LEMBAR PENGESAHAN
PRAKTIKUM PEMROGRAMAN 2

Disusun Oleh : NUR ROKHMAN

NIM : 1197070059

Kelas : A2

Mengetahui,

Asisten Praktikum

Rizal Miftahussalim, S.T.	Mohammad Adhipramana, S.T.
Nabila Safitri Dwi	Saip Ardo Pratama

Menyetujui,

Dosen Praktikum

Aan Eko Setiawan, S.T.

Python Dasar: List, Slicing, Loops, dan Functions

Basic Python: List, Slicing, Loops, and Functions

NUR ROKHMAN[1197070059]
Jurusan Teknik Elektro, Fakultas Sains dan Teknik Elektro
Universitas Islam Negeri Sunan Gunung Djati Bandung
Jl. A.H. Nasution No. 105A, Cibiru, Bandung, Jawa Barat, Indonesia
nurrokhman0302@gmail.com

Abstract, *Python is a multipurpose interpretive programming language. Even at the end of the code, do not have to end with a semicolon ";". There are many ways to run Python, one of which is using Google Colab. Colab is a cloud-based runtime, which runs using a browser (Chrome, Firefox and Safari). Some of the basic python material presented in this paper, including lists, slicing, loops, and functions, is the most fundamental material in the Python programming language.*

Keywords: python, google colab, lists, slicing, loops, functions

Abstrak, *Python adalah bahasa pemrograman interpretatif multiguna. Dibagian akhir kode pun, tidak harus mengakhirnya dengan tanda semicolon ";". Untuk menjalankan Python ada banyak cara yang bisa dilakukan salah satunya dengan Google Colab. Colab merupakan* cloud-based runtime, *yang dijalankan menggunakan browser (Chrome, Firefox dan Safari). Beberapa materi python dasar yang tersaji dalam tulisan ini diantaranya lists, slicing, loops, dan functions adalah merupakan materi paling fundamental didalam Bahasa pemrograman python.*

Kata Kunci: python, google colab, lists, slicing, loops, functions

I. PENDAHULUAN

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Sumber: belajarpython.com

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya.

Dengan kode yang simpel dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error.

```
print("Python sangat simpel")
```

Hanya dengan menuliskan kode print seperti yang diatas, anda sudah bisa mencetak apapun yang anda inginkan di dalam tanda kurung `()`. Dibagian akhir kode pun, anda tidak harus mengakhirnya dengan tanda semicolon `;`

Python banyak digunakan untuk membuat berbagai macam program, seperti: program CLI, Program GUI (desktop), Aplikasi Mobile, Web, IoT, Game, Program untuk Hacking, dsb. Python juga dikenal dengan bahasa pemrograman yang mudah dipelajari, karena struktur sintaknya rapi dan mudah dipahami.

Python memang sangat sederhana dibandingkan bahasa yang lainnya. Tidak perlu ini dan itu untuk membuat program Hello World!. Bahkan tagline di websitenya menjelaskan, kalau python akan membuatmu bekerja lebih cepat dan efektif.

"Python is a programming language that lets you work quickly and integrate systems more effectively".

Tujuan dari praktikum dan penulisan tulisan ini ditunjukkan untuk memenuhi Ujian Akhir Semester matakuliah Praktikum Pemroraman 2 di jurusan Teknik Elektro UIN Sgd Bandung. Dalam hal ini, mahasiswa diharapkan mampu memahami dan menjelaskan secara teori maupun praktek mengenai python dasar diantaranya lists, slicing, loops dan functions.

II. TEORI DASAR

A. Python

Sebelum menggunakan Python, harus menginstalnya terlebih dahulu sesuai sistem operasi komputer. Saat ini Python memiliki 2 versi yang berbeda, yaitu Python versi 3.4.3 dan Python versi 2.7.10. Cara menginstal python sangat mudah, ikuti panduan dibawah ini. Dibawah adalah panduan cara instal python di platform Linux, Windows dan Mac OS.

- Linux
  1. Buka browser kunjungi http://www.python.org/downloads/source/
  2. Download versi terbaru Python berbentuk file zip untuk Unix/Linux
  3. Ekstrak file zip yang baru saja di download
  4. Edit file Modules/Setup jika Anda ingin kostumisasi Python
  5. Jalankan `./configure` script
  6. `Make`
  7. `make install`

  Langkah ini akan menginstal Python di lokasi standar `/usr/local/bin` dan librarydi `/usr/local/lib/pythonXX` dimana `XX` adalah versi terbaru Python yang anda gunakan.

