dalam bentuk komentar di bawah nama fungsi
- Type data **input** didefinisikan **implisit** berdasarkan type data **parameter_input**
  - Jika lebih dari 1, tiap parameter dipisahkan dengan koma (,)
- Type data **output** didefinisikan secara **implisit** berdasarkan type nilai_output yang dituliskan setelah perintah **return**

```
# definisi fungsi
def <nama_fungsi> ( [<parameter_input>] ):
    # spesifikasi_fungsi
    ...
    return [nilai_output]
```

Contoh fungsi Kuadrat:

```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X
    ...
    return hasil
```

Merealisasikan Fungsi

- Merakit program untuk menghasilkan nilai output berdasarkan nilai input
  - Pada dasarnya dapat menggunakan segala jenis instruksi yang mungkin dalam program
- **KAMUS LOKAL**: dimungkinkan ada nama-nama variabel yang hanya terdefinisi lokal di fungsi (tidak bisa dipakai di program utama atau di fungsi yang lain)
- **ALGORITMA**: bagian program yang berisi kode program fungsi dan minimum mengandung 1 buah perintah **return**
  - **return**: perintah untuk menuliskan hasil fungsi

```
# definisi fungsi
def <nama_fungsi> ( [<parameter_input>] ):
    # spesifikasi_fungsi

    # KAMUS LOKAL
    # nama-nama variabel lokal

    # ALGORITMA
    ... # deretan instruksi untuk
        # menghasilkan nilai_output
        # berdasarkan nilai parameter_input
    return [<nilai_output>]
```

Contoh fungsi Kuadrat:

```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X

    # KAMUS LOKAL
    # hasil : int

    # ALGORITMA
    hasil = X * X
    return hasil
```

Contoh-1: Fungsi Kuadrat (1)

- Definisi matematika: **$f(x) = x^2$**
- Bagaimana memindahkannya dalam program?
  - **Nama fungsi**: Kuadrat → ditentukan oleh programmer
  - **Spesifikasi fungsi**: menghasilkan kuadrat dari input
  - **Type domain/input**: integer, didefinisikan oleh parameter input x
  - **Type range/output**: integer → berdasarkan type hasil x^2
  - **Realisasi** fungsi: x * x atau x ** 2 (dalam Python)

```
#  Fungsi Kuadrat

def    Kuadrat    (          X          ):

   # Menghasilkan kuadrat dari X

   # KAMUS LOKAL
   # hasil : int

   # ALGORITMA

   hasil = X * X


   return  hasil
```

Contoh-1: Fungsi Kuadrat (2)

- Alternatif: tidak perlu variabel kamus lokal → langsung ekspresi di bagian return
  - Untuk program-program yang sangat pendek, ini lebih baik

```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X

    # KAMUS LOKAL
    # hasil : int

    # ALGORITMA
    hasil = X * X
    return hasil
```

```
# definisi fungsi Kuadrat
def Kuadrat ( X ):
    # menghasilkan kuadrat X

    # KAMUS LOKAL

    # ALGORITMA

    return X * X
```

Contoh-2: Nilai maksimum 2 integer

- Buatlah fungsi yang menghasilkan nilai terbesar dari 2 buah bilangan integer.
- **Nama fungsi**: Max2
- **Spesifikasi fungsi**: Bila diketahui A dan B bilangan integer, dihasilkan bilangan terbesar antara A dan B
- **Type input**: 2 bilangan integer: A dan B
- **Type output**: bilangan integer (maksimum dari A dan B pasti juga integer)
- **Realisasi** fungsi: menggunakan if-else

```
#  Fungsi Max2

def      Max2      (        A , B         ):
    # Menghasilkan bilangan terbesar
    # antara A dan B
    # KAMUS LOKAL
    # maks : int

    # ALGORITMA
    if (A >= B):
        maks = A
    else:  # A < B
        maks = B

    return maks
```

Contoh-2: Nilai maksimum 2 integer (2)

- Alternatif: tidak perlu variabel kamus lokal

```
# Fungsi Max2
def Max2 ( A, B ):
    # menghasilkan nilai terbesar
    # antara A dan B

    # KAMUS LOKAL
    # maks : int

    # ALGORITMA
    if (A >= B):
        maks = A
    else:   # A < B
        maks = B

    return maks
```

```
# Fungsi Max2
def Max2 ( A, B ):
    # menghasilkan nilai terbesar
    # antara A dan B

    # KAMUS LOKAL

    # ALGORITMA
    if (A >= B):
        return A
    else:   # A < B
        return B
```

Latihan-1:

- Buatlah definisi dan realisasi fungsi **Max3** untuk menghitung nilai maksimum dari 3 bilangan, misalnya A, B, C.
- Contoh: A = 1, B = -10, C = 5 → maksimum = 5

