IT1303 - Programming

FUNCTIONS AND MODULES

Functions

Learning Outcome

Identify user-defined functions and modules.

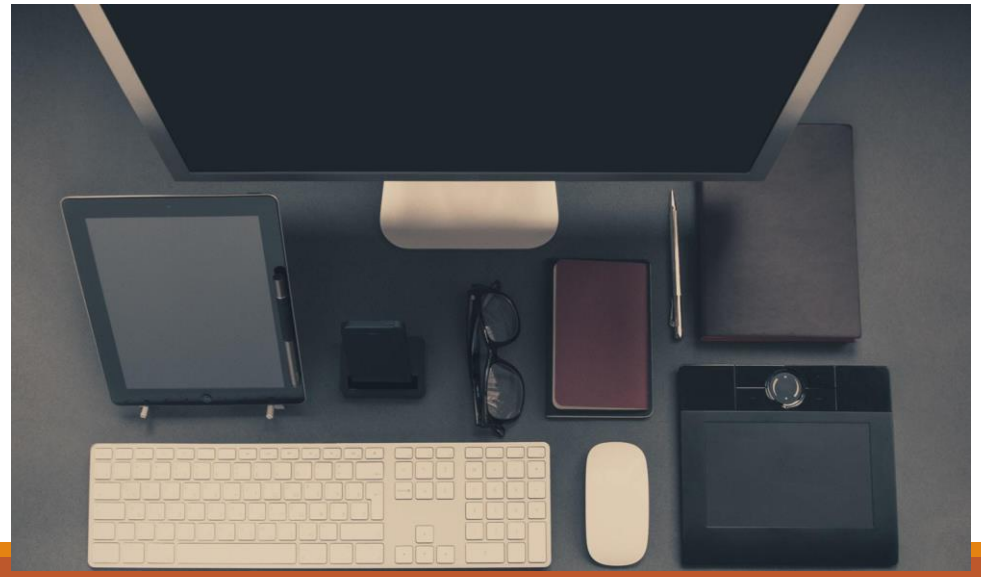
Create user-defined functions and modules to solve programming problem.