"Untuk beberapa distro (distribution store) dari sistem operasi linux sudah terinstal Python di dalamnya. Jadi Anda tidak perlu menginstalnya lagi".

- Windows
  1. Buka browser, kunjungi http://www.python.org/downloads/windows/
  2. ATAU, klik direct link https://www.python.org/ftp/python/3.8.1/python-3.8.1.exe
  3. Buka (klik 2x) file installer python yang baru saja di download
  4. Ikuti langkah instalasi sampai selesai

- Mac OS
  1. Buka browser, kunjungi http://www.python.org/download/mac/
  2. Download versi terbaru Python untuk Macintosh
  3. Buka file yang baru saja di download
  4. Ikuti langkah instalasi sampai selesai

B. Google Colab

Untuk menjalankan Python ada banyak cara yang bisa dilakukan, bisa menggunakan *shell*, terminal atau menggunakan IDE (Integrated Development Environment).

Seperti Google Drive, Google Doc, dan sebagainya, Google Colab adalah salah satu produk Google berbasis cloud yang bisa kita gunakan secara gratis. Perbedaannya adalah Google Colab dibuat khusus untuk para programmer atau researcher yang mungkin kesulitan untuk mendapatkan akses komputer dengan spek tinggi. Google Colab adalah coding environment bahasa pemrograman Python dengan format "notebook" (mirip dengan Jupyter notebook), atau dengan kata lain Google seakan meminjami kita komputer secara gratis! untuk membuat program oleh Google.

Sumber: iglab.tech

Google Colab (atau cukup disebut Colab) adalah produk dari Google Research. Colab adalah *executable document*, yang bisa digunakan untuk menulis, menyimpan, serta membagikan program yang telah ditulis melalui Google Drive. Jika familiar dengan Jupyter Notebook, maka Colab bisa dikatakan sebagai Notebook yang disimpan pada Google Drive. Jika belum pernah mencoba Jupyter Notebook sebelumnya, jangan khawatir.

Colab merupakan *cloud-based runtime*, yang dijalankan menggunakan browser (Chrome, Firefox dan Safari). Colab memungkinkan menjalankan kode Python tanpa memerlukan proses instalasi dan setting lainnya pada komputer pribadi kita. Semua akan diserahkan ke cloud. Kita dapat juga menggunakan berbagai fungsionalitas yang dimiliki oleh Python serta memanfaatkan *built-in* library yang disediakan oleh Colab. Sebagai contoh, jika kita ingin membuat visualisasi data, kita dapat menggunakan contoh koding program yang telah disediakan oleh Colab dan memasukkan cuplikan koding tersebut ke project kita. Terlebih dari itu semua, layanan Colab diberikan secara gratis. Dengan hanya memiliki akun Google, kita dapat menggunakan Colab.

Kode pada Colab dieksekusi pada mesin virtual tersendiri, bersesuaian dengan akun Google yang kita miliki. Karena fasilitas ini sifatnya gratis, maka tentu Google malakukan pengaturan ataupun pembatasan dalam penggunaan Colab. Mesin virtual ini dapat terhenti jika kita dalam posisi *idle* dalam rentang waktu tertentu, dan maksimum menjalankan Notebook saat ini dilaporkan selama maksimum 12 jam. Namun demikian, lama penggunaan dari mesin virtual ini dapat bervariasi. Bagi kalian yang ingin mendapatkan manfaat lebih dari Colab, kalian bisa menggunakan versi Colab berbayar, yaitu Colab Pro.

Menariknya, Colab menawarkan GPU (*graphics processing unit*) yang juga dapat diakses secara Gratis. Saat ini, GPU yang disedikan oleh Colab adalah Nvidia K80s, T4s, P4s dan P100s. Untuk versi gratis, kita tidak dimungkinkan untuk memilih jenis GPU mana yang akan digunakan. Jika ingin mendapatkan fasilitas lebih terkait GPU, lagi-lagi kalian dianjurkan untuk menggunakan Colab Pro.

Memori yang disediakan oleh Colab bervariasi setiap waktu, namun akan stabil selama mesin virtual berjalan. Strategi mengatur 'jatah' memori ini adalah strategi dari Google agar versi free ini bisa tetap ditawarkan ke masyarakat luas. Terkadang kita juga dapat ditawarkan memori lebih jika Google menyimpulkan kalau proses kita memutuhkannya. Google tidak secara resmi mengumumkan kebijakan berapa besar sumber daya yang bisa diberikan kepada para *user*.

