

LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
PERTEMUAN KE-1
Fungsi, Tuple, Library, Exception, dan Data Processing



Disusun oleh :

Nama	: Aura Nisa' Hidayat
NIM	: 21/482690/SV/19983
Hari, Tanggal	: Rabu, 9 Februari 2022
Kelas	: RI1AA
Dosen	: Anni Karimatul Fauziyyah, S.Kom., M.Eng.

LABORATORIUM PERANGKAT KERAS DAN LUNAK
PROGRAM SARJANA TERAPAN (DIV) TEKNOLOGI REKAYASA
INTERNET
DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
2022

Pemrograman Berorientasi Objek
Pertemuan Ke-1
Fungsi, Tuple, Library, Exception, Dan Data Processing

I. TUJUAN

Setelah praktikum ini diharapkan mahasiswa mampu :

- Memahami struktur kode dan konsep fungsi,
- Dapat melakukan pemanggilan fungsi dan mengembalikan hasil dari suatu fungsi,
- Memahami cakupan nama dan variabel,
- Memahami tupel dan tujuannya, serta membangun dan menggunakan tupel,
- Memahami library dan tujuannya, serta Menyusun dan menggunakan library,
- Memahami berbagai exception bawaan Python, melakukan pengujian kode, dan debugging.

II. LANDASAN TEORI

Python merupakan salah satu bahasa pemrograman yang mendukung multi paradigma pemrograman, seperti pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Di dalam Python terdapat berbagai fitur dan fungsi seperti halnya bahasa pemrograman lainnya. Fungsi ialah kumpulan perintah atau blok kode yang terorganisir dan dapat digunakan kembali untuk melakukan sebuah perintah atau action. Fungsi juga memberikan modularitas yang lebih baik untuk aplikasi dan tingkat penggunaan kode yang tinggi.

Terdapat beberapa aturan sederhana untuk mendefinisikan sebuah fungsi dengan Python :

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

Kemudian pada praktikum pemrograman berorientasi objek kali ini mahasiswa akan mempelajari *tuple*, *library*, dan *exception* pada Python. Berikut merupakan penjelasan singkatnya.

Tuple dalam Python merupakan struktur data yang digunakan untuk menyimpan sekumpulan data yang memiliki sifat *immutable* atau tidak dapat diubah dan dihapus. Namun dapat diisi dengan berbagai macam nilai dan objek. Perbedaan yang paling jelas

antara list dan tuple adalah sintaks yang digunakan untuk membuatnya menggunakan tanda kurung (). Sedangkan list menggunakan kurung siku [].

Library pada Python merupakan struktur data yaitu gabungan dari sekumpulan package dan modul dengan fungsi yang sama dan tujuan untuk memudahkan dalam membuah suatu aplikasi, tanpa harus menulis banyak kode. Library pada Python merupakan sebutan untuk sebuah kode program tambahan yang digunakan dalam kebutuhan tertentu. Library juga bukan merupakan list karena urutan datanya sama sekali tidak berarti.

Exception merupakan sebuah peristiwa yang menyebabkan kode berhenti karena adanya *ValueError* pada program yang mengganggu aliran normal intruksi program. Secara umum Ketika skrip Python menemukan situasi yang tidak dapat diatasi, exception menimbulkan pengecualian. Exception adalah objek Python yang mewakili kesalahan. Terdapat dua cabang pada exception, pertama dimulai dengan kata kunci *try*- untuk meletakkan kode yang dicurigai beresiko dan dapat dihentikan jika terjadi kesalahan. Sedangkan yang kedua adalah bagian dari kode yang dimulai dengan kata kunci *except* dirancang untuk menangani pengecualian.

III. ALAT DAN BAHAN

Yang perlu disiapkan sebelum melakukan praktikum adalah :

- Laptop atau computer,
- Jaringan internet,
- Aplikasi VSCode / Google Colabs / PyCharm.

