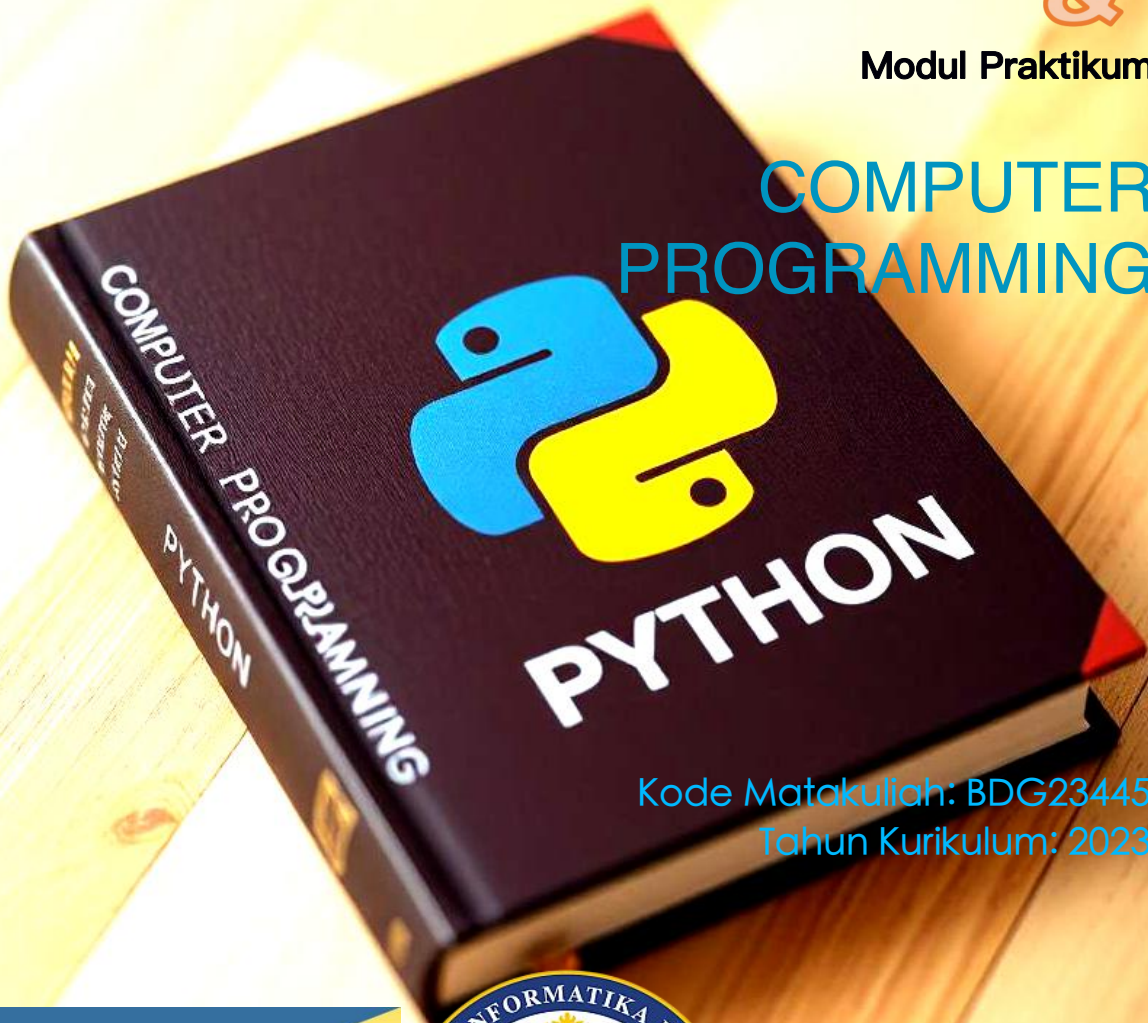




COMPUTER PROGRAMMING



Kode Matakuliah: BDG23445
Tahun Kurikulum: 2023



Penyusun:
Bayu Nugroho, S.Kom., M.Eng



FAKULTAS EKONOMI & BISNIS
INSTITUT INFORMATIKA DAN BISNIS DARMAJAYA
2025

Modul 10

Konsep modul dan paket dalam Python

1. Konsep dan penerapan exception handling Dalam Python

Exception Handling

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:

File Handling

The key function for working with files in Python is the `open()` function.

The `open()` function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

`"r"` - Read - Default value. Opens a file for reading, error if the file does not exist

`"a"` - Append - Opens a file for appending, creates the file if it does not exist

`"w"` - Write - Opens a file for writing, creates the file if it does not exist

`"x"` - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

`"t"` - Text - Default value. Text mode

`"b"` - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
```

The code above is the same as:

```
f = open("demofile.txt", "rt")
```

Because `"r"` for read, and `"t"` for text are the default values, you do not need to specify them

2. Struktur try, except, else, dan finally

Example

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

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

Since the try block raises an error, the except block will be executed.

Without the try block, the program will crash and raise an error:

Example

This statement will raise an error, because `x` is not defined:

```
print(x)
```

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")
```

See more Error types in our [Python Built-in Exceptions Reference](#).

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")
```

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")
```

The program can continue, without leaving the file object open.

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")
```

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")
```