- Algoritma:
  - A >= B and A >= C : maksimum = A
  - B >= A and B >= C : maksimum = B
  - C >= A and C >= B : maksimum = C

Menggunakan Fungsi (1)

- Fungsi dipanggil dalam instruksi program utama atau dalam instruksi di fungsi lain sebagai bagian dari **ekspresi**
- Syarat memanggil fungsi:
  - Nama fungsi harus sama
  - Banyaknya parameter input sama dan type data bersesuaian
    - Dalam proses pemanggilan fungsi akan terjadi asosiasi satu ke satu setiap parameter input dengan nilai masukan
  - Hasil dari pemanggilan fungsi harus dalam type yang sama dengan type output fungsi
  - Pemanggilan fungsi sebagai bagian dari ekspresi → bukan sebuah instruksi terpisah

Menggunakan Fungsi (2) - Contoh

- Nama harus sama: **Kuadrat**
- Banyaknya parameter input sama dan type data bersesuaian
  - Ada parameter input di fungsi Kuadrat yaitu x; dan ada 1 input di pemanggilan Kuadrat di program utama, yaitu y. x dan y sama-sama bertype integer.
- Hasil dari pemanggilan fungsi harus dalam type yang sama dengan type output fungsi.
  - Perintah return di fungsi Kuadrat memberikan data bertype integer
  - Pada pemanggilan di program utama: Kuadrat(y) akan menghasilkan integer dan tepat dengan type variabel
- Pemanggilan fungsi sebagai bagian dari ekspresi
  - Ya, Kuadrat adalah ekspresi yang ditampung hasilnya di variabel hasil

```
# Program HitungKuadrat
# Menerima masukan sebuah integer dan
# menuliskan pangkat 2 dari nilai tsb
# ke layar

# Kamus
# y, hasil : int

# Definisi Fungsi
def Kuadrat ( x ):
    # Menghasilkan pangkat 2 dari x
    # Algoritma
    return x*x

# Algoritma Program Utama
y = int(input("Masukkan bilangan = "))
hasil = Kuadrat(y)
print("Kuadrat dari "+str(y)+" = "+str(hasil))
```

Contoh-3: 10% dari bilangan terbesar

- Buatlah program yang menerima masukan 2 buah integer, misalnya A dan B. Tuliskan ke layar 10% dari nilai terbesar di antara keduanya.
- Contoh: A = 45 B = 50 terbesar = 50

  tercetak di layar = 5.0

- Untuk mencari nilai terbesar, buat dan gunakan fungsi Max2 yang telah didiskusikan di Contoh-2

Contoh-3: 10% dari bilangan terbesar

```
# Program 10persen_dari_terbesar
# Menerima masukan 2 bilangan integer
# Menuliskan ke layar 10% dari bilangan terbesar di antara keduanya

# KAMUS
# A, B : int
# hasil : float

# Definisi Fungsi Max2
def Max2 (A,B):
    # Menghasilkan bilangan terbesar antara A dan B

    # KAMUS LOKAL
    if (A >= B):
        return A
    else: # A < B
        return B

# ALGORITMA PROGRAM UTAMA
A = int(input("Masukkan bilangan pertama = "))
B = int(input("Masukkan bilangan kedua = "))

hasil = 0.1 * Max2(A,B)

print ("10% dari bilangan terbesar = " + str(hasil))
```

Latihan-2

- Selain dipanggil di program utama, fungsi juga bisa dipanggil di fungsi lain.

- Buatlah definisi dan realisasi **fungsi** bernama **Max3** untuk menghitung nilai maksimum dari 3 bilangan, misalnya A, B, C.
- Contoh: A = 1, B = -10, C = 5 → maksimum = 5
- Realisasikan fungsi ini dengan cara membuat dan menggunakan fungsi **Max2** yang telah didiskusikan di Contoh-2.

Prosedur

Prosedur

- **Prosedur**: subprogram mengelompokkan instruksi-instruksi yang sering dipakai di program
  - Tidak harus ada parameter input/output
  - Dapat dipandang sebagai fungsi yang tidak menghasilkan (*return*) nilai
- Dalam Python, didefinisikan dengan **return** tanpa ekspresi/nilai yang dihasilkan di akhir fungsi

```
# definisi prosedur
def <nama_prosedur> ( [<parameter_input>] ):
    # spesifikasi_prosedur

    # KAMUS LOKAL
    # nama-nama variabel lokal

    # ALGORITMA
    ... # deretan instruksi prosedur
    return
```