IV. PERCOBAAN, HASIL DAN PEMBAHASAN

1. Percobaan Fungsi

✚ Pertama Fungsi Berparameter :

```
[ ] def message (number) :  
    print ("Enter a number:", number)  
  
    number = 1234  
    message (1)  
    print (number)
```

Hasil :

```
Enter a number: 1  
1234
```


Pembahasan :

Fungsi parameter merupakan suatu fungsi yang mengaktifkan mekanisme yang disebut *shadowing* atau membayangi variabel apapun dengan nama yang sama, tetapi hanya di dalam fungsi yang mendefinisikan parameter.


Dapat dilihat pada source code di atas parameter *number* memiliki nilai 1, 2, 3, dan 4. Kemudian line selanjutnya adalah “*message (1)*” yang

merupakan parameter dari **number = 1234** tersebut. Maka Ketika terdapat statement **print (number)** kemudian pada bagian output saya diperintahkan memasukan number yaitu 1 karena pada **message** adalah number (1) sebagai parameter. Lalu akan menghasilkan output dari semua number tersebut yaitu 1 2 3 4.

Kedua Fungsi Multi Parameter :

```
 def message (what, number):  
    print ("Enter", what, "number", number)  
  
#invoke the function  
message ("telephone", 11)  
message ("price", 5)  
message ("number", "number")
```


Hasil :


```
 Enter telephone number 11  
Enter price number 5  
Enter number number number
```

Pembahasan :

Fungsi multi parameter, yang mana berarti ketika akan menjalankan suatu fungsi akan membutuhkan dua argumen sebagai parameternya. Parameter baru pertama dimaksudkan untuk memanggil nama nilai yang diinginkan.

Ketiga :

```
 #1  
def my_function (a, b, c):  
    print (a, b, c)  
|  
my_function (1, 2, 3)
```

```
 #2  
def introduction (first_name, last_name):  
    print ("Hello, my name is", first_name, last_name)  
  
introduction ("Skywalker", "Luke")  
introduction ("Quick", "Jesse")  
introduction ("Kent", "Clark")
```

```
[27] #3  
def introduction (first_name, last_name):  
    print ("Helo, my name is", first_name, last_name)  
  
introduction ("Luke", "Skywalker")  
introduction ("Jesse", "Quick")  
introduction ("Clark", "Kent")
```

Hasil :

#1

1 2 3

#2

```
☞ Hello, my name is Skywalker Luke  
Hello, my name is Quick Jesse  
Hello, my name is Kent Clark
```

#3

```
Helo, my name is Luke Skywalker  
Helo, my name is Jesse Quick  
Helo, my name is Clark Kent
```

Pembahasan :

Pada fungsi ini menggunakan teknik parameter posisi yang memberikan argument ke-i (pertama, kedua, dan seterusnya) ke dalam parameter fungsi ke-i (pertama, kedua, dan seterusnya). Sedangkan argument yang diteruskan dengan cara ini disebut argument posisi.

Pertanyaan :

Apa perbedaan hasil source code pada percobaan fungsi 3?

Saya merasa tidak ada perbedaan antara ke tiga source code pada percobaan fungsi ke-3 ini. Karena sudah sesuai dengan ketentuan parameter posisi memberikan argument sesuai urutannya.

Keempat :

```
[2] def introduction (first_name, last_name):  
    print ("Hello, my name is", first_name, last_name)  
  
    introduction (first_name = "James", last_name = "Bond")  
    introduction (last_name = "Skywalker", first_name = "Luke")
```

Hasil :

```
Hello, my name is James Bond  
Hello, my name is Luke Skywalker
```

Pembahasan :

Dalam source code percobaan fungsi ke empat ini menggunakan teknik argumen kata kunci. Karena source code ini menggunakan argument kata kunci yang mana menawarkan konverensi lain untuk meneruskan argument yang ditentukan oleh namanya.

Pada source code ini terdapat 2 kata kunci yaitu "first_name" dan "last_name" kemudian terdapat statement print ("Hello, my name is", first_name, last_name). Dan pada line selanjutnya terdapat introduction yang akan mengkonverensi kata kunci tersebut. Ketika kata kunci di balik seperti pada line terakhir maka output akan menunjukkan hasil yang sesuai dengan argumen kata kunci sebelumnya.