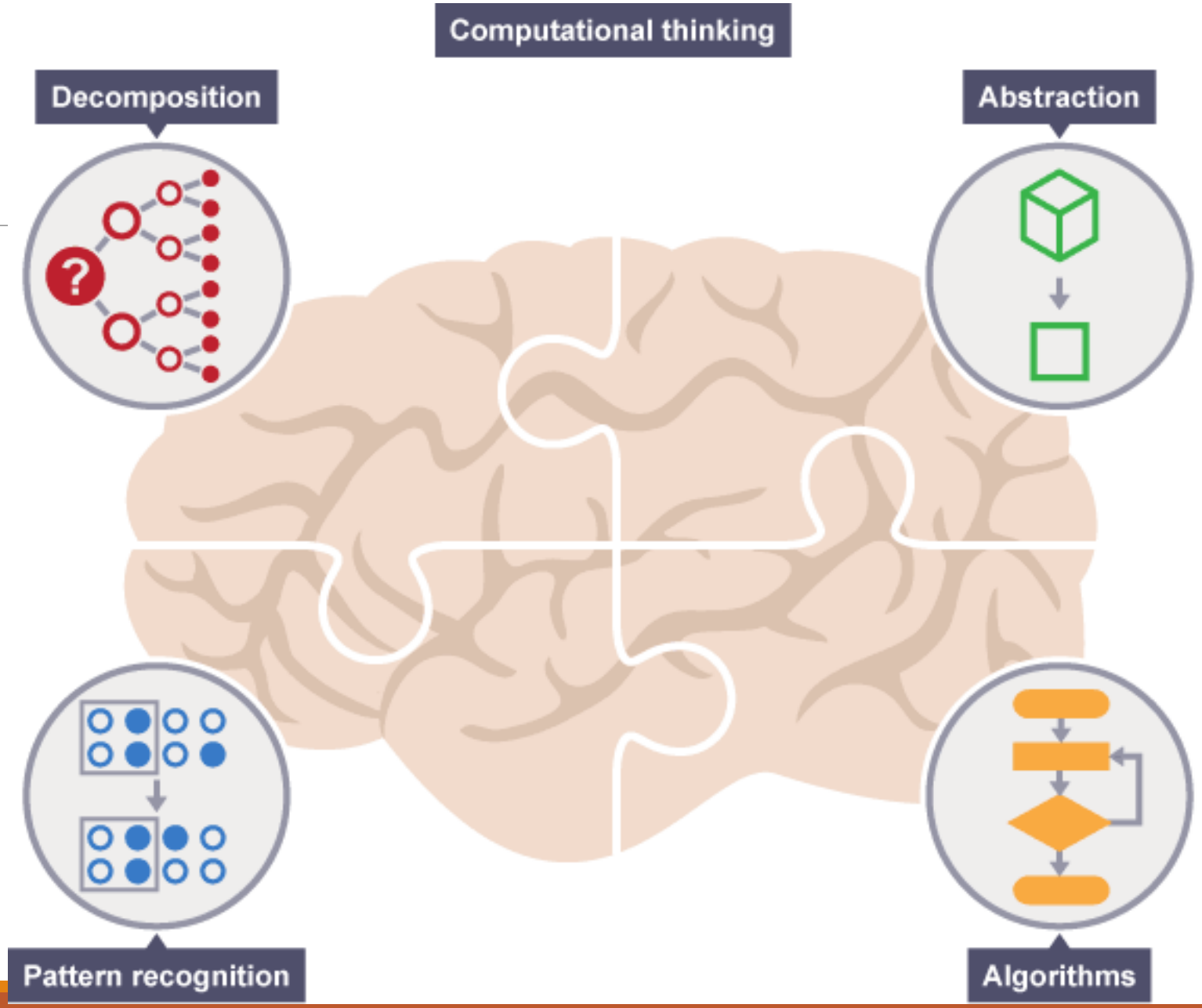
Functions

Group of statements that exist within a program for the purpose of performing a specific task.

Provide better modularity for your application and a high degree of code reusing

print()
input()




Computational thinking
Decomposition
Abstraction
Pattern recognition
Algorithms

Types of Function

Void Functions

Executes the statements it contains and terminates.

Value-Returning Functions

Executes the statements that it contains and returns a value back to the statement that called it.

input() print() int() len() range()

Defining and Calling a Function

These statements define the hello function

This statement calls the hello function

```
# This is a function named hello
def hello():
    print('Hello World')

# Call the function
hello()
```

Defining and Calling a Function

First function, main that calls hello() function

hello() function is defined here

Call main() function

```
def main():
    print('Calling the hello function...')
    hello()
    print('End of main function')

def hello():
    print('Hello World')

main()
```

Defining and Calling a Function

```
def iprint():
    print("A")

def main():
    print("B")
    iprint()
    print("C")

print("D")
main()
print("E")
```

Lets try running this code.
What do you think will be printed out?

A
B
C
D
E

This gets printed out.
Do you know why?

```
D
B
A
C
E
```

Defining and Calling a Function

```
def iprint():
    print("A")

def main():
    print("B")
    iprint()
    print("C")

print("D")
main()
print("E")
```

Outcome

```
D
B
A
C
E
```

Local variable

```
def main():
    get_name()
    print('Hello', name)

def get_name():
    name = input('Enter your name: ')

main()
```

What is wrong with this program?

Local variable

```
def main():
    get_name()
    print('Hello', name)

def get_name():
    name = input('Enter your name: ')

main()
```

A local variable created inside a function can only be accessed by statements inside the function. It cannot be accessed by statments that are outside this function!

Passing Arguments to Functions

```
def main():
    val = int(input('Enter value: '))
    display_double(val)

def display_double(number):
    print(number * 2)

main()
```

argument

parameter

```
Enter value: 5
10
```

Argument vs Parameter

Argument

Any piece of data that is passed into a function when the function is called.

Parameter

A variable that receives from an argument that is passed into a function.

```
def main():
    val = int(input('Enter value: '))
    display_double(val)

def display_double(number):
    print(number * 2)
```

argument

parameter

Multiple Arguments Function

```
def main():
    x = 5
    y = 10
    display_sum(x, y)

def display_sum(val1, val2):
    sum = val1 + val2
    print(sum)

main()
```

```
15
```

2 arguments
x and y

2 parameters val1
and val2

Keyword Arguments

```
def main():
    value1=5
    value2=10
    display_sum(value2,value1)

def display_sum(value1,value2):
    print("val1=",value1)
    print("val2=", value2)

main()
```

Keyword arguments

2 parameters value1 and value2

You can mix and match keyword and positional arguments, but positional arguments must always come first!

Make changes to Parameter

```
def main():
    var = 99
    print('In main(), the initial value of var is', var)
    change(var)
    print('Back in main(), the new value of var is', var)

def change(value):
    value = 1
    print('In change(), the new value is', value)

main()
```

```
In main(), the initial value of var is 99
In change(), the new value is 1
Back in main(), the new value of var is 99
```

Value of var remain unchanged?

Global vs Local variables

```
number = 0

def main():
    global number
    number = 5
    display_number()

def display_number():
    print('The number is', number)

main()
```

```
The number is 5
```

The global keyword in main() function tells the interpreter that the main function intends to assign a value to the global variable, number.

