**PRAKTIKUM 1**

**PENGENALAN PYTHON**

* 1. **Tujuan**

1. Mahasiswa dapat mengenal bahasa pemrograman Python
2. Mahasiswa dapat memasang Python pada perangkat komputer masing-masing
3. Mahasiswa dapat menjalankan Python
4. Mahasiswa dapat menampilkan data dan membaca data dari keyboard
   1. **Dasar Teori**
      1. Pengenalan bahasa pemrograman Python

[Python](https://www.python.org/) merupakan bahasa pemrograman interpretatif multiguna. Python lebih mengutamakan pada kemudahan pembacaan kode agar sintaksnya mudah pula dipahami, tidak seperti bahasa pemrograman yang lain. Hal ini dapat mempermudah bagi pemula maupun bagi yang sudah menguasai bahasa pemrograman yang lain.

Bahasa ini dirancang oleh [Guido van Rossum](https://id.wikipedia.org/wiki/Guido_van_Rossum) pada tahun 1991. Saat ini, Python masih dikembangkan oleh [Python Software Foundation](https://www.python.org/psf/). Hampir semua sistem operasi mendukung bahasa Python, bahkan sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya.

Apa yang dapat dilakukan oleh Python :

* Python dapat digunakan pada sisi server untuk membuat aplikasi web.
* Python dapat digunakan untuk mengembangkan perangkat lunak.
* Python dapat digunakan untuk mengkoneksikan sistem basisdata, juga dapat membaca dan memodifikasi file-file.
* Python dapat digunakan untuk menangani big data dan melakukan perhitungan matematika yang kompleks.
* Python dapat digunakan untuk rapid prototyping, atau untuk pengembangan production-ready software.

Mengapa harus belajar Python :

* Selain, mendukung hampir semua platform yang berbeda, Python juga memiliki aturan penulisan (sintaks) yang sederhana dan mirip dengan bahasa Inggris.
* Python memiliki sintaks yang memungkikan pengembang menulis program dengan baris-baris program yang lebih sedikit dibandingkan dengan bahasa pemrograman yangl ain.
* Python merupakan bahasa dengan metode permrograman interpreter, artinya kode dapat dieksekusi secepat kode itu ditulis.
* Python dapat ditangani secara prosedural, atau object oriented atau secara fungsional.
  + 1. Pemasangan/Instalasi Python

Saat ini python memiliki dua versi yang berbeda, yaitu Phyton versi 3 dan versi 2. Phyton yang digunakan pada panduan praktikum ini adalah versi 3. Cara pemasangan pada paduan praktikum ini hanya untuk sistem operasi windows.

1. Buka browser, kunjungi <https://www.python.org/downloads/windows/>
2. ATAU, klik direct link <https://www.python.org/ftp/python/3.8.2/python-3.8.2.exe>
3. Buka (klik 2x) file installer python yang baru saja di download
4. Ikuti langkah instalasi sampai selesai
   * 1. Menjalankan Python

Python dapat dijalankan dengan banyak cara, yakni bisa menggunakan sheel, terminal atau menggunakan [IDE (Integrated Development Environment)](http://www.belajarpython.com/2015/05/integrated-development-environment-ide.html). Di bawah ini adalah langkah-langkah menjalankan Python dengan cara yang paling mudah. Berikut cara menjalankan Python dengan sistem operasi windows :

1. Buka Python sheel, Anda bisa mencarinya di tombol START.
2. Tuliskan script Python Anda, jika sudah tekan tombol ENTER, dan script Python akan dijalankan/eksekusi.
3. Untuk keluar dari sheel Python ketik exit();
   * 1. Menampilkan data dan membaca data dari keyboard

Dalam Python 3, data ditampilkan dengan menggunakan fungsi print(), dimana sesuatu yang akan dicetak harus diletakkan diantara kurung buka dan kurung tutup. Jika ingin mencetak tipe data string (dijelaskan selanjutnya) langsung, maka harus memasukkannya ke dalam tanda kutip terlebih dahulu.

Saat script di atas dijalankan, maka akan terlihat keluaran berupa teks

print("Selamat Datang")

Selamat Datang

Beberapa data yang akan ditampilkan dapat dipisah menggunakan tanda koma, seperti berikut :

print("Selamat Datang","di Python")

Data-data yang ditampilkan dapat menggunakan format-format tertentu. Terdapat empat format yang sering digunakan, yaitu dapat dilihat pada tabel 1.1 berikut ini :

Tabel 1.1 Penanda format

|  |  |
| --- | --- |
| Penanda Format | Deskripsi |
| %s | Menampilkan data teks |
| %d | Menampilkan data bilangan bulat |
| %f | Menampilkan data bilangan riil |
| %c | Menampilkan data karakter |

Selain itu pula, dapat memanfaatkan method format() yang didefinisikan di dalam str. Contoh penggunaanya sebagai berikut :

print("Sintaks {0}{1}".format("Python”, 3))

Contoh di atas, parameter {0} akan diisi oleh “Python” dan parameter {1} akan diisi dengan bilangan 3.

Selain menggunakan bentuk-bentuk di atas, pembentukan format string juga dapat dilakukan menggunakan kode berikut :

print("Sintaks %(bahasa)s %(versi)d" %{"bahasa":"Python","versi":3})

Dalam keadaan normal (default), fungsi print() akan selalu menyertakan karakter baris baru ("\n") setelah selesai menampilkan data. Dengan demikian data yang ditampilkan berikutnya akan berada di bawah data sebelumnya. Jika ingin menghilangkan karakter baris baru, sertakan parameter end yang berisi karakter kosong.

print("Hello", end=" "); print("Python", end=" "); print(3);

Melalui teknik di atas, Python tidak akan membuat baris baru setelah menampilkan data pertama dan data kedua. Maka dari itu data kedua dan ketiga akan ditampilkan di sebelah kanan.

Python mendukung adanya karakter escape sequence yang disisipkan ke dalam suatu string. Karakter escape sequence merupakan runtunan dua karakter (atau lebih) yang memiliki makna khusus, dan selalu diawali dengan karakter backslash ("\").

Tabel 1.2 menunjukkan beberapa karakter escape sequence yang sering digunakan dalam pemrograman

Tabel 1.1 Karakter escape sequence

|  |  |
| --- | --- |
| Escape Sequence | Deskripsi |
| \\ | Backslash (\) |
| \' | Petik tunggal |
| \" | Petik ganda |
| \b | Backspace |
| \n | ASCII Linefeed (LF) |
| \r | ASCII Carriage Return (CR) |
| \t | ASCII tab (TAB) |
| \v | ASCII vertical tab (VT) |

Python 3 membaca data masukan (input) dari keyboard menggunakan fungsi input().

nama = input("Siapa nama Anda ? ")

Dengan cara ini, variabel nama akan menampung data yang dimasukkan melalui keyboard. Isi data dari variabel tersebut dapat ditampilkan menggunakan fungsi print(). Fungsi input() selalu menghasilkan nilai bertipe str (string). Ini berarti bahwa setiap data yang dimasukkan melalui keyboard akan dianggap sebagai teks, meskipun pengguna program mengetikkan karakter angka.

s = input("masukkan bilangan bulat : ")

Pada contoh di atas variabel s akan bertipe str, maka agar isian data yang dimasukkan bisa diubah menjadi bilangan maka data inputan tadi perlu diubah ke tipe data bilangan misalnya int (bilangan bulat).

s = input("masukkan bilangan bulat : ")

s = int(s)

Variabel s pada baris kedua di sebelah kiri tanda sama dengan sudah bertipe int. atau dapat dilakukan dengan bentuk yang lain, misalnya seperti contoh berikut :

s = int(input("masukkan bilangan bulat : "))

Python bersifat case sensitive, yang berarti bahwa huruf besar dan huruf kecil memiliki perbedaan.

* 1. **Kegiatan Praktek**

Ketiklah script berikut pada Python Shell anda, dan jalankan programnya. jika ada error perbaikilah, kemudian catatlah hasil yang anda dapatkan :

Program #1

print("Selamat Datang")

print("Ini adalah program pertama saya")

If 3 > 1:

Print("Tulisan ini akan benar-benar tercetak ke layar")

print("Demikian juga tulisan ini")

Program #2

#ini adalah komentar

"""

ini juga adalah komentar, jika lebih dari satu baris

"""

(Hallo Dunia)

Tulisan ini tidak boleh tampil ke layar

* 1. **Tugas Mandiri**

1. Berilah penjelasan hasil yang anda dapatkan pada kegiatan praktek
2. Buatlah program untuk mencetak ke layar tulisan seperti yang ditunjukkan berikut ini :

Saya adalah Mahasiswa

Teknik Elektro

Fakultas Teknik

Universitas Negeri Gorontalo

Angkatan 2019

Sebagai catatan, tahun pada tulisan angkatan dimasukkan melalui keyboard dan tipe datanya harus berupa bilangan.

1. Ubahlah program pada nomor 1 untuk menulis keluaran tulisan di atas, namun hanya menggunakan satu baris perintah saja.
2. Tambahkan komentar di bagian akhir awal dan bagian akhir dari program

**PRAKTIKUM 2**

**VARIABEL, TIPE DATA, DAN OPERATOR**

* 1. **Tujuan**

1. Mahasiswa dapat memahami penggunaan variabel dari Python
2. Mahasiswa dapat mengetahui tipe-tipe data dalam bahasa Python
3. Mahasiswa dapat memahami dan menggunakan operator pada bahasa Python
   1. **Dasar Teori**
      1. Variabel

Variabel adalah container atau tempat untuk penyimpanan data. Tidak seperti bahasa pemrograman yang lain, Python tidak memiliki perintah untuk mendeklarasikan sebuah variabel.

Terdapat beberapa ketentuan variabel dalam bahasa Python :

1. Sebuah variabel dibuat saat pertama kali sebuah nilai ditetapkan pada variabel tersebut.
2. Variabel tidak perlu dideklarasikan dengan berbagai cara dan dapat berubah jenisnya setelah variabel tersebut ditetapkan
3. Variabel string dapat dideklarasikan dapat juga dideklarasikan menggunakan tanda petik dua atau petik satu
4. Sebuah variabel dapat memiliki suatu nama singkat (seperti x dan y) atau nama yang lebih mendeskripsikan (umur, namamobil, total\_volume). Aturan penulisan nama untuk variabel-variabel Python sebagai berikut :

* Sebuah nama variabel harus diawali dengan sebuah huruf atau karakter garis bawah (underscore).
* Sebuah nama variabel tidak boleh diawali dengan angka atau bilangan.
* Sebuah nama variabel dapat berisi karakter alpha-numeric dan garis bawah (A – z, 0 – 9, dan \_ )
* Nama-nama variabel adalah case-sensitive (batade, Batade dan BATADE adalah tiga buah variabel yang berbeda).

1. Python memungkinkan untuk menetapkan nilai-nilai ke dalam beberapa variabel dalam satu baris.
2. Penetapan nilai yang sama ke dalam beberapa variabel yang berbeda juga dimungkinkan.

Contoh :

Program #1

x = 5

y = "Batade"

print(x)

print(Y)

x = 4 # x adalah tipe int  
x = "wadala" # x adalah tipe str  
print(x)

x = "Tambalakada"  
# adalah sama dengan  
x = 'Tambalakada'

myvar = "Corona covid-19"  
my\_var = "Corona covid-19"  
\_my\_var = "Corona covid-19"  
myVar = "Corona covid-19"  
MYVAR = "Corona covid-19"  
myvar2 = "Corona covid-19"  
2myvar = "Corona covid-19"  
my-var = "Corona covid-19"  
my var = "Corona covid-19"

x = y = z = "kukudu"  
print(X)  
print(y)  
print(Z)

x, y, z = "Teknik", "Elektro", "FT-UNG"  
print(x)  
print(y)  
print(z)

Pernyataan print sering digunakan untuk menuliskan keluaran dari variabel-variabel. Penggabungan dari teks dan sebuah variabel, Python menggunakan karakter + . Penggunaan karakter + juga untuk menjumlahkan atau menambahkan sebuah variabel ke variabel yang lain. Karakter + pada bilangan atau angka berfungsi sebagai operator matematika. Penggabungan angka/bilangan dengan sebuah string akan menghasilkan error.

Contoh :

Program #2

x = "mantul"  
print("Python " + x)print(x)

x = "Python "  
y = "mantul"  
z =  x + y  
print(z)

x = 5  
y = 10  
print(x + y)

x = 5  
y = "John"  
print(x + y)

Variabel-variabel yang dibuat diluar dari sebuah fungsi (contoh berikutnya), dikenal sebagai variabel-variabel global. Variabel global dapat digunakan, baik di dalam maupun di luar fungsi tersebut. Apabila sebuah variabel dibuat dengan nama yang sama di dalam suatu fungsi, variabel ini akan bersifat lokal, dan hanya dapat digunakan di dalam fungsi. Variabel global dengan nama yang sama akan ditetapkan sebagai variabel yang pertama kali dideklarasikan dengan nilai awal. Secara normal, ketika sebuah variabel dideklarasikan di dalam fungsi, variabel tersebut akan bersifat lokal, dan hanya dapat digunakan di dalam fungsi tersebut. Untuk membuat sebuah variabel global dalam sebuah fungsi, dapat digunakan kata kunci global . Penggunaan kata kunci global bisa juga dilakukan untuk mengganti variabel global yang ada dalam fungsi.