Sumber daya pada Colab diutamakan bagi mereka yang menggunakan jatah sedikit. Hal ini dilakukan Google untuk menghidari monopoli sumber daya oleh sekelompok kecil pengguna. Untuk mendapatkan akses cloud yang lebih lancar, kita disarankan untuk menutup tab Colab pada browser ketika sudah selesai menggunkannya, atau tidak memilih GPU jika memang tidak diperlukan dalam pekerjaan kita. Hal ini akan mengurangi kemungkinan pembatasan penggunaan layanan cloud pada Google.

C. Loops

Secara umum, pernyataan pada bahasa pemrograman akan dieksekusi secara berurutan. Pernyataan pertama dalam sebuah fungsi dijalankan pertama, diikuti oleh yang kedua, dan seterusnya. Tetapi akan ada situasi dimana Anda harus menulis banyak kode, dimana kode tersebut sangat banyak. Jika dilakukan secara manual maka Anda hanya akan membuang-buang tenaga dengan menulis beratus-ratus bahkan beribu-ribu kode. Untuk itu Anda perlu menggunakan pengulangan di dalam bahasa pemrograman Python.

Di dalam bahasa pemrograman Python pengulangan dibagi menjadi 3 bagian, yaitu :

- While Loop
- For Loop
- Nested Loop

➢ While Loop

Pengulangan While Loop di dalam bahasa pemrograman Python dieksesusi statement berkali-kali selama kondisi bernilai benar atau True. Dibawah ini adalah contoh penggunaan pengulangan While Loop.

```
#Contoh penggunaan While Loop
#Catatan: Penentuan ruang lingkup
di Python bisa menggunakan tab
alih-alih menggunakan tanda
kurung
```

```
count = 0
while (count < 9):
    print ("The count is: ",
count)
    count = count + 1

print ("Good bye!")
```

- For Loop

Pengulangan for pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti list atau string. Dibawah ini adalah contoh penggunaan pengulangan For Loop.

```
#Contoh pengulangan for sederhana
angka = [1,2,3,4,5]
for x in angka:
    print(x)

#Contoh pengulangan for
buah = ["nanas", "apel", "jeruk"]
for makanan in buah:
    print ("Saya suka makan",
makanan)
```

- Nested Loop

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain. Bagian berikut menunjukkan beberapa contoh untuk menggambarkan konsep tersebut. Dibawah ini adalah contoh penggunaan Nested Loop.

```
#Contoh penggunaan Nested Loop
#Catatan: Penggunaan modulo pada
kondisional mengasumsikan nilai
selain nol sebagai True(benar)
dan nol sebagai False(salah)

i = 2
while(i < 100):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print(i, " is
prime")
    i = i + 1

print("Good bye!")
```

D. List

Dalam bahasa pemrograman Python, struktur data yang paling dasar adalah urutan atau lists. Setiap elemen-elemen berurutan akan diberi nomor posisi atau indeksnya. Indeks pertama dalam list adalah nol, indeks kedua adalah satu dan seterusnya.

Python memiliki enam jenis urutan built-in, namun yang paling umum adalah list dan tuple. Ada beberapa hal yang dapat Anda lakukan dengan semua jenis list. Operasi ini meliputi pengindeksan, pengiris, penambahan, perbanyak, dan pengecekan keanggotaan. Selain itu, Python memiliki fungsi built-in untuk menemukan panjang list dan untuk menemukan elemen terbesar dan terkecilnya.

- Membuat List Python

List adalah tipe data yang paling serbaguna yang tersedia dalam bahasa Python, yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya.

Membuat list sangat sederhana, tinggal memasukkan berbagai nilai yang dipisahkan koma di antara tanda kurung siku. Dibawah ini adalah contoh sederhana pembuatan list dalam bahasa Python.

```
#Contoh sederhana pembuatan list
pada bahasa pemrograman python
list1 = ['kimia', 'fisika', 1993,
2017]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

- Akses Nilai Dalam List Python

Untuk mengakses nilai dalam list python, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Berikut adalah contoh cara mengakses nilai di dalam list python :

```
#Cara mengakses nilai di dalam
list Python

list1 = ['fisika', 'kimia', 1993,
2017]
list2 = [1, 2, 3, 4, 5, 6, 7 ]

print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
```

Setelah Anda mengeksekusi kode diatas, hasilnya akan seperti dibawah ini:

```
list1[0]:  fisika  list2[1:5]:  [2,
3, 4, 5]
```

- Update Nilai Dalam List Python

Anda dapat memperbarui satu atau beberapa nilai di dalam list dengan

memberikan potongan di sisi kiri operator penugasan, dan Anda dapat menambahkan nilai ke dalam list dengan metode append (). Sebagai contoh :

```
list = ['fisika', 'kimia', 1993, 2017]
print ("Nilai ada pada index 2 : ", list[2])

list[2] = 2001
print ("Nilai baru ada pada index 2 : ", list[2])
```

- Hapus Nilai Dalam List Python

Untuk menghapus nilai di dalam list python, Anda dapat menggunakan salah satu pernyataan del jika Anda tahu persis elemen yang Anda hapus. Anda dapat menggunakan metode remove() jika Anda tidak tahu persis item mana yang akan dihapus. Sebagai contoh:

```
#Contoh cara menghapus nilai pada list python

list = ['fisika', 'kimia', 1993, 2017]

print (list)
del list[2]
print ("Setelah dihapus nilai pada index 2 : ", list)
```

- Operasi Dasar Pada List Python

List Python merespons operator + dan * seperti string; Itu artinya penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah list baru, bukan sebuah String.

Sebenarnya, list merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python.

Python Expression	Hasil	Penjelasan
`len([1, 2, 3, 4])`	`4`	Length
`[1, 2, 3] + [4, 5, 6]`	`[1, 2, 3, 4, 5, 6]`	Concatenation
`['Halo!'] * 4`	`['Halo!', 'Halo!', 'Halo!', 'Halo!']`	Repetition
`2 in [1, 2, 3]`	`True`	Membership
`for x in [1,2,3] : print (x,end = ' ')`	`1 2 3`	Iteration

- Indexing, Slicing dan Matrix Pada List Python

Karena list adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk list seperti yang mereka lakukan untuk String. Dengan asumsi input berikut:

```
L = ['C++'', 'Java', 'Python']
```

Python Expression	Hasil	Penjelasan
`L[2]`	`'Python'`	Offset mulai dari nol
`L[-2]`	`'Java'`	Negatif: hitung dari kanan

Python Expression	Hasil	Penjelasan
`[1:]`	`['Java', 'Python']`	Slicing mengambil bagian

- Method dan Fungsi Build-in Pada List Python

Python menyertakan fungsi built-in sebagai berikut:

Python Function	Penjelasan
cmp(list1, list2) #	Tidak lagi tersedia dengan Python 3
len(list)	Memberikan total panjang list.
max(list)	Mengembalikan item dari list dengan nilai maks.
min(list)	Mengembalikan item dari list dengan nilai min.
list(seq)	Mengubah tuple menjadi list.

Python menyertakan methods built-in sebagai berikut:

Python Methods	Penjelasan
list.append(obj)	Menambahkan objek obj ke list
list.count(obj)	Jumlah pengembalian berapa kali obj terjadi dalam list
list.extend(seq)	Tambahkan isi seq ke list
list.index(obj)	Mengembalikan indeks terendah dalam list yang muncul obj
list.insert(index, obj)	Sisipkan objek obj ke dalam list di indeks offset
list.pop(obj = list[-1])	Menghapus dan mengembalikan objek atau obj terakhir dari list
list.remove(obj)	Removes object obj from list
list.reverse()	Membalik list objek di tempat
list.sort([func])	Urutkan objek list, gunakan compare func jika diberikan

E. Functions (Fungsi)

Fungsi adalah blok kode terorganisir dan dapat digunakan kembali yang digunakan untuk melakukan sebuah tindakan/action. Fungsi memberikan modularitas yang lebih baik untuk aplikasi Anda dan tingkat penggunaan kode yang tinggi. Developer dapat menentukan fungsi untuk menyediakan fungsionalitas yang dibutuhkan. Berikut adalah aturan sederhana untuk mendefinisikan fungsi dengan Python.

- Fungsi blok dimulai dengan def kata kunci diikuti oleh nama fungsi dan tanda kurung (()).
- Setiap parameter masukan atau argumen harus ditempatkan di dalam tanda kurung ini. Anda juga dapat menentukan parameter di dalam tanda kurung ini.
- Pernyataan pertama dari sebuah fungsi dapat berupa pernyataan opsional - string dokumentasi fungsi atau docstring.
- Blok kode dalam setiap fungsi dimulai dengan titik dua (:) dan indentasi.
- Pernyataan kembali [ekspresi] keluar dari sebuah fungsi, secara opsional menyampaikan kembali ekspresi ke pemanggil. Pernyataan pengembalian tanpa argumen sama dengan return None.
  Contoh fungsi:

```
def printme( str ):
   "This prints a passed string
into this function"
   print (str)
   return
```

F. Slicing

Slicing merupakan teknik memilih data dari sebuah set data. Misal kita memiliki data berat badan mahasiswa: 65, 78, 77, 100, 56. Maka jika kita urutkan maka urutan pertama adalah 65, urutan kedua adalah 78, urutan ketiga adalah 77, urutan keempat adalah 100, dan urutan terakhir adala 56. Jika didefinisikan ke dalam list, maka akan seperti berikut:

```
berat_badan=[65,78,77,100,56]
berat_badan[1]
berat_badan[4]
berat_badan[::2]
```

- Jika kita mengeksekusi baris kedua, maka yang keluar adalah 78
- Jika kita mengeksekusi baris ketiga, maka yang keluar adalah 56
- Jika kita mengekseusi baris keempat, maka yang keluar adalah item urutan kedua 65, 77, 56

Pertanyaannya, mengapa berat_badan[1] justru memberi input 78, dan bukan 65? Karena *indexing* (proses urutan) dalam python dimulai dari nol. Sehingga *list* berat_badan dimulai dari 0,1,2,3,4.

```
berat_badan[0]
berat_badan[:]
berat_badan[1:4]
```

- Jika eksekusi baris pertama, maka hasilnya adalah 65, karena index [0] menunjukkan urutan pertama
- Jika eksekusi baris kedua, maka hasilnya semua data dalam list, karena index [:] menunjukkan semua data
- Jika eksekusi baris ketiga, maka hasilnya [78,77,100] karena menunjukkan urutan data kedua sampai keempat.

Perlu diperhatikan bahwa slicing dalam python, urutan terakhir tidak dibaca oleh bahasa python itu sendiri. Maka penulisan [1:4] kita memberikan perintah slicing mulai index 1 sampai index 4 sebagai batas akhir (di mana index 4 ini tidak termasuk ke dalamnya). Sehingga yang dieksekusi adalah index ke 1 sampai 3. Jika di eksekusi:

```
berat_badan[:3]
```

Maka yang keluar adalah [65,78,77] karena kita mengindex dari 0 sampai 2 (3 sebagai batasnya, tidak termasuk). Jika kita menulisnya dengan berat_badan[0:3] hasilnya juga akan sama.

III. LANGKAH PEMBUATAN PROGRAM

1) Membuat Lists

```
xs = [3, 1, 2]   # Create a list
print(xs, xs[2])
print(xs[-1])     # Negative
indices count from the end of the
list; prints "2"
```

```
xs[2] = 'foo'    # Lists can
contain elements of different
types
print(xs)
xs.append('bar') # Add a new
element to the end of the list
print(xs)
x = xs.pop()     # Remove and
return the last element of the
list
print(x, xs)
```

2) Membuat Slicing

```
nums = list(range(5))    # range
is a built-in function that
creates a list of integers
print(nums)         # Prints "[0,
1, 2, 3, 4]"
print(nums[2:4])    # Get a slice
from index 2 to 4 (exclusive);
prints "[2, 3]"
print(nums[2:])     # Get a slice
from index 2 to the end; prints
"[2, 3, 4]"
print(nums[:2])     # Get a slice
from the start to index 2
(exclusive); prints "[0, 1]"
print(nums[:])      # Get a slice
of the whole list; prints ["0, 1,
2, 3, 4]"
print(nums[:-1])    # Slice
indices can be negative; prints
["0, 1, 2, 3]"
nums[2:4] = [8, 9] # Assign a new
sublist to a slice
print(nums)         # Prints "[0,
1, 8, 9, 4]"
```

3) Membuat Loops

- Program 1

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
```

- Program 2

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in
enumerate(animals):
    print('#{}: {}'.format(idx +
1, animal))
```

4) Membuat Functions

- Program 1

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'
```

- Program 2

```
for x in [-1, 0, 1]:
    print(sign(x))
```

- Program 3

```
def hello(name, loud=False):
    if loud:
        print('HELLO,
{}'.format(name.upper()))
    else:
        print('Hello,
{}!'.format(name))

hello('Bob')
hello('Fred', loud=True)
```

IV. PENGUJIAN DAN PEMBAHASAN

1) Membuat Lists

```
xs = [3, 1, 2]   # Create a list
print(xs, xs[2])
```

Struktur kode diatas adalah struktur kode untuk membuat sebuah list. yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya. Pada dasarnya struktur kode diatas adalah variabel biasa dengan nama xs yang diisi oleh list dengan ciri kurung siku yang didalamnya tersimpan beberapa elemen berurutan indeksnya, diantaranya secara berurutan yaitu 3, 1, dan 2.