Contoh-4: Hello Nama

- Buatlah fungsi **CetakNama** yang menerima masukan sebuah string nama dan mencetak “Hello, ” + nama ke layar.
- Tidak ada nilai yang dikeluarkan dari fungsi

```
# Definisi Subprogram
def CetakNama (nama):
    # Mencetak Hello + nama ke layar

    # Algoritma
    print ("Hello, " + nama + "!!")
    return
```

Memanggil Prosedur

- Karena prosedur tidak menghasilkan nilai, pemanggilannya dalam program utama atau fungsi lain juga berbeda.
- Prosedur dipanggil sebagai **1 buah baris instruksi**, bukan sebagai bagian dari ekspresi.
- Asosiasi parameter input dilakukan dengan cara yang sama seperti pada fungsi biasa

Contoh-5: Hello Nama v2

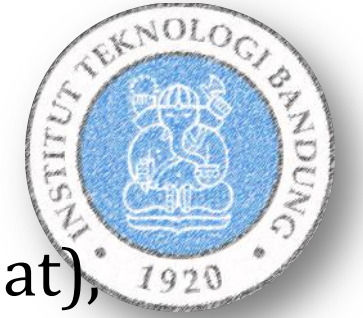
Buatlah program yang menerima masukan sebuah integer > 0, misalnya N, dan sebuah string, misalnya nama lalu mencetak: "Hello, **nama**!" sebanyak **N** kali ke layar

```
# Program HelloHelloNama
# Menerima masukan sebuah integer > 0 N dan string nama
# dan mencetak "Hello" + nama sebanyak N kali

# Kamus
# i, N : int
# nama : string

# Definisi Prosedur CetakNama
def CetakNama (nama):
    # Mencetak Hello + nama ke layar
    # Algoritma
    print ("Hello, " + nama + "!")
    return

# Algoritma Program Utama
nama = input("Masukkan nama = ")
N = int(input("Berapa kali diulang? "))

for i in range(N):
    CetakNama(nama)
```

Fungsi Standar Python

- Dalam Python didefinisikan sangat banyak fungsi standar yang tersedia dan tinggal digunakan → jadi tidak perlu di-*coding* lagi
- Fungsi-fungsi standar ini didefinisikan dalam *library*
- Contoh library standar yang sering dipakai adalah **math**
- Fungsi-fungsi yang didefinisikan dalam library math:
  - **sqrt** → mencari akar kuadrat suatu bilangan
  - **sin** → mencari sinus
  - **cos** → mencari cosinus
  - **pow** → pangkat suatu bilangan
  - dll.
- Memanggil library **math** dengan menambahkan instruksi pada bagian awal program: `from math import *`
- Informasi lebih lanjut: https://docs.python.org/3/library/math.html

Contoh-6: Akar Kuadrat



Buatlah program yang menerima masukan sebuah bilangan riil (float), misalnya N, dan menghasilkan akar kuadrat dari bilangan tersebut

Contoh:

N = 4; sqrt(N) = 2.0

N = 12; sqrt(N) = 3.464…

```
# Program HitungAkar
# Menerima masukan sebuah bilangan riil
# menuliskan akar dari bilangan tersebut

from math import *

# KAMUS
# N : float

# ALGORITMA
N = float(input("Masukkan bilangan = "))
print ("Akar dari = " + str(N) + " = " + str(sqrt(N)))
```

Latihan-3

- Buatlah sebuah fungsi bernama **HitungJarak**, yang menerima masukan: **v** : kecepatan (dalam m/s, bilangan riil) dan **t** : waktu tempuh (dalam s, bilangan riil) dan menghasilkan jarak tempuh **s** dengan rumus: s = v * t.
- Asumsikan nilai t ≥ 0 dan s ≥ 0.

- Selanjutnya, buatlah program utama yang menggunakan fungsi HitungJarak tersebut (bebas).

Latihan-4

- Masih ingat program untuk mencari nilai maksimum array?
- Buatlah fungsi **MaxArray** yang menerima masukan sebuah array of integer, misalnya T, dan panjang array, misalnya N, dan menghasilkan nilai terbesar yang disimpan dalam array tersebut. Asumsikan N > 0.
- Contoh: T = [5, 4, 3, 2, 1]; N = 5

  maka nilai maksimum = 5

Latihan-5

- Salah satu *task* dalam pemrosesan array adalah mencetak semua elemen array ke layar
- Buatlah prosedur **CetakArray** yang menerima masukan sebuah array of integer, misalnya T, dengan panjang N >= 0, dan mencetak semua elemen array ke layar.
- Cara mencetak: setiap elemen ke-i di cetak per baris dengan cara sbb: `[i] <elemen>`
- Contoh: T = [5, 4, 3, 2, 1]; N = 5 akan tercetak:

```
[0] 5
[1] 4
[2] 3
[3] 2
[4] 1
```

- Jika N = 0, cetaklah: 'Array kosong'