Program #3

x = "mantul"  
def myfunc():  
print("Python " + x)  
myfunc()

Program #4

x = "mantul"  
def myfunc():  
x = "mantap"  
print("Python " + x)  
myfunc()  
print("Python " + x)

Program #4

x = "mantul"  
def myfunc():  
x = "mantap"  
print("Python " + x)  
myfunc()  
print("Python " + x)

Program #5

x = "mesin"  
def myfunc():  
  global x  
  x = "Elektro"  
myfunc()  
print("Teknik " + x)

* + 1. Tipe Data

Tipe data dalam pemrograman merupakan konsep yang penting. Variabel dapat disimpan dalam tipe data-tipe data yang berbeda, dan tipe data yang berbeda dapat digunakan untuk keperluan yang berbeda pula. Python memiliki tipe data *built-in* (bawaan) berikut :

|  |  |
| --- | --- |
| Tipe Teks | str |
| Tipe numeric | int, float, complex |
| Tipe sequence (terurut) | list, tuple, range |
| Tipe mapping | dict |
| Tipe set | set, frozenset |
| Tipe Boolean | bool |
| Tipe Biner | bytes, bytearray, memoryview |

Tipe data dapat digunakan dengan menuliskan fungsi type() :

Program #6

Dalam Python, tipe data adalah set atau himpunan ketika ditetapkan sebuah nilai ke dalam sebuah variabel :

x = 6  
print(type(x))

|  |  |
| --- | --- |
| x = "selamat datang" | str |
| x = 100 | int |
| x = 90.5 | Float |
| x = 2j | complex |
| x = ["pepaya","mangga", "pisang","jambu","wacaca"] | list |
| x = ("pepaya","mangga", "pisang","jambu","wacaca") | tuple |
| x = range(10) | range |
| x = {"nama":"tukiyem", "umur":"20"} | dict |
| x = {"pepaya","mangga","pisang","jambu","wacaca"} | set |
| x = frozenset({"pepaya","mangga", "pisang","jambu"}) | frozenset |
| x = True | bool |
| x = b"pepaya" | bytes |
| x = bytearray(4) | bytearray |
| x = memoryview(bytes(4)) | memoryview |

Jika diinginkan untuk menspesifikasikan tipe data, dapat dilakukan dengan menggunakan fungsi berikut :

|  |  |
| --- | --- |
| x = str("selamat datang") | str |
| x = int(100) | int |
| x = float(90.5) | Float |
| x = complex(2j) | complex |
| x = list(("pepaya","mangga", "pisang","jambu")) | list |
| x = tuple(("pepaya","mangga", "pisang","jambu")) | tuple |
| x = range(10) | range |
| x = dict(nama="tukinem", umur:20) | dict |
| x = set(("pepaya","mangga","pisang","jambu")) | set |
| x = frozenset(("pepaya","mangga", "pisang","jambu")) | frozenset |
| x = bool(5) | bool |
| x = bytes(5) | bytes |
| x = bytearray(4) | bytearray |
| x = memoryview(bytes(4)) | memoryview |

Program #7

#tipe data Boolean

print(True)

#tipe data String

print("Ayo belajar Python")

print('Belajar Python Sangat Mudah')

#tipe data Integer

print(20)

#tipe data Float

print(3.14)

#tipe data Hexadecimal

print(9a)

#tipe data Complex

print(5j)

#tipe data List

print([1,2,3,4,5])

print(["satu", "dua", "tiga"])

#tipe data Tuple

print((1,2,3,4,5))

print(("satu", "dua", "tiga"))

#tipe data Dictionary

print({"nama":"Langgeng", 'umur':19})

#tipe data Dictionary dimasukan ke dalam variabel biodata

biodata = {"nama":"Widodo", 'umur':19} #proses inisialisasi variabel biodata

print(biodata) #proses pencetakan variabel biodata yang berisi tipe data Dictionary

type(biodata) #fungsi untuk mengecek jenis tipe data. akan tampil <class 'dict'> yang berarti dict adalah tipe data dictionary

* + 1. Operator

Operator digunakan untuk melakukan operasi pada variabel-variabel dan nilai-nilai. Operator pada Python dibagi menjadi beberapa kelompok :

1. Operator Aritmetika

Operator aritmetika digunakan untuk melakukan operasi matematika secara umum. Operator-operator aritmetika ini disuguhkan pada tabel 2.1 berikut :

Tabel 2.1 Operator-Operator Aritmetika

|  |  |
| --- | --- |
| **Operator** | **Nama** |
| + | Penjumlahan |
| - | Pengurangan |
| \* | Perkalian |
| / | Pembagian |
| % | Modulus |
| \*\* | Pemangkatan |
| // | Pembagian Floor |

1. Operator Assignment

Operator assignment adalah operator untuk menetapkan nilai-nilai ke dalam variabel-variabel. Kelompok operator ini ditampilkan dalam tabel 2.2 berikut ini :

Tabel 2.1 Operator-Operator *Assignment*

|  |  |  |
| --- | --- | --- |
| **Operator** | **Contoh** | **Contoh yang sama** |
| = | x = 5 | x = 5 |
| += | x += 2 | x = x + 2 |
| -= | x -= 2 | x = x – 2 |
| \*= | x \*= 2 | x = x \* 2 |
| /= | x /= 2 | x = x / 2 |
| %= | x %= 2 | x = x % 2 |
| //= | x //= 2 | x = x // 2 |
| &= | x &= 2 | x = x & 2 |
| |= | x |= 2 | x = x | 2 |
| ^= | x ^= 2 | x = x ^ 2 |
| >>= | x >>= 2 | x = x >> 2 |
| <<= | x <<= 2 | x = x >> 2 |

1. Operator Pembanding

Operator pembanding digunakan untuk membandingkan dua nilai. Tabel 2.3 menyajikan operator-operator pembanding.

Tabel 2.3 Operator-Operator Pembanding

|  |  |  |
| --- | --- | --- |
| **Operator** | **Nama** | **Contoh** |
| == | Sama dengan | x == y |
| != | Tidak sama dengan | x != y |
| > | Lebih besar | x > y |
| < | Lebih kecil | x < y |
| >= | Lebih besar atau sama dengan | x >= y |
| <= | Lebih kecil atau sama dengan | x <= y |

1. Opertator Logika

Operator Logika digunakan untuk mengkombinasikan pernyataan-pernyataan kondisional. Operator-operator ini disajikan dalam tabel 2.4.

Tabel 2.4 Operator-Operator Logika

|  |  |  |
| --- | --- | --- |
| **Operator** | **Deskripsi** | **Contoh** |
| and | Menghasilkan kondisi benar (true) jika **kedua pernyataan** benar | x < 5 and x < 10 |
| or | Menghasilkan kondisi benar (true) jika **salah satu pernyataan** benar | x < 5 or x < 4 |
| not | Berlawanan dengan masukan, hasil salah (false) jika masukkan benar | Not(x < 5 and x < 10) |

1. Operator Identitas

Operator indentitas digunakan untuk membandingkan objek, jika tidak sama dengan, tetapi keduanya adalah merupakan objek yang sama. Tabel 2.5 menampilkan operator-operator identitas.

Tabel 2.5 Operator-Operator Pembanding

|  |  |  |
| --- | --- | --- |
| **Operator** | **Deskripsi** | **Contoh** |
| is | Menghasilkan kondisi benar jika **kedua variabel** adalah objek yang **sama** | x is y |
| is not | Menghasilkan kondisi benar **kedua variabel tidak sama** | x is not y |

1. Operator Keanggotaan

Operator-operator keanggotaan digunakan untuk menguji jika suatu urutan dinyatakan dalam sebuah objek. Tabel 2.6 menyajikan operator-operator keanggotaan

Tabel 2.6 Operator-Operator Keanggotaan

|  |  |  |
| --- | --- | --- |
| **Operator** | **Deskripsi** | **Contoh** |
| in | Menghasilkan benar jika satu urutan dengan nilai tertentu dinyatakan dalam objek | x in y |
| not in | Menghasilkan benar jika satu urutan dengan nilai tertentu tidak dinyatakan dalam objek | x not in y |

1. Operator Bitwise

Operator-operator bitwise digunakan untuk membandingkan angka/nilai/bilangan biner. Operator-operator ini ditampilkan pada tabel 2.7.

Tabel 2.7 Operator-Operator Bitwise

|  |  |  |
| --- | --- | --- |
| **Operator** | **Nama** | **Deskripsi** |
| & | AND | tetapkan setiap bit adalah 1 jika kedua bit adalah 1 |
| | | OR | tetapkan setiap bit adalah 1 jika satu dari dua bit adalah 1 |
| ^ | XOR | tetapkan setiap bit adalah 1 jika hanya satu dari dua bit adalah 1 |
| ~ | NOT | Kebalikan dari semua bit |
| << | Nol digeser ruas kiri | Mengeser satu digit ke kiri dengan menambahkan nol pada ujung sebelah paling kanan dan menghilangkan satu digit paling kiri. |
| >> | Ruas kanan bertanda | Mengganti bit paling kanan dengan bit paling kiri |

* 1. **Kegiatan Praktek**

1. Ketiklah script program yang ada pada contoh program dalam bagian 2.2 pada Python Shell anda.
2. Kemudian jalankan,
3. Jika ada yang error, perbaiki errornya,
4. Catatlah hasil yang anda dapatkan
   1. **Tugas Mandiri**
   2. Tulislah program sederhana untuk membuktikan penggunaan operator aritmetika, operator pembanding, operator logika, operator assignment, operator identitas, dan operator keanggotaan yang telah disajikan pada bagian 2.2.3.
   3. Berikan penjelasan jika diperlukan
   4. Berikan penjelasan apa itu Python Casting, tulislah contoh programnya
   5. Berikan penjelasan lebih mengenai penggunaan operator bitwise. Buatlah contohnya dalam program sederhana.

**PRAKTIKUM 3**

**LIST, TUPLE, SET DAN DICTIONARY**

* 1. **Tujuan**

1. Mahasiswa dapat memahami penggunaan list dari Python
2. Mahasiswa dapat memahami penggunaan tuple bahasa Python
3. Mahasiswa dapat memahami penggunaan set bahasa Python
4. Mahasiswa dapat memahami penggunaan set bahasa Python
   1. **Dasar Teori**
      1. List

List adalah suatu himpunan yang diurutkan dan dapat diubah. Dalam Python list ditulis berurutan yang ditempatkan dalam kurun siku.

ini\_list = ["pepaya", "mangga", "pisang", "jambu", "wacaca"]

Cara mengakses item listnya dengan mengacu pada nomor indeksnya.

Contoh

Program #1

ini\_list = ["pepaya", "mangga", "pisang", "jambu", "wacaca"]

print(ini\_list[1])

Penentuan nomor indeks dimulai dari 0, jadi untuk contoh pada program 1, hasil yang akan ditampilkan ke layar adalah mangga. Nomor indeks jika dimasukkan nilai negatif, contoh misalnya nomor indeks pada program 1 di atas diganti dengan -1, maka yang akan tercetak di layar adalah wacaca.

Pengaksesan nomor indeks dapat ditentukan jangkauannya, dengan menentukan nilai awal dan nilai akhir. Dengan menentukan jangkauan tertentu, Python akan menampilkan list baru sesuai dengan jangkauan yang diberikan. Namun sebagai catatan, bahwa nilai yang diakses itu adalah termasuk nomor indeks awal tapi tidak termasuk nilai yang ditunjuk oleh nomor indeks akhir. Jika nomor indeks awal tidak ditulis, dan hanya nilai indeks akhir yang ditulis, maka hasil yang akan ditampilkan adalah semua nilai dari nomor indeks 0 sampai nomor indeks sebelum nomor indeks yang ditulis. Contohnya dapat dilihat pada program 3. Demikian pula sebaliknya (program 4). Pengaksesan nomor indeks negatif ditunjukkan oleh program 5.

Contoh

Program #2

ini\_list = ["pepaya", "mangga", "pisang", "jambu", "wacaca"]

print(ini\_list[2:4])

Program #3

ini\_list = ["pepaya", "mangga", "pisang", "jambu", "wacaca"]

print(ini\_list[:4])

Program #4

ini\_list = ["pepaya", "mangga", "pisang", "jambu", "wacaca"]

print(ini\_list[1:])

Program #5

ini\_list = ["pepaya", "mangga", "pisang", "jambu", "wacaca"]

print(ini\_list[-1:-4])

Pengubahan nilai dari item tertentu, juga didasarkan pada nomor indeks. Contohnya dilihat pada program 6. Untuk mengambil semua nilai dari list dapat digunakan perintah pengulangan for (contoh program 7).

Program 8 adalah contoh untuk mengecek apakah terdapat item tertentu dalam list menggunakan kata kunci in. Selanjutnya, berapa banyak item yang ada dalam list digunakan fungsi len(), seperti yang dicontohkan oleh program 9.

Contoh dalam program 10 menampilkan jika Item baru ingin ditambahkan ke dalam list yakni dengan menggunakan method append(). Demikian juga, jika ingin menambahkan item baru dapat dapat digunakan method insert(), seperti dalam program 11. Penambahan item dengan metode insert() ini harus menuliskan nomor indeks ke berapa item ini akan ditambahkan.

Selain item list dapat diakses, diubah, ditambah, juga dapat dihapus. Penghapusan item dari list digunakan beberapa metode, yaitu remove(), pop() dan clear(). Disamping, kedua method juga item list dapat dihapus dengan menggunakan kata kunci del. Masing-masing method dan kata kunci untuk penghapusan item list dapat dilihat berurutan pada program 12, 13, 14 dan 15.

Item list dapat disalin dengan menggunakan method built-in (bawaan) yakni copy() dan list(). Penggunaan kedua method ini dituliskan dalam program 16 dan 17. Seterunya, list-list dalam Python dapat digabung dengan menggunakan operator + (program 18). Selain operator + ini, penggabungan list-list Python dapat dilakukan dengan beberapa metodh yaitu append() dan extend() (program 19 dan 20).

