10- class Birel:
    def __init__(self, name, Kind, Fly):
        self.name = name
        self.Kind = Kind
        self.Fly = Fly
class eagle(Birel):
    def __init__(self, Fly):
        self.Fly = Fly
class penguim(Birel):
    def __init__(self, run):
        self.run = run
e = eagle("eagle", "fly", "Birel").
e.eagle().
p = penguim("penguim", "run", "Birel").
p.penguim().

11-

4- class circle:
    def __init__(self, a, b):
        self.a=a
        self.b=b
    def area(math.pi.a.b)½
c=circle(2,4)
print(c.area).

5- class rectangle:
    def __init__(self, a, b).
    self.a=a
    self.b=b
    def area(a.b)
~~print~~ r=rectangle(2,6)
    print(r.area).

inheritance and polymorphism exercises.

6- class Animal:
    def __init__(self, name, sound/voice):
        self.name=name.
        self.sound=sound
    ~~def Dog(Animal):~~
class Dog(Animal):
    def __init__(self)
        return name+" sound.

```
else:
    messagebox.showwarning("warning", "you must
    enter the task.")
def remove_task(self):
    try
        select_index = self.task_listbox.curselection()[0]
    del self.task[selected_index]
        self.update_task_listbox()
    except IndexError:
    messagebox.showwaring("warning",
    "you mud selet a task to remove").
def update_task_listbox(self):
    self.task_listbox.delet(0, tk.END, task).
if __name__ = "__main__":
    root = tk.Tk().
    todo_App = ToDoApp(root).
    root.mainloop().
```

9- write a code use from callable().

```
global_vars=globals()
num_vars=len(global_vars)
print(f"there are {num_vars} variables in the globals()
                                    dictionary.")
```

10- certainly if you want to a python program that calculate the hasaddr() function, you can create a program like this.

```
class myclass:
    def __init__(self,x):
        self.x=x
obj=myclass(10)
has_attribute_x=hasaddr(obj,'x')
has_attribute_y=hasaddr(obj,'y')
print(f"the object has attribute 'x':
    {has_attribute_x}")
print(f"the object has attribute 'y':
    {has_attribute_y}")
```

(58) dir() → Trieu to Return Attributes of the object.

print(dir(strings)).

(59) divmod() → Return a Tuple of: quistion and Remainder.

print(divmod(5.2))

(60) enumerate() → Return an object

print(list(enumerate(['b', 'c', 'd'])))

(61) eval() → Run python code with program

x = 'print(50)'
eval(x).

(62) Compile() → is an object

x = compile('print(55)', 'test', 'eval').
exec(x).

تاریخ: / /

موضوع:

① Class account:
    def -_f(self):
        return 'A'
    def g(self):
        print(self._f()).
class D(account):
    def -_f(self):
        return 'B'
d = D()
d.g().

② Class a:
    def __f(self):
        return 'A'
    def g(self):
        print(self.__f()).
class D(a):
    def _a__f(self):
        return 'a'
d = D()
d.g()
w = a()
w.g().

تاريخ: / /

موضوع:

8- class employee:

def --init- (self, name, salary):

self.name = name

self.salary = salary.

class manager(employee):

def --init-- (self, departament):

self.departament):

e=employee ("Ahmad", "2000")

e. employee().

m= manager("Ali", "1200", "IT").

e. manager().

9- class vehicle:

def --init-- (self, drive):

self drive = drive.

class Truck(vehicle):

def --inind-- (self, wire, Kind):

self.wire = wire

self Kind = Kind

class bik (vehicle):

def --init-- (self, wir, fuse):

self. wire = wire

self.fuse = fuse.

v=vehicle(- -)

t= ---- (- - -)

b= ---- (- - -)

11 - write a python program that calculate the hash().

```
string_obj = "Hello"
int_obj = 123
Tuple_obj = (1, 2, 3)
hash_string = hash(string_obj)
hash_int = hash(int_obj)
hash_Tuple = hash(Tuple_obj)
print(f"the hash of the string '{string_obj}' is:
      {hash_string}").
print(f"the hash of the integer '{int_obj}' is:
      {hash_string}")
print(f"the hash of the Tuple '{Tuple_obj}' is:
      {hash_Tuple}")
```

12 - write the program that calculate the help().

```
object_or_module = input("Enter the name of the
                   object or module want help with:")
help_info = help(object_or_module)
print(help_info)
```

(27) my-list = [1, 4, 5, 9]
print(len(my-list)) → 4

(28) A = "Hello!"
listI = list(A)
print(list) → ['H', 'e', 'l', 'o']

(29) def my-function():
z = 1
y = 2
print(z)
my-function().