✚ Kelima :

```
[3] def adding (a, b, c):  
    print (a, "+", b, "+", c, "=", a + b + c)  
  
    #call the adding function here.  
    adding (1, 2, 3)  
    adding (c = 1, a = 2, b = 3)  
    adding (3, c = 1, b = 2)  
    adding (4, 3, c = 2)
```

Hasil :

```
1 + 2 + 3 = 6  
2 + 3 + 1 = 6  
3 + 2 + 1 = 6  
4 + 3 + 2 = 9
```

Pembahasan :

Pada source code kelima terdapat penggabungan antara dua argumen yaitu meletakkan argumen posisi sebelum argumen kata kunci. Dapat dilihat bahwa terdapat statement `print (a, "+", b, "+", c, "=", a + b + c)`. Dimana pada statement tersebut sudah menggunakan teknik argumen posisi dan huruf a, b, dan c merupakan kata kunci yang nantinya akan memanggil angka yang sudah ditentukan pada line selanjutnya yaitu pada `adding`. Maka ketika menambahkan argumen fungsi pada `adding` akan menghasilkan output seperti pada gambar di atas dan sesuai urutan posisi yang telah ditetapkan.

✚ Keenam (Return, List, Fungsi) :

```
#1  
def list_sum(lst):  
    s = 0  
  
    for elem in lst:  
        s += elem  
  
    return s  
print (list_sum(5))
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-7-0db4b12e07c9> in <module>()  
7  
8     return s  
----> 9 print (list_sum(5))  
  
<ipython-input-7-0db4b12e07c9> in list_sum(lst)  
3     s = 0  
4  
----> 5     for elem in lst:  
6         s += elem  
7  
TypeError: 'int' object is not iterable
```

SEARCH STACK OVERFLOW

```
#1
def list_sum(lst):
    s = 0

    for elem in lst:
        s += elem

    return s
print (list_sum([5, 4, 3]))
```

Hasil :

12

```
#2
def strange_list_fun(n):
    strange_list = []

    for i in range (0, n):
        strange_list.insert (0,i)

    return strange_list

print (strange_list_fun(5))
```

Hasil :

[4, 3, 2, 1, 0]

Pembahasan :

Pada source code percobaan ke enam ini menggunakan parameter fungsi tetapi nilai integer tunggal tidak boleh diulang melalui for loop. Pada percobaan pertama disini saya mengalami error, tetapi saya belum bisa menemukan solusi untuk mengatasi eror tersebut.

Pertanyaan :

Jelaskan perbedaan 2 source tersebut!

Pada percobaan pertama menggunakan **list_sum** yang berarti urutan angka tersebut harus dijumlahkan, maka akan menghasilkan output satu angka yaitu hasil dari penjumlahan tersebut.

Sedangkan pada percobaan kedua memiliki arti untuk menampilkan output list angka. Strange_list_fun(5) akan menampilkan output angka sebelum angka 5 jadi 4, 3, 2, 1, 0.

✚ Ketujuh (Fungsi kasus BMI) :

```
def bmi(weight, height):
    return weight / height ** 2

print (bmi(52.5, 1.65))
```

Hasil :

```
19.283746556473833
```

Pembahasan :

Dalam percobaan di atas menggunakan fungsi *return* untuk mengembalikan nilai dari variabel. Dimana secara garis besar fungsi *return* pada Python ialah mengembalikan nilai. Dapat dilihat pada baris atau line kedua bahwa terdapat perintah **return weight / height ** 2** sebagai rumusnya, kembali pada baris pertama bahwa **bmi** memiliki varriable (**weight, height**). Maka pada perintah terakhir untuk mencetak hasil output hanya perlu memanggil kata kunci **bmi** serta nilainya. Karena **bmi (weight, height)** telah di *return* menjadi rumus pada baris ke dua tersebut.