Contoh

Program #6

ini\_list = ["pepaya", "mangga", "pisang", "jambu", "wacaca"]

ini\_list([1]) = "kacang"

print(ini\_list)

Program #7

Ini\_list = ["pepaya", "mangga", "pisang"]  
for x in ini\_list:  
print(x)

Program #8

Ini\_list = ["pepaya", "mangga", "pisang"]  
if "pepaya" in ini\_list:  
print("ada itu pepaya bro di dalam list ")

Program #9

Ini\_list = ["pepaya", "mangga", "pisang"]  
print(len((ini\_list))

Program #10

Ini\_list = ["pepaya", "mangga", "pisang"]

ini\_list.append("jambu")  
print(ini\_list)

Program #11

Ini\_list = ["pepaya", "mangga", "pisang"]  
ini\_list.insert(1,"jambu")  
print(ini\_list)

Program #12

Ini\_list = ["pepaya", "mangga", "pisang"]  
ini\_list.remove("jambu")  
print(x)

Program #13

Ini\_list = ["pepaya", "mangga", "pisang"]  
ini\_list.pop()  
print(ini\_list)

Program #14

Ini\_list = ["pepaya", "mangga", "pisang"]  
ini\_list.clear()  
print(ini\_list)

Program #15

Ini\_list = ["pepaya", "mangga", "pisang"]  
del ini\_list:

Program #16

Ini\_list = ["pepaya", "mangga", "pisang"]  
mylist = thislist.copy()  
print(mylist)

Program #17

Ini\_list = ["pepaya", "mangga", "pisang"]  
mylist = list(thislist)  
print(mylist)

Program #18

list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
list3 = list1 + list2  
print(list3)

Program #19

list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
for x in list2:  
  list1.append(x)  
print(list1)

Program #20

list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
list1.extend(list2)  
print(list1)

Terdapat sebuah method construktor untuk membuat list baru yaitu list(). Contohnya ditunjukkan dalam program 21.

Contoh :

Program #21

thislist = list(("ayam", "bebek", "lele")) # dua tanda kurung ganda  
print(thislist)

Tabel 4.1 menyajikan beberapa method built-in lain yang dapat digunakan pada list

Tabel 4.1 Method built-in Python untuk digunakan pada list

|  |  |
| --- | --- |
| **Method** | **Deskripsi** |
| reverse() | Membalik urutan dari list |
| sort() | Memilih list |

* + 1. Tuple

Tuple adalah kumpulan/himpunan yang terurut dan tidak dapat diubah. Tuple Python ditulis dengan kurung biasa (program 22). Program 23 menampilkan cara untuk mengakses item-item tuple dengan menggunakan nomor indeks, di dalam tanda kurung siku. Nomor indeks negatif -1 menyatakan item yang paling akhir (dari kanan). Nomor indeks -2 berarti item kedua dari kanan dan seterusnya. Pengaksesan tuple hampir sama dengan list, baik pengasesan secara keseluruhan isi tuple, peritem. Namun, yang berbeda tuple tidak bisa ditambah isinya.

Contoh :

Program #22

thistuple = ("pepaya", "mangga", "pisang", "jambu", "wacaca")  
print(thistuple)

Program #23

thistuple = ("pepaya", "mangga", "pisang", "jambu", "wacaca")  
print(thistuple)

Sekali tuple dibuat, nilainya tidak bisa diubah. Tuple bersifat immutable yang berarti tetap. Namun, dalam prakteknya untuk mengubah itemnya dapat dilakukan dengan mengubah tuple menjadi list, setelah itemnya diubah kemudian list tersebut diubah lagi menjadi tuple. Cara mengubah tuple diperlihatkan oleh program 24.

Contoh :

Program #24

x = ("pepaya", "mangga", "pisang")  
y = list(x)  
y[1] = "jambu"  
x = tuple(y)  
print(x)

Tuple dengan satu item dapat dibuat dengan meletakkan tanda koma setelah item. Jika tanda koma dihilangkan maka Python tidak akan mengenalinya sebagai tuple. Tuple tidak bisa dihapus itemnya, namun dapat dihapus secara keseluruhan. Untuk menghapus tuple dapat dilakukan dengan kata kunci del. Sama halnya dengan list, tuple juga dapat diterapkan konstruktor tuple() untuk membuat sebuah tuple.

Program #25

thistuple = ("pepaya", "mangga", "pisang")  
del thistuple  
print(thistuple)

Tabel 4.2 menyajikan beberapa method built-in lain yang dapat digunakan pada tuple

Tabel 4.2 Method built-in Python untuk digunakan pada tuple

|  |  |
| --- | --- |
| **Method** | **Deskripsi** |
| count() | Menghasilkan angka/jumlah berapa nilai tertentu yang ada pada tuple |
| index() | Mencari tuple untuk nilai tertentu dan menghasilkan posisi dimana nilai tersebut ditemukan. |

* + 1. Set

Sebuah set adalah himpunan yang tidak terurut dan tidak terindeks. Dalam Python set ditulis dengan kurung kurawal. Contoh :

Program #26

thisset = {"pepaya", "mangga", "pisang"}  
print(thisset)

Set adalah himpunan yang tidak terurut sehingga tidak bisa dipastikan urutan item yang akan ditampilkan oleh Python. Pengaksesan item dari set tidak bisa dilakukan dengan menggunakan indeks, karena set tidak memiliki nomor indeks. Tetapi, untuk mengambil isi keseluruhan dapat dilakukan dengan menggunakan perintah for. Item yang ada dalam set dapat diperiksa keberadaannya dengan menggunakan kata kunci in. Sintaks for dan in sama dengan penggunaannya seperti pada list dan tuple.

Penambahan satu item dalam sebuah set dapat digunakan method add(), serta untuk menambah lebih dari satu item dapat digunakan method update(). Contoh untuk kedua method ini ditunjukkan oleh program 27 dan 28. Untuk mengetahui ada berapa banyak item dalam set digunakan method len().

Penghapusan item dalam sebuah set digunakan method remove() atau method discard(). Selain kedua method penghapusan item dapat digunakan method pop(), namun method ini akan menghapus item paling terakhir dari set. Ada pula method untuk menghapus item set yaitu kata kunci del. Kata kunci del berfungsi sama dengan method clear(). Kedua cara terakhir ini akan menghapus semua isi set.

Penggabungan dua set atau lebih bisa digunakan method union() (program 29) dan method update() (prorgram 30). Sebagai catatan, kedua method penggabungan set ini akan mengecualikan duplikasi item.

Program #27

thiset = ("pepaya", "mangga", "pisang")  
thisset.add("jambu")  
print(thisset)

Program #28

thisset = ("pepaya", "mangga", "pisang")  
thisset.update("mangga", "wacaca")  
print(thisset)

Program #29

set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
set3 = set1.union(set2)  
print(set3)

Program #30

set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
set1.update(set2)  
print(set1)

Tabel 4.3 menyajikan beberapa method built-in lain yang dapat digunakan pada set

Tabel 4.3 Method built-in Python untuk digunakan pada set

|  |  |
| --- | --- |
| **Method** | **Deskripsi** |
| difference() | Menghasilkan sebuah set yang berisi perbedaan antara dua atau lebih set |
| difference\_update() | Menghapus item dalam set yang ada juga di set yang lain, set yang ditentukan |
| discard() | Menghapus item tertentu |
| intersection() | Menghasilkan sebuah set, dimana merupakan irisan dari dua buah set |
| intersection\_update() | Menghapus item dalam suatu set yang tidak ada dalam set yang lain, dengan set yang ditentukan. |
| isdisjoint() | Menghasilkan apakah dua set memiliki irisan atau tidak |
| issubset() | Menghasilkan apakah set yang lain berisi set ini atau tidak |
| issuperset() | Menghasilkan apakah set yang ini berisi set lain atau tidak |
| symmetric\_difference() | Menghasilkan sebuah set dengan difference simetris dari dua set |
| symmetric\_difference\_update() | Mengisi difference simetris dari set ini dan set yang lain |

* + 1. Dictionary

Sebuah dictionary adalah sebuah himpunan/kumpulan yang tidak terurut, dapat diubah dan terindeks. Dalam Python, dictionary ditulis dengan tanda kurung kurawal dan memiliki kunci-kunci dan nilai-nilai. Contoh penulisan dictionary dapat dilihat pada program 31.

Program #31

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict)

Pengaksesan item dari sebuah dictionary didasarkan pada nama kunci, yang ditulis dengan tand kurung siku. Pengaksesan ada dua cara dengan kunci misalnya “model” dan dengan method get(). Kedua cara ini memberikan hasil yang sama (program 32).

Program #32

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

x = thisdict["model"]

y = thisdict.get("model")  
print(x)

print(y)

Nilai-nilai dari item-item tertentu dapat diubah berdasarkan nama kuncinya. Contohnya dalam program 33, misalnya “year” diubah menjadi 2020.

Program #33

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

thisdict["year"] = 2020

print(thisdict)

Pengulangan for digunakan untuk mengambil item-item dictionary. Saat melakukan pengulangan ke dalam dictionary, nilai kembalian adalah kunci-kunci dari dictionary. Namun, ada metode yang lain yang dapat digunakan untuk mengakses nila-nilainya. Program 34 menunjukkan bagaiman menggunakan pengulangan for dan cara menampilkan hasilnya. Program 35 menampilkan contoh penggunaan method value() untuk mengambil nilai, sedangkan program 36 menampilkan contoh method items() bagaimana mengambil kunci bersama nilainya.

Pengecekan apakah terdapat kunci tertentu dalam sebuah dictionary digunakan kata kunci in. penggunaan kata kunci ini sama dengan ketiga bagian yang sudah dijelaskan sebelumnya. Demikian pula untuk method len() yang digunakan untuk menghitung berapa banyak pasangan kunci dan nilai yang terdapat dalam dictionary.

Program #34

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

for x in thisdict:  
  print(x)

print (thisdict[x])

Program #35

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

for x in thisdict.values():  
  print(x)

Program #36

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

for x, y in thisdict.items():  
  print(x, y)

Penambahan sebuah item pada dictionary dilakukan dengan menggunakan sebuah indeks kunci dan ditetapkan sebuah nilai padanya. Contohnya seperti yang dituliskan dalam program 37. Program 38 adalah contoh untuk menghapus item dari sebuah dictionary, dengan menggunakan metod pop(). Method pop() ini berguna untuk menghapus item dengan spesfikasi nama kunci yang diberikan. Ada juga method popitem(), yaitu method yang digunakan untuk menghapus item terakhir yang dimasukkan. (jika anda menggunakan versi 3.7 ke bawah, yang dihapus oleh method ini adalah item secara acak). Method popitem() ini ditunjukkan pada program 39. Cara lain untuk melakukan penghapusan adalah kata kunci del (program 40 dan 41). Kata kunci del ini digunakan untuk menghapus dictionary secara keseluruhan atau menghapus hanya suatu kunci yang dapat diberikan secara spesifik. Selain itu pula, ada method yang berguna untuk mengosongkan dictionary yaitu clear(). Penggunaan method clear() ini dapat dilihat pada program 42.

Program #37

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

thisdict["color"] = "red"  
print(thisdict)

Program #38

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

thisdict.pop("model")  
print(thisdict)

Program #39

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

thisdict.popitem()  
print(thisdict)

Program #40

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

del thisdict["model"]  
print(thisdict)

Program #41

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

del thisdict  
print(thisdict) #this will cause an error because "thisdict" no longer exists.

Program #42

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

thisdict.clear()  
print(thisdict)

Sebuah dictionary disalin dengan menggunakan method copy(). Penggunaan method ini dituliskan dalam program 43. Method lain untuk menyalin sebuah dictionary adalah method built-in dict() (program 44).

Program #43

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

mydict = thisdict.copy()  
print(mydict)

Program #44

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

mydict = dict(thisdict)  
print(mydict)

Sebuah dictionary dapat juga berisi beberapa dictionary, konsep ini disebut nested dictionary (dictionary bersarang). Nested dictionary ini dicontohkan dalam program 45, dimana sebuah dictionary berisi tiga buah dictionary. Atau, jika diinginkan untuk menyarangkan tiga buah dictionary yang sudah ada juga dapat dilakukan dengan cara seperti yang ditunjukkan oleh program 46. Seperti halnya, list, tuple dan set, dictionary juga dapat menggunakan construktor dict() untuk membuat sebuah dictionary yang baru (program 47).

Python juga memiliki method built-in yang dapat digunakan untuk mengolah dictionary. Method-method lain yang selain dijelaskan sebelumnya dapat dilihat pada tabel 3.4.

Program #45

myfamily = {  
  "anak1" : {  
    "nama" : "Langgeng",  
    "tahun" : 2004  
  },  
  "anak2" : {  
    "nama" : "Siska",  
    "tahun" : 2007  
  },  
  "anak3" : {  
    "nama" : "Novi",  
    "tahun" : 2011  
  }  
}

Program #46

anak1 = {  
  "nama" : "Langgeng",  
  "tahun": 2004  
}  
anak2 = {  
  "nama" : "Siska",  
  "tahun" : 2007  
}  
anak3 = {  
  "nama" : "Novi",  
  "tahun" : 2011  
}  
  
myfamily = {  
  "anak1" : anak1,  
  "anak2" : anak2,  
  "anak3" : anak3  
}

Program #47

thisdict = dict(brand="Ford", model="Mustang", year=1964)  
# note that keywords are not string literals  
# note the use of equals rather than colon for the assignment  
print(thisdict)

Tabel 4.4 Method built-in Python untuk digunakan pada dictionary

|  |  |
| --- | --- |
| **Method** | **Deskripsi** |
| fromkeys() | Menghasilkan sebuah dictionary dengan kunci dan nilai yang spesifik |
| keys() | Menghasilkan sebuah list yang berisi kunci-kunci dictionary |
| setdefault() | Mengembalikan nilai dari kunci yang spesifik. Jika kunci tidak ada : masukkan kuncinya, dengan nilai yang spesifik. |
| update() | Memperbaharui dictionary dengan pasangan kunci dan nilai yang spesifik |

* 1. **Kegiatan Praktek**

1. Ketiklah script semua program yang ada dalam bagian 3.2 pada Python Shell anda.
2. Kemudian jalankan,
3. Jika ada yang error, perbaiki errornya,
4. Catatlah hasil yang anda dapatkan
   1. **Tugas Mandiri**
   2. Berikan penjelasan secara singkat, padat dan jelas tentang kesimpulan apa yang anda dapatkan dari hasil kegiatan praktikum ini. (penjelasan ini akan dipertanggungjawabkan secara lisan saat asistensi laporan)
   3. Buatlah list, tuple, set dan dictionary, kemudian lakukan pengolahan mulai dari membuat, mengakses, memasukan, mengubah dan menambah serta menghapus item.
   4. Pengerjaan untuk soal nomor 2 pada tugas mandiri ini, untuk setiap mahasiswa tidak boleh sama.