Print yang diikuti tanda kurung pada python adalah perintah untuk menampilkan sebuah data yang ada pada tanda kurungnya tersebut. Dalam hal ini, print(xs, xs[2]) akan menampilkan elemen list xs tanpa menggunakan indeks dan elemen list dengan menggunakan indeksi yaitu xs[2] yang berarti elemen list xs dengan indeks 2.

```
print(xs[-1])     # Negative
indices count from the end of the
list; prints "2"

xs[2] = 'foo'    # Lists can
contain elements of different
types
print(xs)
```

```
xs.append('bar') # Add a new
element to the end of the list
print(xs)

x = xs.pop()     # Remove and
return the last element of the
list
print(x, xs)
```

xs[-1] yaitu list xs dengan indeks minus 1. Dimana, minus menandakan elemen indeksnya dihitung dari kanan ke kiri dan dimulai dari indeks -1, indeks -2 dan seterusnya. Dalam hal ini, indeks minus 1 didalam list xs adalah elemen 2.

xs[2] = ‘foo’ adalah list xs dengan indeks ke-2 akan diinisiasi/digantikan dengan sebuah karakter ‘foo’.

xs.append(‘bar’) ialah perintah untuk menambahkan karakter ‘bar’ pada elemen list xs. perintah append pada list xs menambahkan karakter ‘bar’ pada elemen urutan terakhir di list xs.

x = xs.pop() adalah sebuah varibel biasa dengan nama x yang diisikan dengan sebuah perintah list xs.pop(). Perintah pop akan menghapus elemen yang ada pada urutan terakhir pada list xs namun pop juga mengembalikan elemen tersebut ke x sehingga elemen xs tidak ada karaker ‘bar’ karena dihapus pop.

Sehingga struktur kode-kode list diatas Ketika dijalankan akan menghasikan output sebagai berikut:

- **OUTPUT LISTS**

```
[3, 1, 2] 2
2
[3, 1, 'foo']
[3, 1, 'foo', 'bar']
bar [3, 1, 'foo']
```

2) Membuat Slicing

```
nums = list(range(5))    # range
is a built-in function that
creates a list of integers
print(nums)         # Prints "[0,
1, 2, 3, 4]"
```

Struktur kode diatas adalah struktur kode untuk membuat sebuah list. Namun, Pada dasarnya struktur kode diatas adalah variabel biasa dengan nama *nums* yang diinisiasi/diisi oleh list dengan *range(5)*. *Range(5)* artinya membuat sebuah list dengan jangkauan 5 indeks dengan tipe data integer. Oleh karena itu, list yang dibuat pada source code tersebut memiliki elemen-elemen yang berisi [0, 1, 2, 3, 4] pada variabel *nums*.

Print yang diikuti tanda kurung pada python adalah perintah untuk menampilkan sebuah data yang ada pada tanda kurungnya tersebut. Dalam hal ini, *print(nums)* akan menampilkan semua elemen yang ada didalam list *nums*.

```
print(nums[2:4])    # Get a slice
from index 2 to 4 (exclusive);
prints "[2, 3]"
print(nums[2:])     # Get a slice
from index 2 to the end; prints
"[2, 3, 4]"

print(nums[:2])     # Get a slice
from the start to index 2
(exclusive); prints "[0, 1]"

print(nums[:])      # Get a slice
of the whole list; prints ["0, 1,
2, 3, 4]"

print(nums[:-1])    # Slice
indices can be negative; prints
["0, 1, 2, 3]"

nums[2:4] = [8, 9] # Assign a new
sublist to a slice
print(nums)         # Prints "[0,
1, 8, 9, 4]"
```

pada dasarnya struktur kode-kode diatas digunakan untuk mengambil nilai-nilai elemen pada list yang telah dibuat (*slicing*). Perintah nums[2:4] adalah perintah untuk mengambil elemen yang ada didalam list nums dengan indeks 2 sampai 4. Artinya akan mengambil indeks 2 dan indeks 3 tanpa mengambil indeks 4 karena didalam python indeks terakhir yang dituliskan tidak ikut dieksekusi.