✚ Kedelapan Fungsi dengan or atau and :

#1

```
1 def is_a_triangle(a,b,c):
2     if a+b<=c:
3         return False
4     if b + c <= a:
5         return False
6     if c + a <= b :
7         return False
8     return True
9 print(is_a_triangle(1,1,1))
10 print(is_a_triangle(1,1,3))
```

Hasil :

```
True
False
```

#2

```
[ ] 1 def is_a_triangle(a,b,c):
2     return a + b > c and b + c > a and c + a > b
3 print(is_a_triangle(1,1,1))
4 print(is_a_triangle(1,1,3))
```

Hasil :

```
True
False
```

#3

```
[ ] 1 def is_a_triangle(a,b,c):
2     if a + b <= c or b + c <= a or c + a <= b:
3         return False
4     return True
5 print(is_a_triangle(1,1,1))
6 print(is_a_triangle(1,1,3))
```


Hasil :

```
True
False
```

Pembahasan :

Or dan **and** merupakan operator logika atau operator yang digunakan untuk membuat kesimpulan logis dari dua kondisi Boolean (*true* atau *false*). Dalam bahasa Python sendiri terdapat 3 operator logika yaitu, or, and, dan not.

Jika dilihat pada source code di atas juga menggunakan percabangan atau fungsi logika yaitu **if**. Kemudian operator or hanya akan menghasilkan True jika salah satu operand bernilai False, dan operator or hanya bernilai False jika kedua operand juga bernilai False. Sedangkan and hanya akan menghasilkan True ketika kedua operand bernilai True, selain itu hasilnya False.

✚ Kesembilan (Rumus segitiga):

```
def is_a_triangle(a, b, c):
    return a + b > c and b + c > a and c + a > b

def heron(a, b, c):
    p = (a + b + c) / 2
    return (p * (p-a) * (p-b) * (p-c)) ** 0.5

def area_of_triangle(a, b, c):
    if not is_a_triangle(a, b, c):
        return None
    return heron(a, b, c)

print(area_of_triangle(1., 1., 2. ** .5))
```

Hasil :

```
0.49999999999999983
```

Pembahasan :

Dapat dilihat pada source code percobaan ke Sembilan ini terdapat **def** karena merupakan suatu fungsi. Kemudian juga menggunakan fungsi **return** untuk mengembalikan suatu nilai, dan juga operator **and**.

Arti **return** disini untuk mengembalikan nilai yang tersimpan di dalam variabel pada line atau baris sebelumnya.

✚ Tuple dan Library :

```
#1
my_tuple = tuple((1, 2, "string"))
print(my_tuple)

my_list = [2, 4, 6]
print(my_list)           #outputs: [2, 4, 6]
print(type(my_list))     #outputs: <class 'list'>
tup = tuple(my_list)
print(tup)               #outputs: (2, 4, 6)
print(type(tup))         #outputs: <class 'tuple'>
```

Hasil :

```
(1, 2, 'string')
[2, 4, 6]
<class 'list'>
(2, 4, 6)
<class 'tuple'>
```

```
[ ] #2
my_tuple = (1, 10, 100)

t1 = my_tuple + (1000, 10000)
t2 = my_tuple * 3

print(len(t2))
print(t1)
print(t2)
print(10 in my_tuple)
print(-10 not in my_tuple)
```

Hasil :

```
9
(1, 10, 100, 1000, 10000)
(1, 10, 100, 1, 10, 100, 1, 10, 100)
True
True
```

Pembahasan :

Selanjutnya pada program ini menggunakan Tuple dan Library di dalam source codenya. Tuple merupakan urutan seperti daftar atau list hanya saja tidak dapat diubah. Tuple ini juga menggunakan tanda kurung (), sedangkan list Python menggunakan tanda kurung siku [].

Untuk mengakses nilai dalam Tuple dapat menggunakan tanda kurung siku beserta indeks nya. Seperti pada line ke tiga percobaan pertama. Sedangkan pada source code percobaan kedua dapat dilihat bahwa terdapat fungsi **len** yang digunakan untuk memberikan total panjang tuple seperti dapat dilihat pada hasil output.

Pada 2 line terakhir juga terdapat expression **in** dan **not in** yang akan menghasilkan output True atau False. Maka artinya **10 in my_tuple** jawabannya adalah benar atau True, dan **-10 not in my_tuple** jawabannya juga True. Karena dalam program tersebut adalah (1, 10, 100, 1000, 10000) ada 10, dan tidak ada -10.