**PRAKTIKUM 4**

**BILANGAN DAN STRING**

* 1. **Tujuan**

1. Mahasiswa dapat memahami penggunaan bilangan dalam Python
2. Mahasiswa dapat memahami penggunaan string dalam Python
   1. **Dasar Teori**
      1. Bilangan (Numbers)

Terdapat tiga (3) jenis bilangan dalam Python :

* Int
* Float
* Complex

Variabel dari bilangan dibuat ketika menetapkan sebuah nilai ke dalam variabel bilangan tersebut. Berikut ini contoh penetapan nilai ke dalam variabel bilangan :

x = 1 #int

y = 2.8 #float

z = 1j #complex

1. int

int, atau integer, adalah semua bilangan, positif, negatif tanpa desimal dengan panjang yang tidak terbatas. Berikut ini adalah contoh bilangan integer :

x = 1

y = 35656222554887711

z = -3255522

1. Float

Float, atau , “floating point number” adalah sebuah bilangan, positif, atau negatif yang berisi satu atau lebih angka desimal. Berikut ini adalah contoh bilangan integer :

x = 1.10  
y = 1.0  
z = -35.59

Float dapat juga berupa bilangan-bilangan dengan simbol “e” yang berarti pangkat 10. Contohnya dapat ditunjukkan sebagai berikut :

x = 35e3  
y = 12E4  
z = -87.7e100

1. Complex

Bilangan kompleks ditulis dengan sebuah simbol “j” yang berarti bagian imajiner. Contohnya :

x = 3+5j  
y = 5j  
z = -5j

Sebuah tipe dapat diubah menjadi tipe yang lain dengan menggunakan metode int(), float(), dan complex(). Program 1 berikut adalah contoh pengubahan dari satu tipe ke tipe yang lain :

Program #1

x = 1 # int  
y = 2.8 # float  
z = 1j # complex  
  
#convert from int to float:  
a = float(x)  
  
#convert from float to int:  
b = int(y)  
  
#convert from int to complex:  
c = complex(x)  
  
print(a)  
print(b)  
print(c)

Namun, sebagai catatan bilangan complex tidak dapat diubah menjadi tipe bilangan yang lain.

Python tidak memiliki fungsi random() untuk membuat suatu bilangan random, tetapi Python memiliki modul built-in yang disebut random, yang dapat digunakan untuk membuat angka-angka random. Penggunaan modul random ini dapat dilakukan dengan mengimport modul random tersebut sebelum digunakan. Contoh :

import random  
  
print(random.randrange(1,10))

Selain fungsi random.randrange ini masih banyak fungsi-fungsi random lain yang dapat dilihat di <https://www.w3schools.com/python/module_random.asp>.

Python memiliki fungsi-fungsi bilangan yang dapat dipanggil secara langsung tanpa harus mengimpor modul tertentu. Fungsi-fungsi ini adalah sebagai berikut :

* Fungsi abs()
* Fungsi bin()
* Fungsi complex()
* Fungsi float()
* Fungsi hex()
* Fungsi int()
* Fungsi oct()
* Fungsi max()
* Fungsi min()
* Fungsi round()
* Fungsi sum()

Selain itu juga ada modul math yang berisi fungsi untuk melakukan perhitungan-perhitungan matematika, seperti pembagian, perpangkatan, pencarian akar kuadrat dari suatu bilangan, trigonometri dan lain-lain. Fungsi-fungsi yang terdapat dalam modul math adalah sebagai berikut :

* Fungsi math.acos()
* Fungsi math.asin()
* Fungsi math.atan()
* Fungsi math.ceil()
* Fungsi math.cos()
* Fungsi math.degrees()
* Fungsi math.e
* Fungsi math.exp()
* Fungsi math.fabs()
* Fungsi math.factorial()
* Fungsi math.floor()
* Fungsi math.fmod()
* Fungsi math.fsum()
* Fungsi math.gcd()
* Fungsi math.hypot()
* Fungsi math.log()
* Fungsi math.log10()
* Fungsi math.modf()
* Fungsi math.pi
* Fungsi math.pow()
* Fungsi math.radians()
* Fungsi math.sin()
* Fungsi math.sqrt()
* Fungsi math.tan()
* Fungsi math.trunc()

Python masih memiliki fungsi dan modul yang lain yang tidak dapat disebutkan secara satu persatu dalam panduan praktikum ini. Modul-modul yang dituliskan di atas adalah modul yang sering atau umum digunakan.

* + 1. String

Penulisan String dalam Python diapit oleh tanda petik satu atau petik dua. Selajutnya, seperti yang dibahas pada bagian praktikum yang kedua, untuk menampilkan string ini dapat digunakan fungsi print(). Contoh dapat dilihat dalam program 2.

Program #2

print("Hello")  
print('Hello')

Penetapan sebuah string ke dalam sebuah variabel dilakukan dengan nama variabel yang diikuti oleh tanda sama dengan dan stringya.

Program #3

a = "Hello"  
print(a)

String dengan beberapa baris dapat dimasukkan ke dalam sebuah variabel dengan menggunakan tiga kali tanda petik dua atau tiga kali tanda petik satu. Contohnya seperti ditunjukkan oleh program 4.

Program #4

a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)

b = '''Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.'''  
print(b)

Seperti halnya bahasa pemrograman populer lainnya, string dalam Python adalah byte-byte array yang dinyatakan dalam karakter unicode. Bagaimanapun, Python tidak memiliki suatu tipe data karakter, suatu karakter tunggal string secara sederhana dinyatakan dengan panjang 1. Kurung siku dapat digunakan untuk mengakses elemen-elemen dari string. Program 5 menunjukkan cara pengaksesan suatu karakter dari string.

Program #5

a = "Halo, Dunia!"  
print(a[1])

Sebagai catatan, untuk mengambil sebuah karakter string dimulai dari dari karakter paling kiri dengan nomor posisi (indeks) dimulai 0. Selain itu juga, pengambilan karakter dapat dilakukan dengan range atau jangkauan tertentu. Misalnya, ingin diambil beberapa karakter. Contohnya seperti ditunjukkan oleh program 6.

Program #5

b = "Selamat Datang!"  
print(b[2:5])

Pengambilan beberapa karakter, seperti contoh di atas mulai dari posisi 2 sampai dengan posisi 5. Tetapi yang perlu diperhatikan bahwa karakter yang pada posisi 5 dalam contoh tidak termasuk. Pengambilan karakter dengan range atau jangkauan tertentu juga bisa menggunakan indeks negatif. Indeks negatif ini menyatakan mulai pengambilan dari akhir karakter (paling kanan). Contoh dalam program 6.

Program #6

b = "Selamat Datang!"  
print(b[-5:-2])

Fungsi len() digunakan untuk mendapatkan panjang dari suatu string. Panjang string ini berarti bahwa banyaknya karakter dari string tersebut. Program 7 merupakan contoh untuk menghitung panjang dari suatu string.

Program #7

a = "Selamat Datang!"  
print(len(a))

Terdapat beberapa metode built-in dari Python untuk string. Diantaranya :

* + - strip()
    - lower()
    - upper()
    - replace()
    - split()

Untuk mengecek jika terdapat frasa atau karakter tertentu dalam string, dapat digunakna kata kunci in atau not in. Sebagai contoh diberikan oleh program 8.

Program #8

Txt1 = "Adanya COVID-19, kita diharapkan tetap di rumah"  
x = "rap" in txt1  
print(x)

Txt2 = "Adanya COVID-19, kita diharapkan tetap di rumah in"  
y = "mah" not in txt2  
print(y)

Dua buah atau lebih string dapat digabungkan, dengan menggunakan operator +. Contohnya ditunjukkan oleh program 9. Dalam program 9, penggabungan string a dan b dikombinasikan dengan penambahan karakter spasi.

Program #9

a = "Teknik"  
b = "Elektro"  
c = a + " " + b  
print(c)

Sebagai catatan, string tidak dapat digabungkan dengan angka. Namun, untuk kondisi tertentu diperlukan penggabungan angka dan string dapat dilakukan dengan metode format(). Contoh 10 dan 11 menunjukkan penggabungan string dan angka/bilangan.

Program #10

age = 20  
txt = "Nama saya Langgeng, dan umur saya {}"  
print(txt.format(age))

Program #11

quantity = 3  
itemno = 567  
price = 49.95  
myorder = "Saya ingin {} buah barang {} untuk {} dollar."  
print(myorder.format(quantity, itemno, price))

Untuk memasukkan karakter secara langsung ke dalam string tidak diperbolehkan oleh Python. Cara yang bisa dilakukan untuk memasukkan karakter ke dalam string adalah menggunakan escape character (\). Karakter yang ingin dimasukkan dituliskan setelah tanda backslash ini. Contohnya seperti yang ditunjukkan oleh program 12.

Program #11

txt = "Kami adalah mahasiswa Teknik \"Elektro\" FT-UNG."

Tabel 3.1 menyajikan beberapa metode built-in yang digunakan untuk string pada Python

Tabel 4.1 Metode built-in untuk string

|  |  |
| --- | --- |
| **Method** | **Deskripsi** |
| [capitalize()](https://www.w3schools.com/python/ref_string_capitalize.asp) | Mengubah karakter pertama menjadi huruf besar |
| [casefold()](https://www.w3schools.com/python/ref_string_casefold.asp) | Mengubah string menjadi huruf kecil |
| [center()](https://www.w3schools.com/python/ref_string_center.asp) | Mengembalikan string rata tengah |
| [count()](https://www.w3schools.com/python/ref_string_count.asp) | Mengembalikan angka, berapa kali nilai tertentu yang terdapat dalam sebuah string |
| [encode()](https://www.w3schools.com/python/ref_string_encode.asp) | Menghasilkan sebuah versi encode dari string |
| [endswith()](https://www.w3schools.com/python/ref_string_endswith.asp) | Menghasilkan true jika akhir dari string dengan nilai tertentu |
| [expandtabs()](https://www.w3schools.com/python/ref_string_expandtabs.asp) | Menset panjang tab dari string |
| [find()](https://www.w3schools.com/python/ref_string_find.asp) | Mencari string dengan nilai tertentu dan mengembalikan posisi dimana yang dicari berada |
| [format()](https://www.w3schools.com/python/ref_string_format.asp) | Mem-format nilai-nilai tertentu dalam sebuah string |
| format\_map() | Mem-format nilai-nilai tertentu dalam sebuah string |
| [index()](https://www.w3schools.com/python/ref_string_index.asp) | Mencari string untuk nilai tertentu dan mengembalikan posisi dari string yang dicari berada |
| [isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp) | Mengembalikan True jika semua karakter dalam string adalah alphanumeric |
| [isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp) | Mengembalikan True jika semua karakter dalam string adalah alphabet |
| [isdecimal()](https://www.w3schools.com/python/ref_string_isdecimal.asp) | Mengembalikan True jika semua karakter dalam string adalah decimal |
| [isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp) | Mengembalikan True jika semua karakter dalam string adalah digit |
| [isidentifier()](https://www.w3schools.com/python/ref_string_isidentifier.asp) | Mengembalikan True jika string adalah sebuah identifier |
| [islower()](https://www.w3schools.com/python/ref_string_islower.asp) | Mengembalikan True jika semua karakter dalam string adalah huruf kecil |
| [isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp) | Mengembalikan True jika semua karakter dalam string adalah bilangan |
| [isprintable()](https://www.w3schools.com/python/ref_string_isprintable.asp) | Mengembalikan True jika semua karakter dalam string adalah dapat dicetak |
| [isspace()](https://www.w3schools.com/python/ref_string_isspace.asp) | Mengembalikan True jika semua karakter dalam string adalah spasi |
| [istitle()](https://www.w3schools.com/python/ref_string_istitle.asp) | Mengembalikan True jika string mengikuti aturan dari sebuah judul |
| [isupper()](https://www.w3schools.com/python/ref_string_isupper.asp) | Mengembalikan True jika semua karakter dalam string adalah huruf kapital |
| [join()](https://www.w3schools.com/python/ref_string_join.asp) | Menggabungkan elemen-elemen dari sebuah iterable pada akhir dari string |
| [ljust()](https://www.w3schools.com/python/ref_string_ljust.asp) | Mengembalikan versi rata kiri dari string |
| [lower()](https://www.w3schools.com/python/ref_string_lower.asp) | Mengubah sebuah string ke dalam huruf kecil |
| [lstrip()](https://www.w3schools.com/python/ref_string_lstrip.asp) | Mengembalikan versi trim kiri dari string |
| maketrans() | Mengembalikan sebuah tabel translasi yang digunakan dalam translasi |
| [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Mengembalikan sebuah tuple dimana string dibagi menjadi tiga bagian |
| [replace()](https://www.w3schools.com/python/ref_string_replace.asp) | Mengembalikan string dimana nilai tertentu diganti dengan nilai tertentu |
| [rfind()](https://www.w3schools.com/python/ref_string_rfind.asp) | Mencari string untuk nilai tertentu dan mengembalikan di posisi terakhir nilai tersebut ditemukan |
| [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) | Mencari string untuk nilai tertentu dan mengembalikan posisi terakhir dimana nilai tersebut ditemukan |
| [rjust()](https://www.w3schools.com/python/ref_string_rjust.asp) | Mengembalikan versi rata kanan dari string |
| [rpartition()](https://www.w3schools.com/python/ref_string_rpartition.asp) | Mengembalikan sebuah tuple dimana string dibagi menjadi tiga bagian |
| [rsplit()](https://www.w3schools.com/python/ref_string_rsplit.asp) | Membagi string berdasarkan pemisah tertentu, dan mengembalikan sebuah list. |
| [rstrip()](https://www.w3schools.com/python/ref_string_rstrip.asp) | Mengembalikan versi trim kanan dari string |
| [split()](https://www.w3schools.com/python/ref_string_split.asp) | Membagi string berdasarkan pemisah tertentu, dan mengembalikan list |
| [splitlines()](https://www.w3schools.com/python/ref_string_splitlines.asp) | Membagi string pada baris baru dan mengembalikan list |
| [startswith()](https://www.w3schools.com/python/ref_string_startswith.asp) | Mengembalikan True jika string mulai dengan nilai tertentu. |
| [strip()](https://www.w3schools.com/python/ref_string_strip.asp) | Mengembalikan versi trimed dari string |
| [swapcase()](https://www.w3schools.com/python/ref_string_swapcase.asp) | Kasus penukaran, huruf kecil menjadi kapital dan sebaliknya. |
| [title()](https://www.w3schools.com/python/ref_string_title.asp) | Mengubah karakter pertama dari setiap kata menjadi huruf kapital |
| translate() | Mengembalikan sebuah string yang tertranslate |
| [upper()](https://www.w3schools.com/python/ref_string_upper.asp) | Mengubah string menjadi huruf kapital |
| [zfill()](https://www.w3schools.com/python/ref_string_zfill.asp) | Mengisi string dengan suatu bilangan tertentu dengan nilai 0 pada awal |