nums[2:] adalah perintah untuk mengambil elemen yang ada didalam list nums dari indeks 2 hingga sampai indeks-indeks seterusnya secara berurutan karena terdapat perintah [:].

nums[:2] adalah perintah untuk mengambil elemen yang ada didalam list nums dari indeks 0 hingga indeks 2 secara berurutan karena terdapat perintah [:]. Namun, indeks terakhir dalam perintah ini yaitu indeks 2 tidak dieksekusi.

nums[:] adalah perintah untuk mengambil semua elemen yang ada didalam list nums dari indeks pertama hingga sampai indeks-indeks seterusnya secara berurutan karena terdapat perintah [:].

nums[:-1] adalah perintah untuk mengambil elemen yang ada didalam list nums dari indeks pertama hingga sampai indeks-indeks seterusnya secara berurutan karena terdapat perintah [:]. Tanda minus dalam list python artinya mengambil nilai dari urutan elemen paling kanan sampai kiri. Dalam hal ini, struktur kode tersebut akan mengambil elemen yang ada didalam list nums dari indeks 0 sampai indeks 3.

nums[2:4] = [8, 9] adalah list nums dengan indeks 2 sampai indeks 3 akan diisi dengan nilai 8 dan 9 dengan menggantikan elemen-elemen sebelumnya yang ada di indeks 2 sampai indeks 3 tersebut.

Sehingga struktur kode-kode *slicing* diatas Ketika dijalankan akan menghasikan output sebagai berikut:

- **OUTPUT SLICING**

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]
```

3) Membuat Loops
   - Program 1

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
```

animals = [‘cat’, ‘dog’, ‘monkey’] adalah struktur kode untuk membuat sebuah list. yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya. Pada dasarnya struktur kode diatas adalah variabel biasa dengan nama *animals* yang diisi oleh list dengan ciri kurung siku yang didalamnya tersimpan beberapa elemen berurutan indeksnya, diantaranya secara berurutan yaitu *cat, dog dan monkey*.

For merupakan jenis perulangan yang countable. Pada dasarnya struktur for diatas ialah sebagai berikut:

```
for variabel in urutan:
      #dalam blok kode for
#keluar blok kode for
```

Variabel dalam deklarasi for di atas digunakan untuk menampung nilai sementara dari tipe data (list). Beralih ke baris di bawahnya, agar kode tersebut bisa diulang nantinya, maka beri ruang atau spasi agar menjorok ke kanan. Dalam hal ini, pada struktur kode diatas sebagai varibel penampungnya adalah animal yang akan menampung elemen-elemen list animals untuk di ulang-ulang secara berurutan dalam blok kode loops.

Sehingga struktur kode-kode *program 1 loops* diatas Ketika dijalankan akan menghasikan output sebagai berikut:

- **OUTPUT PROGRAM 1**

```
cat
dog
monkey
```

- Program 2

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in
enumerate(animals):
    print('#{}: {}'.format(idx +
1, animal))
```

animals = [‘cat’, ‘dog’, ‘monkey’] adalah struktur kode untuk membuat sebuah list. yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya. Pada dasarnya struktur kode diatas adalah variabel biasa dengan nama *animals* yang diisi oleh list dengan ciri kurung siku yang didalamnya tersimpan beberapa elemen berurutan indeksnya, diantaranya secara berurutan yaitu *cat, dog dan monkey*.

Variabel dalam deklarasi for di atas digunakan untuk menampung nilai sementara dari tipe data (list). Beralih ke baris di bawahnya, agar kode tersebut bisa diulang nantinya, maka beri ruang atau spasi agar menjorok ke kanan. Dalam hal ini, pada struktur kode diatas

sebagai varibel penampungnya adalah animal yang akan menampung elemen-elemen list animals untuk di ulang-ulang secara berurutan dalam blok kode loops.

Fungsi enumerate() mengembalikan sebuah objek enumerate yaitu objek iterable (list) yang tiap itemnya memiliki indeks. Karena berjenis iterable, maka itemnya bisa di looping menggunakan for.

Sehingga struktur kode-kode *program 2 loops* diatas Ketika dijalankan akan menghasikan output sebagai berikut:

- **OUTPUT PROGRAM 2**

```
#1: cat
#2: dog
#3: monkey
```

4) Membuat Functions

- Program 1

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'
```

Fungsi blok dimulai dengan def kata kunci diikuti oleh nama fungsi dan tanda kurung (()). Nama fungsinya yaitu sign dengan parameter yang didalam tanda kurung yaitu x.

