Function

→ group of commands grouped & executed when called

→ reusable block of code

```
def function_name(arguements):
    _____
    _____
    _____
```

2 kinds of Function → ① Built-in Functions ② user-defined function

→ function may have a return statement

⊛ Default Arguements — def function_name(arguement = "____"):

⊛ Order of Arguements
- order of arguements is vital
- or specify arguement_name = value

⊛ Arbitrary Arguement

- variable-length arguement → number of values passed isn't specified

def function_name(*arguement_name):

Default Arguements

```
def function_name(arguement = "____"):
```

Order of Arguements

- order of arguements is vital
- or specify arguement_name = value

Arbitrary Arguement

- variable-length arguement → number of values passed isn't specified

```
def function_name(*arguement_name):
```

Arbitrary Keyword Arguement

- if keyword arguements number is unknown

```
def function_name(**name):
```

Lambda Function (anonymous function)

```
variable = lambda arguement : statement
```

Errors & Exceptions

→ **Syntax Errors** occurs when python can't interpret the code

→ **Exceptions** occurs when unexpected things occur during program execution

→ **Try / Except / Finally** used to handle exceptions

use while loop to allow user to re-enter input if incorrectly inputed the 1st time

⊛ you can specify the type of exception & deal with every kind seperately

◆ you can access the error message except ____ as e :

print (e)

```
print(e)
```

Files

→ Reading a File

1) Open File

```
f = open('path/file_name', 'r')
```

2) Read File

```
d = f.read()
```

3) Close File

```
f.close()
```

→ Writing to a File

1) Open

```
f = open('      ', 'w')
```

2) Write

```
f.write("_")
```

3) Close

```
f.close
```

→ Special Syntax

```
with open ('      ', 'r') as f :
          ('      ', 'w')
```

It automatically closes file

Import

```
import python_file as alias_name
```

Import

```
import python_file as alias_name
```

import an individual function(s)

```
from module_name import object_name (s)
```

or

```
import package_name . submodule_name
```