* 1. **Kegiatan Praktek**

1. Ketiklah script semua program yang ada dalam bagian 3.2 pada Python Shell anda.
2. Kemudian jalankan,
3. Jika ada yang error, perbaiki errornya,
4. Catatlah hasil yang anda dapatkan
   1. **Tugas Mandiri**
   2. Buatlah flowchart dan program untuk menghitung panjang sisi miring suatu segitiga siku-siku. Sisi miring suatu segitiga dapat dihitung dengan menggunakan rumus phytagoras yaitu :
   3. Buatlah flowchart dan program untuk mengetahui berapa banyak kemunculan suatu karakter/huruf vokal dan konsonan dalam sebuah string. Misalnya jika ingin diketahui berapa banyak kemunculan karakter vokal dan konsonan dalam string ‘tambalakada’ maka program akan menampilkan angka 5 untuk karakter vokal dan angka 6 untuk karakter konsonan.
   4. Buatlah flowchart dan program untuk mengubah setiap huruf tertentu yang ada di suatu string menjadi huruf lain. Jika misalnya kita ingin mengubah semua huruf ‘e’ menjadi ‘o’ pada kalimat ‘Mengapa terjadi bencana ?’, maka kalimat tersebut akan menjadi ‘Mongapa torjadi boncana ?’.
   5. Buatlah flowchart dan program untuk menampilkan string secara menurun, sehingga membentuk model seperti iklan dari lampu neon. Sebagai contoh bila disikan HULONDALANGI, maka hasilnya akan tampak di layar monitor adalah :

H

U

L

O

N

D

A

L

A

N

G

I

* 1. Pembuatan kata sandi sederhana dapat dilakukan dengan mengganti suatu huruf vokal menjadi huruf lain yang berada di posisi setelahnya dalam daftar abjad A-Z. sebagai contoh, sebuah string yang berisi kalimat “Aku Cinta Padamu” bila disandikan akan berubah menjadi maka string tersebut akan berubah menjadi “Bkv Cjntb Pbdbmv”. Jadi huruf ‘A’ pada kata “Aku” berubah menjadi ‘B’, huruf ‘u’ menjadi ‘v’ dan seterusnya. Buatlah flowchart dan program untuk melakukan penyandian ini.

**PRAKTIKUM 5**

**KONDISI PADA PYTHON**

* 1. **Tujuan**

1. Mahasiswa dapat memahami penggunaan if untuk satu kondisi
2. Mahasiswa dapat memahami penggunaan if…else untuk dua kondisi
3. Mahasiswa dapat memahami penggunaan if…elif…else untuk tiga kondisi atau lebih
   1. **Dasar Teori**

Python mendukung penggunaan kondisi logis dari matematika, yaitu penggunaan :

* Sama dengan : a == b
* Tidak sama dengan : a != b
* Lebih kecil : a < b
* Lebih besar : a > b
* Lebih besar atau sama dengan : a >= b

Kondisi-kondisi di atas dapat digunakan dalam berbagai keperluan, umumnya digunakan pada pernyataan if dan perulangan. Beberapa bentuk kondisi (3 bentuk) akan dijelaskan secara singkat pada panduan praktikum ini. Hal yang penting yang perlu dibahas sebelum masuk ke persoalan kondisi, yaitu indentasi. Python bergantung Indentation atau indentasi. Indentasi adalah pemberian spasi pada awal sebuah baris yang mendefinisikan scope atau cakupan dari kode. Bahasa pemrograman yang lain biasanya menggunakan tanda kurung kurawal untuk menyatakan hal ini. Program 1 adalah contoh penggunaan indentasi ini.

Program #1

a = 33  
b = 200  
if b > a:  
print("b is greater than a") # anda akan mendapatkan error jika program ini dijalankan

Program 1 ketika dijalankan akan memberikan pesan error. Hal ini terjadi karena pada baris dengan fungsi print() tidak diberi spasi atau inden. Program dapat diperbaiki, dengan memberikan spasi sebelum fungsi print(). Pemberian banyaknya spasi tergantung dari keinginan programer. Yang terpenting di sini adalah pemberian spasi setelah baris dengan pernyataan if yang berarti bahwa baris di bawahnya adalah merupakan cakupan atau scope dari pernyataan if tersebut.

* + 1. Kondisi if

Bentuk umum dari perintah if untuk satu kondisi adalah sebagai berikut :

if kondisi:  
 aksi

Pada bentuk di atas, aksi hanya akan dilaksanakan saat kondisi bernilai True (benar). Jika bernilai False (salah), maka aksi akan diabaikan, dan program akan melanjutkan proses evakuasi ke baris yang ditulis setelah blok pemilihan.

Jika aksi hanya terdiri dari satu perintah, maka aksi dapat ditulis di samping (setelah tanda titik dua), seperti berikut :

if kondisi: aksi

Untuk aksi yang berjumlah lebih dari satu perintah, maka perlu ditulis seperti berikut ini :

if kondisi:  
 aksi1

aksi2

aksi3

aksi4

Program 2 menunjukkan penggunaan perintah if dengan satu kondisi.

Program #2a

a = 5

if a > 0:print('a adalah bilangan positif')

Atau juga bisa ditulis :

Program #2b

a = 5

if a > 0:

print('a adalah bilangan positif')

* + 1. Kondisi if…else

Bentuk umum dari perintah if dua kondisi adalah sebagai berikut :

if kondisi:  
 aksi1

else

aksi2

Blok pemilihan di atas menyertakan penanganan terhadap dua kondisi, kondisi benar dan kondisi salah. Pada bentuk ini, aksi1 hanya dilakukan ketika kondisi bernilai True. Pada saat program mengeksekusi aksi1, aksi2 akan diabaikan. Jika kondisi bernilai False, maka yang dilakukan adalah aksi2, dan aksi1 diabaikan. Program 3 menunjukkan penggunaan perintah if dengan dua kondisi.

Program #3

a = -5

if a > 0:

print('a adalah bilangan positif')

else

print('a bukan bilangan positif')

Dalam kasus-kasus tertentu ketika diinginkan untuk masukkan nilai ke dalam suatu variabel tertentu, perintah if dapat ditulis seperti bentuk berikut :

variabel = nilai1 if kondisi else nilai2

Program 4 berikut memperlihatkan penggunaan bentuk if ini :

Program #4a

a = 100

b = 999

maksimum = a if a > b else b

Program 4a di atas merupakan penyederhanaan dari program 4b berikut :

Program #4b

a = 100

b = 999

if a > b :

maksimum = a

else

maksimum = b

* + 1. Kondisi if…elif…else

Bentuk umum dari perintah if untuk tiga kondisi adalah sebagai berikut :

if kondisi1:  
 aksi1

elif kondisi2:

aksi2

else

aksi3

Apabila kondisi lebih dari tiga, maka dapat digunakan bentuk umum berikut :

if kondisi1:  
 aksi1

elif kondisi2:

aksi2

elif kondisi3:

aksi3

…

elif kondisiN:

aksiN

else

aksiAlternatif

Pada bentuk di atas, mula-mula program akan memeriksa nilai dari kondisi1, jika kondisi1 bernilai True, maka program akan melaksanakan aksi1. Jika False, maka program akan memeriksa kondisi2. Jika kondisi2 bernilai True, maka aksi2 yang akan dilaksanakan. Jika ternyata masih bernilai False, maka program akan memeriksa kondisi3. Demikian seterusnya, sampai ditemukan kondisi yang bernilai True. Seandainya semua kondisi yang didefinisikan bernilai False, maka program akan melaksanakan aksi alternatif yang terdapat di bagian else.

Program 5 berikut akan mencontohkan penggunaan if dengan tiga kondisi

Program #5

a = 0

if a > 0 :

print('a adalah bilangan positif')

elif a == 0 :

print('a bernilai nol')

else

print('a adalah bilangan negatif')

Untuk pemilihan yang melibatkan banyak kondisi (lebih dari tiga), penggunaannya dituliskan seperti dalam program 6 :

Program #6

no = 6

if no == 1 :

s = 'Januari'

elif no == 2 :

s = 'Februari'

elif no == 3 :

s = 'Maret'

elif no == 4 :

s = 'April'

elif no == 5 :

s = 'Mei'

elif no == 6 :

s = 'Juni'

elif no == 7 :

s = 'Juli'

elif no == 8 :

s = 'Agustus'

elif no == 9 :

s = 'September'

elif no == 10 :

s = 'Oktober'

elif no == 2 :

s = 'November'

elif no == 2 :

s = 'Desember'

else

s = 'Tidak ada bulan yang dipilih'

print(s)

* 1. **Kegiatan Praktek**
  2. Periksalah apakah program-program berikut ini terdapat kesalahan. Jika ada perbaikilah.

Program PRK-1

a = input('Masukkan a: '))

b = str(input('Masukkan b: '))

if b == 0

print('\nKESALAHAN: Nilai b tidak boleh nol')

c = a / b

print('\n%.3f / %.3f = %.3f' % (a, b, c))

Program PRK-2

import sys

uts = float(input('Masukkan Nilai UTS : '))

if (0,0 >= Uts >= 100)

print('nilai UTS harus berada dalam rentang 0..100')

sys.exit(1)

Uas = float(input('Masukkan Nilai UAS : '))

if not(0.0 < uas < 100)

print('nilai UAS harus berada dalam rentang 0..100')

sys.exit(1)

nilai = (0,4\*uTs)+(0.6\*UAS)

if Nilai >= 80 :

indeks = 'A"

elif nilai > 70.0 :

indeks == 'B"

elif nilai >= 60,0 :

indekS = 'C"

elif nilai >= 40.0 :

indeks == "D"

else

indeks == 'E"

print('\nNilai Akhir\t= %f' nilai')

print('\nNilai Akhir\t= %f' nilai')

* 1. Kedua program PRK di atas program untuk mencari apa ?
  2. Catatlah hasil yang anda dapatkan.
  3. **Tugas Mandiri**

1. Buatlah flowchart dan program untuk menentukan semester yang telah dilalui oleh seorang mahasiswa jika yang diinputkan adalah tahun masuk pertama kali kuliah dan tahun sementara berjalan
2. Buatlah flowchart dan program untuk menentukan apakah angka yang diinputkan lebih kecil dari atau lebih besar dari nol.
3. Tulis program untuk menentukan biaya parkir yang dihitung berdasarkan lama parkir. Lama parkir dihitung dari selisih jam masuk dan jam keluar diinput. Biaya parkir 2 jam pertama 2000, perjam berikutnya 500.

Contoh Masukan dan keluaran:

|  |  |  |  |
| --- | --- | --- | --- |
| Jam masuk | Jam keluar | Lama | keluaran/tampilan |
| 10 | 11 | 1 | Biaya = 2000 |
| 10 | 2 | 4 | Biaya = 3000 |

**PRAKTIKUM 6**

**PENGULANGAN (LOOP)**

* 1. **Tujuan**

1. Mahasiswa dapat memahami penggunaan while
2. Mahasiswa dapat memahami penggunaan for
   1. **Dasar Teori**

Blok pengulangan adalah konstruksi yang digunakan untuk melakukan eksekusi perintah (atau kumpulan perintah) yang sama secara berulang. Untuk keperluan pengulangan Python hanya menyediakan dua perintah yaitu while dan for.

* + 1. Perintah while

Bentuk umum penggunaan perintah while adalah sebagai berikut :

while kondisi:  
 aksi

pada konstruksi di atas, aksi akan terus dilakukan oleh program selama kondisi bernilai True. Dengan demikian, ketika menggunakan while, kita memerlukan suatu cara untuk menjadikan kondisi bernilai False. Hal ini dilakukan agar suatu proses pengulangan tersebut dapat berhenti. Jika kondisi tidak pernah bernilai False, maka pengulangan akan dijalankan secara terus menerus.

Program 1 menunjukkan penggunaan while dalam program.

Program #1

i = 0

while i < 9 :

print('Python 3.%d' %i)

i += 1

Variabel i digunakan sebagai indeks atau pencatat langkah pengulangan. Pada contoh di atas, pengulangan dilakukan sebanyak 9 kali, mulai dari indeks ke-0 sampai dengan indeks ke-8. Setiap kali program mengeksekusi perintah yang ada di dalam badan pengulangan, nilai i akan ditambah 1.

Dalam program 1, pengulangan dilakukan secara menaik, mulai dari i=0 sampai i=8, perintah while juga dapat digunakan untuk melakukan pengulangan seara menurun, misalnya mulai dari i=8 sampai i=0. Program 2 memperlihatkan pengulangan secara menurun.

Program #2

i = 8

while i > -1 :

print('Python 3.%d' %i)

i -= 1

Kali ini setiap kali program mengeksekusi perintah yang ada di dalam badan pengulangan, nilai i akan dikurangi 1.

