Khoi Duong

Prof. Yang

CE450

12/13/2022

FINAL

GitHub link: <https://github.com/MynameisKoi/CE450/tree/main/Final>

***Part I: Python Programming***



Source code: <https://github.com/MynameisKoi/CE450/blob/main/Final/1.py>

def trc1(*g*):

    def f(*args*):

        print('Calling', *g*.\_\_name\_\_, 'on argument', *args*)

        return *g*(*args*)

    return f

@trc1

def sqr(*x*):

    return *x*\**x*

@trc1

def sum\_sqr(*n*):

    return sum(sqr(x) for x in range(*n*+1))

print(sqr(3))

print(sum\_sqr(3))

Run program & result:

Text

Description automatically generated



Source code: <https://github.com/MynameisKoi/CE450/blob/main/Final/2.py>

def verify\_add(*m*, *ls*):

    # return true if addition of any two different elements in ls is m

    # return false otherwise

    for i in range(len(*ls*)):

        if *m* - *ls*[i] in *ls* and *m* - *ls*[i] != *ls*[i]:

            return True

    return False

print(verify\_add(100, [1,2,3,4,5]))

print(verify\_add (7, [1, 2, 3, 4, 2]))

print(verify\_add(10, [5,5]))

print(verify\_add(151, range(0, 200000, 3)))

print(verify\_add(50004, range(0, 200000, 4)))

Run program & result:

Text

Description automatically generated



Source code: <https://github.com/MynameisKoi/CE450/blob/main/Final/3.py>

def deep\_rvrs(*tup*):

    # reverse tuple with possible tuple elements

    # return reversed tuple

    if len(*tup*) == 0:

        return *tup*

    elif type(*tup*[0]) == tuple:

        return deep\_rvrs(*tup*[1:]) + (deep\_rvrs(*tup*[0]),)

    return deep\_rvrs(*tup*[1:]) + (*tup*[0],)

a = (11,12,13,14)

print(deep\_rvrs(a))

tpl = (11,(12,(13,113),14),15)

print(deep\_rvrs(tpl))

Run program & result:

Text

Description automatically generated



Source code: <https://github.com/MynameisKoi/CE450/blob/main/Final/4.py>

class Fibonacci():

    def \_\_init\_\_(*self*):

*self*.val = 0

*self*.prev = 0

*self*.trigger = False # set the trigger off at the beginning

    def nxt(*self*):

        if *self*.trigger == False:

            # reset the values if the trigger is off

*self*.val = 0

*self*.prev = 0

*self*.trigger = True

        if *self*.val == 0:

*self*.val = 1

        elif *self*.val == 1 and *self*.prev == 0:

*self*.val = 1

*self*.prev = 1

        else:

            # fibonacci

            temp = *self*.val

*self*.val = *self*.val + *self*.prev

*self*.prev = temp

        return *self*

    def \_\_repr\_\_(*self*):

*self*.trigger = False

        # set the trigger off again for the next execution

        return str(*self*.val)

a = Fibonacci()

print(a)

print(a.nxt())

print(a.nxt().nxt())

print(a.nxt().nxt().nxt())

print(a.nxt().nxt().nxt().nxt())

print(a.nxt().nxt().nxt().nxt().nxt())

print(a.nxt().nxt().nxt().nxt().nxt().nxt())

Run program & result:

Text

Description automatically generated



Source code: <https://github.com/MynameisKoi/CE450/blob/main/Final/5.py>

class Student():

    def \_\_init\_\_(*self*, *name* = '', *number* = 0):

*self*.name = *name*

*self*.course = *number*

    def \_\_add\_\_(*self*, *other*):

        res = Student()

        res.course = *self*.course + *other*.course

        return res

    def \_\_repr\_\_(*self*):

        return str(*self*.course)

    def \_\_lt\_\_(*self*, *other*):

        return *self*.course < *other*.course

    def \_\_gt\_\_(*self*, *other*):

        return *self*.course > *other*.course

    def \_\_eq\_\_(*self*, *other*):

        return *self*.course == *other*.course

    def \_\_ne\_\_(*self*, *other*):

        return *self*.course != *other*.course

a = Student('Peter', 3)

b = Student('Mike', 4)

c = Student('John', 5)

d = Student('Kelvin', 3)

print(a+b+d) # 10

print(a+b+c+d) # 15

print(a!=d) # False

print(b<c) # True

Run program & result:

Text

Description automatically generated

***Part I: Embedded System Design Theory***

1. Describe the application areas of the real-time operating system (RTOS)

The real-time operating system (RTOS) can be applied to different areas: vehicle control systems (automobiles, ships, railways, airplanes, etc.), satellite communications, or in real-time simulations, controlling robots.

1. Explain why the middleware is needed and where

The middleware is used to integrate different software components into other applications. Furthermore, it can connect disparate systems within an organization. Middleware is needed in legacy systems, cloud-based applications, mobile apps, etc. It acts like a bridge to connect gaps between applications, tools, and databases to provide unified services to users.

1. Describe each component’s function of any operating system

An operating system has many components:

* Process Management: monitor and handle various processes that the operating system is trying to execute at the same time.
* File Management: operate and responsible for the maintenance of files, organizing and managing data files.
* Network Management: administer and maintain computer networks, help develop routing and connection techniques that can be applied to connectivity and security
* Main Memory Management: a huge array of storage of bytes, responsible for reads or writes of specified memory addresses
* I/O Device Management: In computer I/O, the operating system is responsible for managing and controlling I/O processes and devices. Because the functions and speeds of devices attached to the computer (I/O devices) vary so considerably, multiple ways of controlling them are required.
* Security Management: Guarantee that the OS authorizes the use of operating files, memory, CPU, and hardware resources.
* Command Interpreter System: responsible for the interpretation and the execution of interactively entered or program-generated commands.

1. What general functions are there in any of the device drivers, including the description for each?

Device drivers provide a software interface to hardware devices. They are needed to permit a computer to interact with specific devices.

Some of the general functions of the device driver are:

* Accessing storage systems: allow users to store data and retrieve it when needed.
* I/O device: the device driver interacts with the computer’s OS to ensure its hardware functions as expected.
* Digital camera: allow communication between computers and cameras, transfer photos, and print photos
* Drivers for mobile OS: the device drivers can be used to help connect the mobile phones with computers.