Didalam blok kode fungsi terdapat kondisi percabangan if-else if-if. Jika if x>0 maka program akan mengeksekusi dan mengembalikan nilai karakter ‘positive’. Jika maka elif x<0 maka program akan mengeksekusi dan mengembalikan nilai karakter ‘negative’. Namun jika kedua kondisi tersebut tidak terpenuhi, maka else yang akan dieksekusi yaitu mengeksekusi dan mengembalikan nilai karakter ‘zero’.

- Program 2

```
for x in [-1, 0, 1]:
    print(sign(x))
```

Variabel x dalam deklarasi for di atas digunakan untuk menampung nilai sementara dari tipe data (list [-1, 0, 1]). Beralih ke baris di bawahnya, agar kode tersebut bisa diulang nantinya, maka beri ruang atau spasi agar menjorok ke kanan. Dalam hal ini, pada struktur kode diatas sebagai variabel penampungnya adalah x yang akan menampung elemen-elemen list [-1, 0, 1] untuk di ulang-ulang secara berurutan dalam blok kode for loops.

Sehingga struktur kode-kode *program 2 functions* diatas Ketika dijalankan akan menghasikan output sebagai berikut:

- **OUTPUT PROGRAM 2**

```
negative
zero
positive
```

- Program 3

```
def hello(name, loud=False):
    if loud:
        print('HELLO,
{}'.format(name.upper()))
    else:
        print('Hello,
{}!'.format(name))

hello('Bob')
hello('Fred', loud=True)
```

Fungsi blok dimulai dengan def kata kunci diikuti oleh nama fungsi dan tanda kurung (()). Nama fungsinya yaitu hello dengan parameter yang didalam tanda kurung yaitu name dan load=False.

Didalam blok kode fungsi terdapat kondisi percabangan if-else. Jika kondisi if terpenuhi maka yang akan dieksekusi adalah blok kode yang ada didalam if. Namun, jika if tidak terpenuhi maka blok kode else yang akan dieksekusi. Upper() adalah perintah untuk menampilkan teks dengan huruf KAPITAL.

nama_fungsi(parameter) adalah bentuk struktur kode untuk memanggil nama fungsi beserta isinya. Dalam hal ini untuk memanggil blok kode fungsi hello yaitu hello(parameter).

Sehingga struktur kode-kode *program 3 functions* diatas Ketika dijalankan akan menghasikan output sebagai berikut:

- **OUTPUT PROGRAM 3**

```
Hello, Bob!
HELLO, FRED
```

V. PENUTUP

5.1 Kesimpulan

Python memang sangat sederhana dibandingkan bahasa yang lainnya. Tidak perlu ini dan itu untuk membuat program Hello World!. Bahkan tagline di websitenya menjelaskan, kalau python akan membuatmu bekerja lebih cepat dan efektif.

Beberapa hal yang menjadi kajian dalam tulisan ini mengenai python dasar diantaranya:

- *Lists* merupakan Setiap elemen-elemen berurutan akan diberi nomor posisi atau indeksnya.
- *Slicing* merupakan teknik memilih data dari sebuah set data.
- *Loops* merupakan teknik untuk perulangan kode program secara terstruktur.
- *Functions* adalah blok kode terorganisir dan dapat digunakan kembali yang digunakan untuk melakukan sebuah tindakan/action.

DAFTAR PUSTAKA

[1] Muhardian, Ahmad. 2016. *"Belajar Pemrograman Python: Memahami Perulangan"*. www.petanikode.com (diakses pada 25 Januari 2021)

[2] Neko, Kuro. 2020. *"Belajar Perulangan di Python"*. www.kopiding.in (diakses pada 25 Januari 2021)

[3] Anonim. 2020. *"Loop, List, Fungsi"*. www.belajarpython.com (diakses pada 25 Januari 2021)

[4] Anonim. 2019. *"Fungsi Enumerate()"*. www.pythonindo.com (diakses pada 25 Januari 2021)

[5] Herlambang, MB. 2018. *"Slicing pada Python"*. www.megabagus.com (diakses pada 25 Januari 2021)