Dalam pengulangan ada pernyataan break yang digunakan untuk menghentikan pengulangan meskipun kondisi while bernilai True.

Program #3

i = 1

while i < 6 :

print(i)

if i == 3:

break

i += 1

Selain pernyataan break, ada juga pernyataan continue, yang digunakan untuk menghentikan iterasi dan melanjutkan dengan yang berikutnya.

Program #4

i = 0

while i < 6 :

i += 1

if i == 3:

continue

print(i)

Perbedaan hasil program 3 dan 4 yaitu, ketika program 3 dijalankan, program akan berhenti mencetak i, ketika i=3, sedang untuk pada program 4, ketika i=3 program tidak akan menampilkan isi i=3 tapi langsung menampilkan i=4 dan program akan berlanjut sampai dengan i=6.

Perintah while dapat dikombinasikan dengan pernyataan else. Dengan adanya pernyataan else, program akan menjalankan blok kode dalam else ketika kondisi tidak lagi bernilai True.

Program #4

i = 1

while i < 6 :

print(i)

i += 1

else

print("i tidak lagi lebih dari 6")

* + 1. Perintah for

Fungsi range() digunakan untuk menghasilkan kelompok nilai (koleksi) yang berada dalam rentang tertentu. Objek yang dihasilkan oleh fungsi ini akan bertipe range. Bentuk umum penggunaan range() dapat dilihat di bawah ini :

Untuk step bernilai positif :

range(indeksAwal, indeksAkhir+1, step)

Untuk step bernilai negatif :

range(indeksAwal, indeksAkhir-1, step)

Fungsi di atas akan membuat sekelompok nilai mulai dari indeksAwal sampai dengan indeksAkhir. Parameter step adalah selisih nilai dari masing-masing data. Jika parameter step tidak disertakan maka dianggap bernilai 1. Sebagai contoh untuk menciptakan sekelompok nilai dalam rentang 0 sampai 9, dapat ditulis kode seperti dalam program 5.

Program #5

range1 = range(0,10)

list(range1)

Baris pertama dalam program 5, program akan menciptakan nilai yang dimulai dari 0 sampai dengan 9 dengan selisih 1. Baris kedua variabel range1 tadi diubah menjadi tipe data list untuk bisa diakses. Jika pada program 5 tersebut disertakan parameter step, misalnya 2 maka data yang akan dihasilkan pada variabel range1 adalah data bilangan yang dimulai dari 0, 2, 4, 6, dan 8. Selain itu juga, dalam menciptakan data dengan fungsi range(), bisa saja indeksAwal dan parameter step tidak disertakan, dengan bentuk seperti ini maka otomatis program akan menetapkan nilai indeksAwal bernilai 0 dan nilai parameter step adalah 1.

Fungsi range() banyak digunakan dalam melakukan pengulangan menggunakan perintah for. Bentuk umum dari perintah for dalam penggunaannya pada range() sebagai berikut :

for indeks in range(indeksAwal, indeksAkhir, step):

aksi

program 6 adalah contoh penggunaan perintah for dengan fungsi range().

Program #6

for i in range1 = range(9):

print("Python 3.%d" %i)

Selain untuk objek bertipe range, perintah for juga dapat digunakan untuk menelusuri elemen-elemen dari objek koleksi lain, seperti list, tuple, set dan dict; juga termasuk str (string). Program 7 menunjukkan contoh penggunaan for untuk tipe data list. Selanjutnya, untuk tipe data str contohnya dalam program 8, yaitu mengambil setiap huruf atau karakter yang ada dalam sebuah kata.

Program #7

fruits = ["pepaya", "mangga", "pisang"]  
for x in fruits:  
  print(x)

Program #8

for x in "tambalakada":  
  print(x)

Sama halnya dengan perintah while, pernyataan break, continue juga bisa digunakan. Demikian pula untuk kata kunci else.

Dalam pengulangan dikenal istilah nested loop (pengulangan bersarang). Nested loop artinya bahwa dalam pengulangan terdapat pengulangan. Pengulagan bagian dalam akan dieksekusi satu kali setiap kali pengulangan bagian luar. Program 9 menunjukkan bentuk sederhana dari nested loop ini.

Program #9

buah = ["pepaya", "mangga", "pisang"]

sifat = ["merah", "besar", "enak"]  
for x in buah:  
  for y in sifat:  
    print(x, y)

Hasil dari program 9 ini ketika dijalankan yaitu akan tercetak ke layar :

pepaya merah

pepaya besar

pepaya enak

mangga merah

mangga besar

mangga enak

pisang merah

pisang besar

pisang enak

Pengulangan for tidak boleh kosong, tetapi jika dimungkinkan ingin membuat pengulangan for tanpa ada konten, maka dapat digunakan pernyataan pass untuk menghindari terjadinya error. Program 10 menunjukkan cara penulisannya.

Program #10

for x in [0, 1, 2]:  
  pass

* 1. **Kegiatan Praktek**

1. Buatlah program untuk mencetak nama dan nim anda dengan sejumlah angka yang ditentukan oleh user melalui keyboard.
2. Buatlah program untuk mencetak ke layar monitor secara berurutan bilangan ganjil dari 1 sampai 20.
3. Berikan komentar anda tentang hasil kegiatan praktek yang anda telah lakukan
   1. **Tugas Mandiri**
4. Buatlah flowchart dan program untuk menampilkan deret bilangan genap dengan tanda minus dan plus secara berselang-seling sebanyak N suku. Sebagai contoh bila N berisi angka 5, maka deret yang akan ditampilkan di layar monitor adalah: -2 +4 -6 +8 -10.
5. Buatlah flowchart dan program untuk menampilkan digit-digit sebuah bilangan secara menurun mulai dari digit yang paling depan ke digit yang paling belakang. Misalkan bilangan tersebut adalah 1234, maka hasil yang akan terlihat di layar monitor adalah:

1

2

3

4

1. Buatlah flowchart dan program untuk mencari bilangan terbesar dari sekumpulan bilangan yang diinputkan. Penginputan berakhir bila user memasukkan nol. Meskipun demikian, nilai nol tersebut tidak dianggap sebagai bilangan yang diinput (jadi angka nol tersebut hanya dipakai sebagai penanda bahwa proses penginputan sudah berakhir).

**PRAKTIKUM 7**

**FUNGSI**

* 1. **Tujuan**

1. Mahasiswa dapat memahami penggunaan fungsi dalam Pyhton
2. Mahasiswa dapat mengetahui Python lambda dan cara penggunaannya
3. Mahsiswa dapat memahami penggunaan fungsi di dalam fungsi
   1. **Dasar Teori**
      1. Fungsi Python

Dalam pembuatan program, kode dapat dibagi menjadi bagian-bagian lain yang lebih kecil dan memiliki tugas yang spesifik. Bagian-bagian program ini sering disebut fungsi. Fungsi hanya perlu didefinisikan satu kali, tapi dapat dipanggil berkali-kali (sesuai kebutuhan). Python sudah menyediakan banyak fungsi built-in (sebelumnya pada modul-modul sebelumnya diartikan sebagai fungsi bawaan). Fungsi built-in ini adalah fungsi yang sudah disediakan oleh Python dan sudah siap digunakan tanpa kita perlu mendefinisikannya lagi. Misalnya, print(), len(), max() dan sebagainya. Namun, untuk memenuhi kebutuhan program yang dikembangkan, kadangkala diperlukan untuk mendefinisikan sendiri fungsi khusus untuk tugas yang spesifik, yang disebut user-defined function. Dalam python, fungsi dinyatakan sebagai objek dari kelas function, dengan demikian maka fungsi memiliki atribut dan method.

Suatu fungsi adalah sebuah blok kode yang hanya dijalankan ketika fungsi tersebut dipanggil. Data dapat dikirim ke dalam fungsi yang disebut dengan parameter. Sebuah fungsi dapat menghasilkan atau mengembalikan data sebagai hasil atau nilai balik dari fungsi tersebut. Fungsi dibedakan menjadi dua :

* + - Fungsi dengan nilai balik
    - Fungsi tanpa nilai balik

Fungsi dengan nilai balik adalah fungsi yang ketika dipanggil dapat mengembalikan suatu nilai, sedangkan fungsi tanpa nilai balik adalah fungsi yang tidak menghasilkan nilai. Nilai yang dikembailkan atau dihasilkan oleh fungsi dapat bertipe apa saja, misalnya bilangan, teks, atau objek koleksi. Bahkan, dalam Python, suatu fungsi dapat mengembalikan nilai yang berupa fungsi lain (objek dari kelas function).

Fungsi Python didefinisikan dengan menggunakan kata kunci def. Bentuk umum dari fungsi adalah sebagai berikut :

def namaFungsi(daftar-parameter):

‘dokumentasi fungsi’

aksi1

aksi2

return[nilai]

Kata kunci return hanya wajib digunakan untuk fungsi yang menghasilkan nilai. Untuk fungsi yang tidak menghasilkan nilai, kata kunci return bersifat opsional; bisa disertakan bisa juga tidak. Kalaupun disertakan, return tidak diikuti dengan nilai tertentu. Dokumentasi fungsi (bersifat string) juga bersifat optional.

Selanjutnnya, sebagai contoh sederhana dapat dilihat pada program 1, cara pemanggilannya dituliskan dalam program 2.

Program #1

def my\_function():  
  print("Tulisan ini berasal dari fungsi")

Program #2

def my\_function ():  
  print("Tulisan ini berasal dari fungsi")

my\_function()

Contoh lain, dari fungsi yang menghasilkan nilai dapat dilihat pada program 3.

Program #3

def tambah(a, b):

‘fungsi untuk menjumlahkan nilai a dan b’

c = a + b  
  return c

my\_function()

Ketika mengkonstruksi fungsi terdapat hal penting yang perlu dipahami, yaitu variabel lokal dan variabel global. Variabel lokal adalah variabel yang didefinisikan di dalam fungsi dan hanya akan dikenal dalam lingkup fungsi bersangkutan. Perhatikan program 3, variabel c disebut dengan variabel lokal. Variabel a dan b yang diperankan sebagai parameter fungsi juga bersifat lokal. Artinya, a dan b hanya dikenal oleh fungsi tambah(). Berbeda dengan variabel lokal, variabel global akan dikenal oleh semua fungsi yang didefinisikan. Sebagai contoh pada program 4. Dalam program 4, variabel global didefinisikan bernilai 0, selanjutnya fg2() mendefinisikan variabel lokal dengan nama g. hal ini menyebabkan variabel global tidak dapat diakses dari fungsi fg2(). Fungsi fg3() mendefinisikan variabel global dengan menuliskan kata kunci global, dan mengubah nilai variabel global yang tadinya 0 menjadi 2.

Program #4

g = 0 #variabel global

def fg1():

print(g) #menampilkan nilai 0

def fg2():

g = 1 #variabel lokal

#menyebabkan variabel global tidak dapat diakses dari fungsi ini

print(g) #menampilkan nilai 1

def fg3():

global g

g = 2 #mengubah nilai variabel global

#menyebabkan variabel global tidak dapat diakses dari fungsi ini

print(g) #menampilkan nilai 2

Selanjutnya, dalam fungsi ada yang disebut dengan parameter opsional, yakni parameter yang nilai awalnya sudah didefinisikan pada saat fungsi dibuat. Nilai untuk parameter opsional dapat diubah pada saat fungsi dipanggil. Contohnya dapat dilihat dalam program 5.

Program #5

def daerah(**daerah\_asal = "Kota Gorontalo"**):  
  print("Saya dari " + daerah\_asal)  
  
my\_function("Toili")  
my\_function("Luwuk")  
my\_function()  
my\_function("Marisa")

my\_function("boroko")

Ketika melakukan pemanggilan fungsi, normalnya perlu disertakan nilai untuk parameter aktual (disebut argumen) sesuai dengan urutan parameter formal yang didefinisikan pada saat pembuatan fungsi. Sebagai contoh perhatikan program 6.

Program #6a

def tampil(a, b, c): #a, b, dan disebut sebagai parameter formal  
  print("a = %d" %a)

print("b = %d" %b)

print("c = %d" %c)  
  
tampil(50, 60, 70) #memanggil fungsi

dalam program 6a, 50, 60 dan 70 adalah nilai argumen yang dikirim ke dalam fungsi tampil(). Dalam contoh ini, nilai 50 akan dimasukkan ke dalam variabel a, 60 ke dalam variabel b, dan 70 dimasukkan ke varibael c. Hal ini disebabkan karena urutan argumen akan disesuaikan dengan urutan parameter yang didefinisikan di dalam fungsi tampil().

Python mendukung pemanggilan fungsi menggunakan keyword argument. Dengan keyword argument, urutan untuk nilai a, b dan c dapat diubah (tidak harus sesuai dengan urutan yang sebenarnya). Cara yang dilakukan yaitu dengan menyebutkan nama parameter dan nilainya, sehingga pemanggilan fungsi dalam program 6a dapat dimodifikasi seperti yang ditunjukkan oleh program 6b.

Program #6b

def tampil(a, b, c): #a, b, dan disebut sebagai parameter formal  
  print("a = %d" %a)

print("b = %d" %b)

print("c = %d" %c)  
  
tampil(c=50, b=60, a=70) #memanggil fungsi

Pada fungsi-fungsi yang sudah didefinisikan oleh Python, banyak dijumpai fungsi yang memiliki parameter \*args dan \*\*kwargs. Dalam Python, parameter seperti ini disebut variable-length-arguments. Kedua parameter ini mengizinkan suatu fungsi untuk dapat dipanggil menggunakan jumlah argumen yang berbeda.

Parameter \*args digunakan pada parameter formal untuk mengantisipasi jika berapa banyak argumen (parameter aktual) yang dikirim ke fungsi. Penulisannya dengan menambahkan sebuah karakter \* sebelum parameter formal pada saat fungsi didefinisikan. Dengan cara ini fungsi dapat menerima argumen dengan tipe tuple. Contohnya program 7 menunjukkan cara penggunaan \*args ini.

