

Bahan Ajar

Chapter 13



Materi Pembelajaran

Matakuliah :

# INTERFACING PERIPHERAL

Kode Matakuliah : SKO 20416

Prodi : SISTEM KOMPUTER

Dosen Pengampu Matakuliah:

Bayu Nugroho, S.Kom., M.Eng

# Tables of Content

## Python Try Except

- Many Exceptions
- Else
- Finally
- Raise an exception

## Tugas Mandiri



# Python Try Except

The **try** block lets you test a block of code for errors.

The **except** block lets you handle the error.

The **else** block lets you execute code when there is no error.

The **finally** block lets you execute code, regardless of the result of the try- and except blocks.

When an error occurs, or exception as we call it, Python will normally stop and generate an error message. These exceptions can be handled using the try statement:

## Example

The **try** block will generate an exception, because **x** is not defined:

```
try:
    print(x)
except:
    print("An exception occurred")
```

# Many Exceptions

You can define as many exception blocks as you want, e.g. if you want to execute a special block of code for a special kind of error:

## Example

Print one message if the try block raises a `NameError` and another for other errors:

```
try:
    print(x)
except NameError:
    print("Variable x is not defined")
except:
    print("Something else went wrong")
```

# Else

You can use the else keyword to define a block of code to be executed if no errors were raised:

## Example

In this example, the `try` block does not generate any error:

```
try:  
    print("Hello")  
except:  
    print("Something went wrong")  
else:  
    print("Nothing went wrong")
```

# Finally

The finally block, if specified, will be executed regardless if the try block raises an error or not.

## Example

```
try:  
    print(x)  
except:  
    print("Something went wrong")  
finally:  
    print("The 'try except' is finished")
```

# Finally

This can be useful to close objects and clean up resources:

## Example

Try to open and write to a file that is not writable:

```
try:
    f = open("demofile.txt")
    try:
        f.write("Lorum Ipsum")
    except:
        print("Something went wrong when writing to the file")
    finally:
        f.close()
except:
    print("Something went wrong when opening the file")
```

# Raise an exception

As a Python developer you can choose to throw an exception if a condition occurs.

To throw (or raise) an exception, use the raise keyword.

## Example

Raise an error and stop the program if x is lower than 0:

```
x = -1

if x < 0:
    raise Exception("Sorry, no numbers below zero")
```

# Raise an exception

The raise keyword is used to raise an exception.

You can define what kind of error to raise, and the text to print to the user.

## Example

Raise a TypeError if x is not an integer:

```
x = "hello"

if not type(x) is int:
    raise TypeError("Only integers are allowed")
```

## **Tugas Mandiri (teori):**

1. Can you wrap your mainline function within a try except statement? Explained
2. I've heard I can raise my own exceptions. Is that true? Explained

## **Tugas Mandiri (prakt):**

Tuliskan script program python dengan fungsi **Try, Except, Else, Finaly** untuk memodifikasi 6 buah LED di Raspberry Pi.

end