🚦 Percobaan Library 1:

```
#1
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}
phone_numbers = {'boss': 5551234567, 'Suzy': 22657854310}
empty_dictionary = {}

#print the value here.
print (dictionary['cat'])
print (phone_numbers['Suzy'])
```

Hasil :

```
chat
22657854310
```

```
#2
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

for key in dictionary.keys():
    print (key, "->", dictionary[key])
```

Hasil :

```
cat -> chat
dog -> chien
horse -> cheval
```

```
[ ] #3
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}
words = ['cat', 'lion', 'horse']

for word in words:
    if word in dictionary:
        print (word, "->", dictionary[word])
    else:
        print (word, "is not in dictionary")
```

Hasil :

```
cat -> chat
lion is not in dictionary
horse -> cheval
```

Pembahasan :

Source code yang pertama adalah mengakses nilai dalam dictionary. Artinya disini bahwa kata kunci **cat** memiliki *value* **chat**. Maka ketika statement print dikeluarkan untuk memanggil value dari dictionary cat akan menghasilkan output **chat**. Kemudian pada line kedua phone numbers juga terdapat key beserta valuenya. Maka Ketika statement print phone numbers Suzy akan menghasilkan output valuenya yaitu 22657865310 tersebut.

Source code kedua, dapat dilihat bahwa terdapat perulangan dalam program tersebut. Yang artinya akan memanggil dan menampilkan key dan value sebuah dictionary satu per satu. Maka dapat dilihat hasil outputnya seperti pada gambar di atas.

Source code ketiga juga sama terdapat *dictionary* yang dikenal memiliki *value* dan *key*. Pada program ini akan mengakses nilai dalam dictionary Python dengan perulangan. **Print (word, "->", dictionary[word])** artinya adalah perintah untuk menampilkan word yang disini sebagai kata kunci (key) yang terdapat pada nilai list dictionary. Maka output menghasilkan cat -> chat yang merupakan True. Sedangkan else print (word, "is not in dictionary") maka akan menampilkan output yang salah bahwa word lion tidak ada pada dictionary.

🚩 Percobaan Tuple dengan Library :

```

▶ school_class = {}
while True:
    name = input("Enter the student's name:")
    if name == '':
        break

    score = int(input("Enter the student's score (0-10):"))
    if score not in range (0, 11):
        break

    if name in school_class:
        school_class[name] += (score,)
    else:
        school_class[name] = (score,)

for name in sorted (school_class.keys()):
    adding = 0
    counter = 0
    for score in school_class[name]:
        adding += score
        counter += 1
    print (name, ":", adding/counter)

```

Hasil :

```

▶ Enter the student's name:yasa
Enter the student's score (0-10):8
Enter the student's name:dindung
Enter the student's score (0-10):1
Enter the student's name:indy
Enter the student's score (0-10):12
dindung : 1.0
yasa : 8.0

```

Pembahasan :

Pada source code di atas merupakan contoh sederhana yang menunjukkan bagaimana tuple dan library dapat bekerja sama. Dalam program ini yaitu menghitung nilai rata-rata siswa, dimana nanti pada run bagaian output meminta untuk input nama beserta skor nilainya. Kemudian list semua nama Bersama skor rata-rata yang dievaluasi kemudian harus dikeluarkan.

Expection :

```

▶ #percobaan 1
try:
    value = int(input('Enter a natural number:'))
    print ('The reciprocal of', value, 'is', 1/value)
except:
    print ('I do not know what to do.')

```

Hasil :

```

Enter a natural number:4
The reciprocal of 4 is 0.25

```

```

#percobaan 2
while True :
    try:
        number = int(input("Enter an int number:"))
        print (5/number)
        break
    except ValueError:
        print ("Wrong value.")
    except ZeroDivisionError:
        print ("Sorry. I cannot divide by zero.")
    except:
        print ("I don't know what to do...")

```

Hasil ;

```

I don't know what to do...
Enter an int number:0
Sorry. I cannot divide by zero.
Enter an int number:a
Wrong value.
Enter an int number:5
1.0

```

```

#percobaan 3
try:
    value = int(input("Enter a value:"))
    print (value/value)
except ValueError:
    print ("Bad input...")
except ZeroDivisionError:
    print ("Verru bad input...")
except:
    print ("Booo!")