Program #7

def mahasiswa(\*mhs):  
  print("Mahsiswa tertua adalah " + kids[2])  
  
my\_function("Langgeng", "Siska", "Saripudin")

Fungsi yang mendefinisikan parameter \*args ini masih dapat memiliki parameter normal. Jika parameter \*args ditempatkan sebelum parameter normal, maka pada saat pemanggilan fungsi, pada parameter formal, parameter normal ini harus menyebutkan namanya. Namun, jika pada saat pendefinisian fungsinya, parameter formal ditempatkan sebelum (sebelah kiri) parameter \*args, maka saat pemanggilan fungsinya tidak harus menyebutkan namanya.

Parameter \*\*kwargs digunakan untuk membuat fungsi yang dapat dipanggil menggunakan jumlah argumen berbeda dan disertai dengan nama. Argumen yang dikirim ke fungsi harus berupa pasangan nama dan nilai, dalam format : nama=nilai. Di dalam fungsi, daftar argumen tersebut selanjutnya akan dijadikan sebagai objek dictionary. Contohnya seperti dalam program 8.

Program #8

def mah(\*\*mhs):  
  print("Nama belangkangnya " + mhs["lname"])  
  
mah(fname = "Langgeng", lname = "Widodo")

* + 1. Lambda Python

Fungsi nilai balik yang melakukan operasi sederhana dapat dibuat menggunakan operator lambda, dengan menggunakan bentuk umum seperti berikut :

lambda arguments : expression

Perhatikan contoh fungsi pada program 9 berikut :

Program #9a

def maksimum(a, b):  
 if a > b

return a

else

   return b

maksimum(20, 75)

Fungsi maksimum di atas dapat ditulis lebih sederhana menggunakan operator lambda, seperti ditunjukkan oleh program 9b.

Program #9b

maksimum = lambda a, b: a if (a > b) else b

maksimum(20, 75)

Python mengizinkan untuk mendefinisikan fungsi di dalam fungsi lain. Fungsi seperti ini akan bersifat lokal sehingga hanya dikenal oleh fungsi pemilik. Konsep ini dicontohkan dalam program 10.

Program #10

def luasLingkaran(r): #fungsi bagian luar  
 def kuadrat(x): #fungsi bagian dalam

return x \* x

import math

return math.pi\*kuadrat(r)

luasLingkaran(7)

Dalam Python dikenal kata kunci nonlocal yang memiliki fungsi mirip dengan kata kunci global. Perbedaannya hanya terletak pada ruang lingkupnya saja. Kata kunci nonlocal digunakan ketika mendefinisikan fungsi di dalam fungsi, tepatnya ketika fungsi bagian dalam ingin mengakses variabel milik fungsi bagian luar. Contohnya pada program 11 berikut :

Program #11

def fluar():

x = 1 #variabel lokal milik fluar

def fdalam():

nonlocal x #menggunakan x milik fluar()

x = 100 #mengubah nilai x

print("Nilai x dalam fdalam() : %d" %x)

print("Nilai x awal dalam fluar() : %d" %x)

fdalam() #memanggil fungsi fdalam()

print("Nilai x akhir dalam fluar() : %d" %x)

fluar() #memanggil fluar()

* 1. **Kegiatan Praktek**

1. Ketiklah semua program pada bagian 6.2 di atas dalam Pyhton shell anda, kemudian jalankan dan perhatikan hasilnya.
2. Buatlah program untuk menghitung nilai rata-rata dari sekelas mahasiswa untuk dua buah mata kuliah. Catatan: buatlah fungsi untuk perhitungan nilai rata-rata, dengan data yang dibuat menggunakan tuple.
3. Berikan komentar anda tentang hasil kegiatan praktek yang anda telah lakukan.
   1. **Tugas Mandiri**
4. Buatlah flowchart dan program untuk menghitung total nilai dari M bilangan prima yang pertama. Sebagai contoh bila M dimasukkan angka 3, maka diharapkan program akan mengeluarkan total dari 3 bilangan prima yang pertama yaitu 10 (didapat dari 2+3+5). Buatlah fungsi tertentu yang dapat membantu penyelesaian program secara keseluruhan. Data bilangan prima dibuat dengan menggunakan tuple. Argumen pada fungsi dibuat dengan menggunakan gabungan parameter normal dan parameter \*args.
5. Buatlah program untuk mengetahui berapa banyak kemunculan suatu karakter tertentu dalam sebuah string. Misalnya, jika ingin diketahui berapa banyak kemunculan karakter ‘o’ dalam string “tonggolo’o’o”, maka program akan menampilkan angka 5. Gunakan fungsi untuk membantu penyelesaian program tersebut.
6. Buat program menghitung selisih waktu (dalam jam dan menit) berdasarkan jam mulai (jam, menit) dengan jam selesai (jam, menit). Data yang diinput adalah jam dan menit mulai, jam dan menit selesai.

Contoh Masukan dan Keluaran:

Masukan Keluaran

|  |  |  |  |
| --- | --- | --- | --- |
| **Mulai (jam,menit)** | **Selesai (jam,menit)** | **Lama (menit)** | **Lama (jam,menit)** |
| 10:30 | 12:10 | 100 | 1 jam 40 menit |
| 10:30 | 1:40 | 130 | 3 jam 10 menit |

**PRAKTIKUM 8**

**KELAS DAN OBJEK**

* 1. **Tujuan**

1. Mahasiswa dapat memahami pengertian kelas dan objek
2. Mahasiswa dapat mendefinisikan kelas, objek, atribut dan metode
3. Mahsiswa dapat memahami penggunaan kelas, objek, atribut dan metode
4. Mahasiswa dapat memahami kelas turunan
   1. **Dasar Teori**

Python adalah bahasa yang mendukung pemrograman berorientasi objek. Bahkan dalam python, semua elemen program dinyatakan dalam bentuk objek. Untuk mencipatkan objek, diperlukan kelas.

* + 1. Pengertian Kelas dan Objek

Dalam Python, semua nilai yang digunakan oleh program merupakan objek. Masing-masing objek memiliki tipe atau jenis data. Contoh 9 adalah objek yang bertipe bilangan bulat; ‘Pyhton’ adalah objek yang bertipe teks (string); dan [10,20,30] adalah objek yang bertipe list. Intinya, dalam Python kelas adalah tipe atau jenis data dari suatu objek. Gunakan fungsi type() untuk mengetahui tipe dari suatu objek.

Program #1

type(9)

<class 'int'> #hasil

Perintah dan hasil dalam program 1 menunjukkan bahwa 9 adalah objek dari kelas int. artinya, dalam contoh ini, 9 disebut sebagai objek atau instance dan int disebut sebagai kelas.

Jika objek yang diperiksa sudah ditunjuk oleh variabel tertentu, maka variabel tersebut dapat digunakan sebagai argumen pada saat memanggil fungsi type(), seperti yang ditunjukkan oleh program 2.

Program #2

b = 20

type(b)

<class 'int'> #hasil

Kelas didefinisikan sebagai blok program yang dibuat untuk memodelkan atau menggambarkan atribut dan karakteristik/perilaku dari suatu objek tertentu. Dalam pembuatan kelas, atribut dinyatakan dalam bentuk variabel, sedangkan perilaku dinyatakan dalam bentuk fungsi. Perlu diketahui bahwa fungsi yang didefinisikan di dalam suatu kelas di sebut sebagai **metode**. Setiap kelas akan memiliki daftar atribut dan metode yang berbeda-beda, tergantung dari sifat objek yang dimodelkan. Sebagai contoh, dalam permrograman GUI, kelas digunakan untuk memodelkan elemen-elemen seperti form, tombol, label dan sebagainya. Daftar atribut dan metode yang dimiliki oleh kelas Form (misalnya) tidak sama dengan daftar atribut dan metode yang dimiliki oleh kelas Button.

Atribut dan metode yang didefinisikan di dalam suatu kelas selanjutnya dapat diakses melalui objek dari kelas bersangkutan. Sebagai contoh, jika li adalah objek dari kelas list, maka dapat diakses metode append() yang didefinisikan di dalam kelas list melalui li. Contohnya dapat dilihat dalam program 3.

Program #3

li = [20,30]

[10,20] #hasil dari baris pertama

li.append(30)

[10,20,30] #hasil dari baris ketiga

* + 1. Mendefinisikan Kelas

Kelas perlu didefinisikan menggunakan kata kunci class. Bentuk umumnya adalah sebagai berikut :

class NamaKelas :

# badan kelas

Program 3, menunjukkan pendefinisian kelas di dalam Python

Program #3

class segitiga :

def \_\_init\_\_(self) :

self.alas = 0

self.alas = 0

Pada kode dalam program 3, kelas didefinisikan dengan nama segitiga yang memiliki atribut alas dan tinggi. Tetapi dalam hal ini metode tambahan pada kelas tersebut belum didefinisikan. Satu-satunya metode yang didefinisikan adalah \_\_init\_\_() (diawali dan diakhiri oleh dua tanda underscore). Dalam Python, metode \_\_init\_\_() digunakan untuk melakukan inisialisasi nilai ke dalam atribut-atribut yang dimiliki oleh kelas.

self adalah suatu nama yang digunakan untuk mengacu ke objek dari kelas yang didefinisikan. self.alas menyatakan bahwa variabel alas adalah atribut dari kelas segitiga, bukan variabel lokal yang di definisikan di dalam metode \_\_init\_\_(). Setiap metode yang didefinisikan di dalam kelas parameter pertamannya selalu berupa self.

* + 1. Membuat Objek

Setelah kelas didefinisikan, maka objek dapat dibuat dari kelas tersebut menggunakan bentuk umum :

referensi = NamaKelas()

Untuk membuat objek dari kelas segitiga, gunakan kode berikut :

obj = segitiga()

Kode di atas akan menciptakan objek dari kelas segitiga di dalam memori komputer (dalam suatu alamat tertentu) dan objek tersebut ditunjuk oleh referensi atau variabel obj. Untuk mengisi nilai ke dalam atribut alas dan tinggi dari objek yang dibuat, gunakan operator titik (.), seperti berikut :

obj.alas = 5  # mengisi nilai 5 ke dalam atribut alas

obj.tinggi = 3 # mengisi nilai 5 ke dalam atribut tinggi

atribut-atribut tersebut dapat juga digunakan untuk operasi-operasi yang dibutuhkan oleh program, misalnya untuk perhitungan.

luas = (obj.alas \* obj.tinggi)/2

Program 4 adalah contoh kode lengkap yang akan menunjukkan cara pendefinisian kelas segitiga, pembuatan objek dari kelas tersebut, dan akses atribut-atributnya.

Program #4

class segitiga :

def \_\_init\_\_(self) :

self.alas = 0.0

self.tinggi = 0.0

# membuat objek dari kelas segitiga

Obj = segitiga()

#mengisi nilai ke dalam atribut-atribut obj

obj.alas = 5

obj.tinggi = 3

# menggunakan atribut-atribut untuk perhitungan

luas = (obj.alas \* obj.tinggi)/2

print("Luas segitiga : %.2f" %luas)

Pada saat pembuatan objek, Python secara implisit akan memanggil dua metode : \_\_new\_\_() dan \_\_init\_\_(). Metode \_\_new\_\_() bertugas menangani proses pembuatan objek di memori komputer dan \_\_init\_\_() bertugas untuk proses instalasi objek (mengisi nilai awal untuk objek bersangkutan). Untuk kasus-kasus normal, tidak perlu diutak-atik tentang metode \_\_new\_\_(). Untuk mengisi nilai awal dari atribut-atribut yang didefinisikan di dalam kelas, hanya perlu membuat implementasi untuk metode \_\_init\_\_(). Nilai-nilai yang digunakan untuk proses inisialisasi biasanya disertakan sebagai parameter metode \_\_init\_\_(). Program 5 berikut adalah modifikasi dari program 4 dimana metode \_\_init\_\_() didefinisikan dengan parameter.

Program #5

class segitiga :

def \_\_init\_\_(self, a, b) :

self.alas = a

self.tinggi = b

# membuat objek dari kelas segitiga

obj = segitiga(5, 3)

# menggunaka atribut-atribut untuk perhitungan

luas = (obj.alas \* obj.tinggi)/2

print("Luas segitiga : %.2f" %luas)

* + 1. Mendefinisikan Metode

Metode adalah fungsi yang didefinisikan di dalam suatu kelas. Metode menggambarkan operasi yang dapat dilakukan oleh suatu objek. Sebagai contoh, operasi perhitungan luas untuk objek segitiga dapat dijadikan sebagai metode. Proses pendefinisian metode tidak ada bedanya dengan pendefinisian fungsi. Suatu hal yang perlu diperhatikan hanyalah paramter awal dari suatu metode perlu diisi oleh self.

Program #6

class segitiga :

def \_\_init\_\_(self, a, t) :

self.alas = a

self.tinggi = t

#metode untuk menghitung luas

def luas(self) :

return(self.alas\*self.tinggi)/2

#metode untuk menghitung luas

def cetakluas(self) :

print("Luas segitiga : %.2f" %self.luas())

# membuat objek dari kelas segitiga

Obj = segitiga(5, 3)

# memanggil metode cetak luas

Obj.cetakluas()

Pada program 6, metode luas() dan metode cetakluas() didefinisikan di dalam kelas segitiga. Untuk memanggil metode, digunakan operator titik : obj.luas() dan obj.cetakluas() sama seperti pemanggilan atribut.

Normallnya atribut dari suatu kelas akan dimiliki oleh masing-masing objek dari kelas tersebut. Nilai atribut dari setiap objek dapat berbeda. Sebagai contoh, jika obj1 dan obj2 adalah objek dari kelas segitiga, maka nilai obj1.panjang bisa saja tidak sama dengan obj2.panjang. untuk membuat atribut yang nilainya dapat dibagi (share) ke seluruh objek yang ada, maka perlu didefinisikan atribut tersebut sebagai atribut statis. Atribut statis sering disebut sebagai *class variable* (variabel yang dimiliki oleh kelas), sedangkan atribut normal sering disebut sebagai *instance variable* (variable yang dimiliki oleh setiap instance atau objek).