⚠

Value Returning Function

```
import random
number = random.randint(1, 100)
print(number)
```

```
47
```

module name → random

function name → randint()

arguments → 1, 100

return type → int

Value Returning Function

```
def is_number(s):
    try:
        float(s)
    except ValueError:
        return False
    else:
        return True
s = input('Enter a number: ')
print(is_number(s))
```

```
Enter a number: 123.5
True
```

function → is_number()

arguments → string (s)

return type → boolean

Value Returning Function

```
def is_valid_phone(s):
    if len(s) == 8:
        if s[0] == '6' or s[0] == '9':
            return True
    return False

number = input('Enter phone number: ')
print(is_valid_phone(number))
```

```
Enter phone number: 67876722
True
```

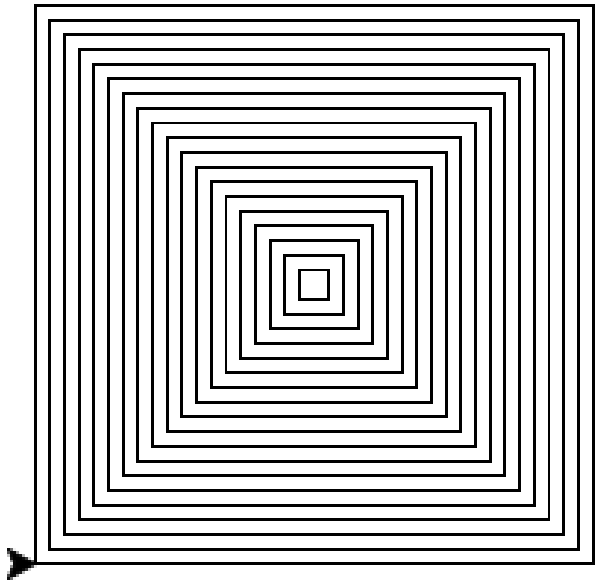
function →is_valid_phone()

arguments → string (s)

return type → boolean

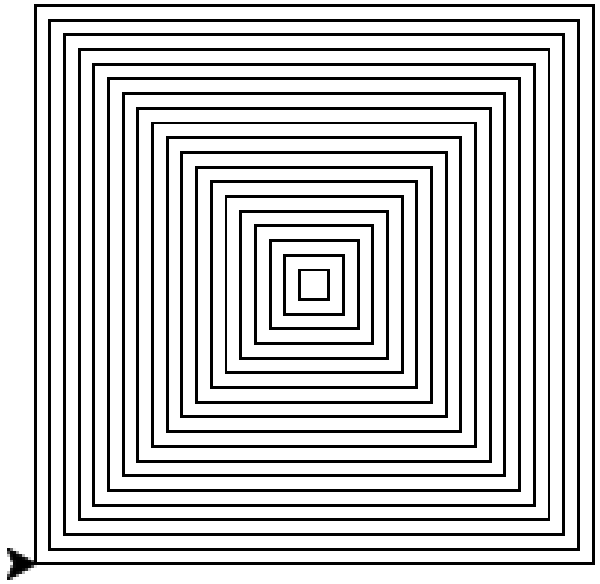
Modules

```
import turtle
for r in range(5, 100, 5) :
    turtle.penup()
    turtle.setpos(-r, -r)
    turtle.pendown()
    for i in range(4) :
        turtle.forward(2*r)
        turtle.left(90)
turtle.exitonclick()
```



Modules

```
import turtle
for r in range(5, 100, 5) :
    turtle.penup()
    turtle.setpos(-r, -r)
    turtle.pendown()
    for i in range(4) :
        turtle.forward(2*r)
        turtle.left(90)
turtle.exitonclick()
```



Modules

A file consisting of Python code.
Can define functions, classes and variables.
Can also include runnable code.

```
import turtle
for r in range(5, 100, 5) :
    turtle.penup()
    turtle.setpos(-r, -r)
    turtle.pendown()
    for i in range(4) :
        turtle.forward(2*r)
        turtle.left(90)
turtle.exitonclick()
```

Abstraction

Use import keyword

Easier to understand and use

Modules

Python's from statement lets you import all names from a module into the namespace

An easy way to import all the items from a module into the current namespace. ⚠⚠⚠

*from <module> import **

```
from turtle import *
forward(25)
left(120)
forward(25)
left(120)
forward(25)
exitonclick()
```

Locating Modules

When you import a module, the Python interpreter searches for the module in the following sequences:

1. The current directory
2. The directory in the shell variable PYTHONPATH
3. The default path where Python is installed

Create your own Module

Maintainability

Organize your code into different modules make it easier for you to maintain it. Make changes only at one location.

Reusability

With modules, you could reuse functions across different project.

Create your own Module

```
# validator.py
# returns True if string is a valid number
def is_number(string):
    try:
        float(string)
    except ValueError:
        return False
    else:
        return True

# returns True if string is a valid local phone number
def is_valid_phone(string):
    if len(string) == 8:
        if string[0] == '6' or string[0] == '9':
            return True
    return False
```

Using Module

```
# main.py
import validator

while True:
    value = input('Enter a valid phone number: ')
    if validator.is_valid_phone(value):
        print('Your phone number is', value)
        break
    else:
        print('Invalid phone number, please try again')
```

Using your module

import validator

```
import validator
value = input('Enter a valid number: ')
if validator.is_number(value):
  print('Valid number!')
```

import validator as val

```
import validator as va
value = input('Enter a valid number: ')
if va.is_number(value):
  print('Valid number!')
```

Summary

- Identify user-defined functions and modules.
- Create user-defined functions and modules to solve programming problem.