```

Hasil :

```

Enter a value:1
1.0

```

Pembahasan :

Pada program-program exception ini seperti sebuah program kalkulator sederhana yang hanya membutuhkan input angka. Ketika tidak sengaja memasukkan input huruf maka pasti akan terjadi error.

Percobaan pertama merupakan input **int**. karena bisa saja memasukkan input yang salah maka sebelumnya terdapat *try*. Dan terdapat statem **except** yang akan dieksekusi ketika muncul error karena input yang tidak sesuai seperti huruf. Dan outputnya akan menjadi “I don’t know what to do”

Percobaan kedua menambahkan **except zerodivissionerror**, dimana artinya error ini akan muncul ketika terdapat operasi pembagian bilangan dengan angka nol. Dapat dilihat saat memasukkan nilai 0 akan muncul “Sorry. I cannot divide by zero.” Dan akan eror juga ketika memasukkan selain

integer atau angka. Tetapi tidak terjadi error ketika saya mencoba memasukkan angka 5 atau selain angka 0 ke dalam output tersebut.

Percobaan ketiga dengan memasukkan **nilai 0**, sama halnya dengan percobaan kedua. Tetapi ketika kita memasukkan nilai 0 akan menghasilkan output “Verru bad input...”

V. KESIMPULAN

Setelah melakukan praktikum pemrograman berorientasi objek kali ini dapat ditarik kesimpulan bahwa :

- 1) Fungsi pada Python memiliki syntax yang diawali dengan *def*.
- 2) *Return* pada fungsi di Python digunakan untuk mengembalikan suatu nilai.
- 3) Tipe data Tuple dapat dituliskan di dalam tanda kurung () dan dengan tanda koma sebagai pemisah.
- 4) Dictionary merupakan urutan yang berisi key dan value yang dipisahkan oleh titik dua dan tertutup dalam kurung kurawal.
- 5) Try-Except merupakan salah satu kondisi pada python yang penggunaannya mirip dengan if-else yang digunakan untuk menguji apakah sebuah statement menghasilkan error atau tidak.
- 6) ZeroDivisionError artinya error ini akan muncul ketika terdapat operasi pembagian bilangan dengan angka nol
- 7) Percobaan ini dapat dilakukan melalui beberapa platform seperti Google Colab, software Visual Studio Code serta PyCharm.

DAFTAR PUSTAKA

- Andre. (2021, November 23). *Tutorial Belajar Python Pat 32 : Fungsi Perintah Return pada Function Python*. Retrieved from Dunia Ilkom: <https://www.duniailkom.com/tutorial-belajar-python-fungsi-perintah-return-pada-function-python/>
- Fajar, R. (2016, Agustus 13). *Mengenal Statement Try Except di Python*. Retrieved from CodePolitan: <https://www.codepolitan.com/mengenal-statement-try-except-di-python>
- Huda, N. (2021, Februari 5). *Python Dasar : Fungsi (def)*. Retrieved from Jago Ncoding: <https://jagongoding.com/python/dasar/fungsi/#:~:text=Fungsi%20pada%20python%20adalah%20kumpulan,dipanggil%20berkali%20kali%20secara%20independen.>
- Muhardian, A. (2022, Februari 14). *Belajar Python : Apa itu Tuple dalam Python?* Retrieved from Petani Kode: <https://www.petanikode.com/python-tuple/>
- Python, T. (2019, Juli 28). *Tutorial Belajar Python Part 19 : Jenis-jenis Operator Logika Python*. Retrieved from Dunia Ilkom: <https://www.duniailkom.com/tutorial-belajar-python-jenis-jenis-operator-logika-python/>
- Rumah, B. D. (2021, Mei 18). *Library Python Kenali Perbedaan Module, Package, dan Library Pada Python*. Retrieved from DQlab: <https://www.dqlab.id/library-python-kenali-perbedaan-module-package-dan-library-pada-python#:~:text=Library%20pada%20Python%20merupakan%20gabungan,yang%20digunakan%20dalam%20kebutuhan%20tertentu.>
- Wahyu. (n.d.). *Menggunakan Try Except Pada Python*. Retrieved from Jurnal Mas Wahyu: <https://www.jurnalmaswahyu.com/2020/12/menggunakan-try-except-pada-python.html>