Dalam Python, atribut statis perlu didefinisikan di luar metode dan tanpa diawali oleh self, seperti yang ditunjukkan oleh contoh program 7 berikut :

Program #7

class segitiga :

# atribut statis

objectCount = 0

def \_\_init\_\_(self, a, t) :

self.alas = a

self.tinggi = t

#menaikkan nilai atribut statis

Segitiga.objectCount += 1

#metode untuk menghitung luas

def luas(self) :

return(self.alas\*self.tinggi)/2

#metode untuk menghitung luas

def cetakluas(self) :

print("Luas segitiga : %.2f" %self.luas())

# membuat tiga objek dari kelas segitiga

Obj1 = segitiga(6, 5)

Obj2 = segitiga(5, 4)

Obj3 = segitiga(4, 3)

# memanggil atribut segitiga.objectCount

print("Jumlah objek dari kelas segitiga : %.d" %segitiga.objectCount)

Dalam program 7, di atas, atribut statis didefinisikan dengan nama objectCount. Atribut ini digunakan untuk menghitung jumlah objek dari kelas segitiga. Atribut statis diakses menggunakan nama kelas. Dengan demikian, tidak perlu membuat objek terlebih dahulu. Untuk mengakses atribut statis, gunakan bentuk umum berikut :

NamaKelas.NamaAtributStatis

Python mengizinkan untuk mengakses atribut statis melalui referensi, seperti yang ditunukkan oleh program 8.

Program #8

# membuat tiga objek dari kelas segitiga

Obj1 = segitiga(6, 5)

print("Jumlah objek saat ini : %.d" %obj1.objectCount)

Obj2 = segitiga(5, 4)

print("Jumlah objek saat ini : %.d" %obj2.objectCount)

Obj3 = segitiga(4, 3)

print("Jumlah objek saat ini : %.d" %obj3.objectCount)

Selain atribut, sifat statis ini juga dapat diterapkan untuk metode. Metode statis perlu didefinisikan menggunakan dekorator @staticmethod dan tidak disertai dengan self sebagai parameter pertamanya. Sama seperti atribut statis, untuk mengakses metode statis ini tidak perlu membuat objek terlebih dahulu. Pemanggilan metode statis dilakukan melalui nama kelas, dengan menggunakan bentuk umum :

NamaKelas.NamaMetodeStatis

Contoh penggunaanya dapat dilihat pada program 9.

Program #9

class segitiga :

def \_\_init\_\_(self, a, t) :

self.alas = a

self.tinggi = t

#metode untuk menghitung luas

def luas(self) :

return(self.alas\*self.tinggi)/2

#metode statis

@staticmethod

def daftarAtribut():

return(‘alas’,’tinggi’)

# memanggil metode statis

print("Daftar atribut dari kelas segitiga: %s" %repr(segitiga.daftarAtribut()))

Properti merupakan jenis atribut khusus yang nilainya dihitung ketika diakses. Properti ini sebenarnya adalah metode yang ditandai dengan dekorator @property. Sebagai contoh metode luas() dalam kelas segitiga dapat dijadikan sebagai properti luas. Contohnya dapat dilihat dalam program 10.

Program #10

class segitiga :

def \_\_init\_\_(self, a, t) :

self.alas = a

self.tinggi = t

#properti luas

@property

def luas(self) :

return(self.alas\*self.tinggi)/2

# membuat objek dari kelas segitiga

Obj = segitiga(6, 5)

# mengakses properti luas

print("Luas segitiga: %.2f" %obj.luas)

Properti luas yang dibuat hanya bersifat read-only (hanya dibaca).

Dalam keadaan normal, semua metode yang didefinisikan di dalam kelas akan bersifat publik. Ini berarti bahwa bagian program di luar kelas diizinkan untuk mengakses metode tersebut secara langsung. Pada contoh-contoh sebelumnya, metode luas() dan cetakLuas() dapat diakses dari luar kelas. Hal ini karena metode-metode tersebut bersifat publik. Pada kasus-kasus tertentu, kadang diperlukan suatu metode yang hanya digunakan di lingkungan kelas.

Untuk menjadikan metode tidak dapat diakses dari luar (seakan-akan bersifat private), dapat disertakan tanda underscore yang ditulis dua kali (\_\_) didepan metode tersebut.

* + 1. Kelas Turunan

Suatu kelas dapat diturunkan dari kelas yang sudah didefinisikan sebelumnya. Jika kelas B diturunkan dari kelas A, maka B akan mewarisi sifat-sifat (daftar atribut dan metode) yang dimiliki oleh kelas A. Selanjutnya, sifat-sifat baru dapat didefinisikan dalam kelas B. Dalam hal ini kelas A disebut sebagai kelas induk (superclass) dan kelas B disebut sebagai kelas anak (subclass).

Dalam Python, kelas turunan didefinisikan menggunakan bentuk umum, seperti berikut ini :

class NamaKelasTurunan(NamaKelasInduk):

*#badan kelas*

Jika NamaKelasInduk tidak disertakan, maka kelas tersebut secara otomatis dapat diturunkan dari kelas object, yaitu kelas moyang dari semua kelas yang terdapat dalam Python. Dengan demikian kelas segitiga yang sudah dibuat sebelumnya sebenarnya merupakan turunan dari kelas object. Untuk memeriksa apakah benar kelas segitiga adalah turunan dari kelas object dapat digunakan fungsi issubclass() berikut :

issubclass(segitga, object)

hasil eksekusi baris di atas akan menghasilkan True.

Program 11 merupakan contoh pembuatan kelas turunan. Dalam program 11, segitigasiku diturunkan dari segitiga. Maka dari itu, kelas segitigasiku akan mewarisi atribut alas dan tinggi serta metode luas() dan cetakLuas() yang dimiliki oleh kelas segitiga. Selanjutnya, kelas segitigasiku dapat mendefinisikan metode baru yang sebelumnya tidak dimiliki oleh kelas segitiga yaitu \_\_hipotenusa(), keliling(), dan cetak keliling().

Python mengizinkan pembuatan kelas turunan yang diwariskan yang diwariskan dari beberapa kelas induk. Sebagai contoh, jika kelas C diturunkan dari kelas A dan B, maka C akan mewarisi sifat-sifat milik kelas A dan B. proses ini dikenal dengan pewarisan jamak (multiple inheritance). Kelas turunan yang memiliki lebih dari satu kelas induk didefinisikan menggunakan bentuk umum berikut :

class NamaKelasTurunan(NamaKelasInduk1, NamaKelasInduk2, …):

*#badan kelas*

Program #11

#kelas induk

class segitiga :

def \_\_init\_\_(self, a, t) :

self.alas = a

self.tinggi = t

def luas(self) :

return(self.alas\*self.tinggi)/2

def cetakluas(self) :

print("Luas segitiga : %.2f" %self.luas())

#kelas turunan

class segitigasiku(segitiga)

def \_\_init\_\_(self, a, t) :

super().\_\_init\_\_(a,t)

def \_\_hipotenusa(self) :

import math

return math.sqrt((self.alas\*\*2)+(self.tinggi\*\*2))

def keliling(self) :

return self.alas +self.tinggi+self.\_hipotenusa()

def cetakKeliling(self) :

print("Keliling segitiga \t: %.2f" %self.keliling())

# membuat objek dari kelas segitigasiku

Obj = segitigasiku(3, 4)

# memanggil metode cetakLuas() dan cetakKeliling()

obj.cetakLuas()

obj.cetaKeliling()

Program 12 merupakan contoh pewarisan jamak dalam Python. Pada program 12 ini, kelas datetime diturunkan dari dua kelas induk, yaitu date dan time. Oleh karena itu, kelas datetime mewarisi atribut dan metode yang didefinisikan di dalam kelas date dan time.

Program #12

#kelas induk 1

class date :

def \_\_init\_\_(self, dd, mm, yyyy) :

self.setDate(dd, mm, yyyy)

def setDate(self, dd, mm, yyyy) :

self.dd = dd

self.mm = mm

self.yyyy = yyyy

def \_\_repr\_\_(self):

return "%d-%d-%d" %(self.dd, self.mm, self.yyyy)

#kelas induk 2

class time :

def \_init\_(self, hh, nn, ss) :

self.hh = hh

self.nn = nn

self.ss = ss

def setTime(self, hh, nn, ss) :

self.hh = hh

self.nn = nn

self.ss = ss

def \_\_repr\_\_(self):

return "%d-%d-%d" %(self.hh, self.nn, self.ss)

#kelas turunan

class datetime(date,time)

def \_\_init\_\_(self, dd, mm, yyyy, hh, nn, ss) :

date. \_\_init\_\_(self, dd, mm, yyyy)

time. \_\_init\_\_(self, hh, nn, ss)

def \_\_repr\_\_(self):

return "%s-%s" % \

(date.\_\_repr\_\_(self),

time.\_\_repr\_\_(self))

# membuat objek dari kelas datetime

Obj = datetime(12,04,2020,17,17,17)

print("Sekarang : %s" %repr(obj))

* 1. **Kegiatan Praktek**

1. Ketiklah semua program pada bagian 7.2 di atas dalam Pyhton shell anda, kemudian jalankan dan perhatikan hasilnya.
2. Buatlah kelas segiempat dan buatlah metode untuk menghitung luas segiempat
3. Buatlah kelas turunan bujursangkar dan persegipanjang
4. Buatlah satu buah metode baru untuk masing-masing kelas turunan.
5. Berikan komentar anda tentang hasil kegiatan praktek yang anda telah lakukan.
   1. **Tugas Mandiri**
6. Buatlah program untuk menentukan bilangan prima dari sekumpulan bilangan. Semua bilangan adalah bilangan positif. Buatlah dengan konsep objek oriented.
7. Buatlah program untuk menghitung untuk menghitung Indeks Prestasi Mahasiswa per Semester dengan konsep objek oriented

**REFERENSI**

Buku:

Raharjo, B. 2017. Belajar Singkat Pemrograman Python 3. Penerbit MODULA

Internet:

<https://www.w3schools.com/python>

<https://belajarpython.com/>

<https://www.tutorialspoint.com/python3>

<https://www.programiz.com/python-programming/tutorial>

<https://www.python-course.eu/python3_course.php>

<https://www.sololearn.com/Course/Python/>

Lain-Lain:

Selain sumber referensi yang telah disebutkan di atas, tidak menutup kemungkinan terdapat sumber-sumber lain yang dapat digunakan untuk menunjang pembelajaran pemrograman dasar dengan bahasa Python 3.

**LAMPIRAN 1**

**LAPORAN**

Setiap selesai melaksakan satu kegiatan praktikum, mahasiswa secara individu diwajibkan memasukkan laporan sebagai hasil pelaksanaan praktikum. Laporan praktikum dibagi menjadi dua jenis, yaitu laporan praktikum untuk setiap satu kegiatan praktikum dan laporan lengkap yang merupakan gabungan dari seluruh laporan kegiatan setiap praktikum. Sehingga laporan lengkap terdiri atas 8 laporan praktikum setiap kegiatan (yang telah disetujui oleh dosen pelaksana praktikum) sesuai dengan materi-materi yang ada pada panduan praktikum.

1. **Laporan 1 (satu) kegiatan praktikum**

Format laporan untuk satu kegiatan praktikum adalah sebagai berikut:

1. Kertas:

* Ukuran: A4
* Jenis: HVS atau jenis dibawahnya yang berwarna putih.
* Margin atas 3 cm , kiri 3 cm, kanan 2 cm, dan bawah 3 cm

1. Butir-butir komponen yang harus ada dalam laporan dan berurut seperti :

* Lembar pertama adalah lembar Assistensi (tersendiri)
* Lembar berikutnya berisi:

1. **Judul**: judul setiap kegiatan praktikum
2. **Praktikan**:

**Nama**: nama praktikan (peserta praktikum)

**NIM**: nim praktikan (peserta praktikum)

**Program Studi**: Program Studi S1 Teknik Elektro FT-UNG

1. **Tujuan Praktikum**: sesuai dengan panduan praktikum
2. **Dasar Teori**: sesuaikan dengan panduan praktikum
3. **Kegiatan Praktek**:

Butir ini berisi:

* hasil *listing program* dan tampilan keluaran (output running program).
* Komentar yang merupakan analisis praktikan terhadap hasil kegiatan praktikum yang telah dilakukan.

1. **Tugas**: berisi listing program dan tampilan keluaran (output running program)
2. Laporan ditulis tangan dengan rapi dengan ketentuan :

Laporan keseluruhan ditulis dengan menggunakan dengan tinta biru, kecuali :

* Listing program ditulis dengan tinta hijau
* Keluaran program ditulis dengn tinta merah*.*
* Tabel ditulis dengan menggunakan tinta hitam

1. **Laporan Lengkap**

Ketentuan umum laporan lengkap hasil pelaksanaan praktikum disesuaikan dengan format yang dikeluarkan oleh pihak Laboratorium Teknik Elektro FT-UNG.

**LAMPIRAN 2**

**PENILAIAN**

Komponen-komponen penilaian untuk keseluruhan kegiatan praktikum adalah sebagai berikut :

1. **Test Awal: 30 %**

Pada setiap awal kegiatan setiap praktikum akan diadakan ujian pendahuluan untuk mengetahui kesiapan praktikan untuk melaksanakan setiap kegiatan praktikum.

1. **Penguasaan Materi: 60 %**

Penilaian penguasaan materi ini berdasarkan kemampuan praktikan dalam menganalisa dan menyelesaikan setiap latihan dan tugas yang diberikan. Penilaian untuk komponen ini dilaksanakan pada saat praktikum dan hasil asistensi laporan per kegiatan praktikan

1. **Laporan: 10 %**

Aspek yang dinilai pada komponen ini adalah kesesuaian dengan format laporan yang telah ditetapkan, kerapian tulisan, dan ketepatan waktu pemasukan laporan setiap kegiatan praktikum ataupun laporan lengkap.