(30) list1 = [10, 20, 30]
result = map(lambda x: x * 2, list1)
print(list(result)) → [20, 40, 60]

(31) A = bytearray(b'ABC')
m = memoryview(A).
print(m[0]) → A = 65
print(m[2]) → B = 66

(11) list = iter([1,2,3,])
print(next(list+1) → 2

(12) n1 = 35.6667
print(round(n1)) → 36

(13) n2 = 'hello'
print(set(n2)) → hello

(14) class person:
name = 'John'
age = 36
setattr(person, 'age', 40)

(15) a1 = ('a', 'b', 'c', 'd')
x = slice(3, 4)
print(a1[-7])

(16) a1 = (1, 11, 5)
x = sorted(a1)
print(x) → (1, 5, 11)

تاریخ: / /

موضوع:

```
class Cat(Animal):
    def __init__(self):
        return name "+" sound.
D = Dog("Dog", "Bark")
D.Dog()
C = Cat("Cat", "Meow")
C.Cat()
```

7- import math

```
class Rectangle:
    def __init__(self, taal, arz):
        self.taal = taal
        self.arz = arz
    def masahat(self):
        return self.taal * self.arz
class circle:
    def __init__(self, shoa):
        self.shoa = shoa
    def masahat(self):
        return (self.shoa ** 2) * math.pi
class Triagle:
    def __init__(self, a, b)
        self.a = a
        self.b = b
    def masahat(self):
        return (a * b) / 2
```

5-

6

تاریخ: / /

موضوع:

(19) class campani:

def __init__(self, c, f, i):

self.i = i

self.f = f

self.c = c

def __str__(self):

list = []

list.append(f'{self.c} - {self.f} - {self.i}')

return ' '.join(list)

a = campani('Kabul', 'Irafet', 123)

print(a)

(20) class Animal:

def __init__(self, name, specise):

self.name = name

self.specise = []

def add_anime(self, anime)

self.anime = anime

print(f"{anime} has added")

else:

print("Invalid anime object")

تاريخ: / / موضوع:

(25) Class Report:

```
def __init__(self, file_path):
    self.file_path = file_path
def generate_report(self):
    try:
        with open(self.file_path, 'r') as file:
            data = file.read()
            report = f"Report generated from
                {self.file_path}: {data}"
            return report
    except FileNotFoundError:
        return f"File '{self.file_path}' not found"
    except PermissionError:
        return f"permission denied to read file
            '{self.file_path}'"
    except Exception as e:
        return f"file '{self.file_path}' not found
            {self.file_path}': {e}"
```

~~for~~

```
report_generator = Report('data.txt')
print(report_generator.generate_report())
```

تاریخ: / /

موضوع:

(23)

```
class config:
    def __init__(self, filename):
        self.filename = filename
        self.setting = {}
    def load_setting(self):
        with open(self.filename, 'r') as file:
            for line in file:
                key, value = line.strip().split('=')
                self.setting[key.strip()] = value.strip()
    def get_setting(self, key):
        return self.settings.get(key, None)
config = config('setting.config')
Config.load_seittings()
print(config.get_setting("username"))
```

(13) def func(l)
y = l+2
l = [1, 2, 3, 4]
f = map(func, l)
print(list(f)) → Run [8, 7, 5, 6]

(14) Max()
print(max(1, 2, 3, 4, 7) → Run → 7

(15) - open
open('text.txt', 'r')

(16) print()
print('Hello World')

(17) pow()
x = pow(2, 3)
print(x) → 8

(18) print(ord('a')) → 0013

(19) class A(object)
def __init__(self, name)
self.name = name.
x = A('Ali')
→ print(x.name) → Ali

47 class calculator:
    def total(num1, num2):
        return num1+num2
calculator.total = staticmethod(calculator.total)
sum = calculator.total(5,7)
print('sum', sum).

48 n3 = 16
print(str(n3))

49 number = [1,2,3,4,5]
sum = sum(number).
print(sum, number).
n6 = 'Zamon'

50 print(type(n6))

51 print(vars(int)).

52 lan = ['Reza', 'Ameer']
versions = [14,13]
result = zip(lan, version)
print(list(result))

(22) memory view()

x = 'Hellow world'

print(memory.view(x)) → Run

(23) local(6())

def fume():

x = 3

y = 7

v = local.(v)

print(v.x) → Run=3

(24) - num = 5

print(is instance (num, int)) → True

(25)

class Animal:

pass

class Dog(Animal):

pass

print (is instance (Dog, Animal)) → True

(26)

my_l = [1, 2, 3]

my_l = iter [my_list]

print (next (my_list)) → 1

OOP (two question)

1- class person:

```
    def __init__(self, name, lname, age):
        self.name = name
        self.lname = lname
        self.age = age
p = person('Ali', 'Ahmadi', 22)
print(p.name), lname, age.
```

2-

```
class greeting:
    def __init__(self, greeting):
        self.greeting = greeting
g = greeting('Hi how are you do want to eam with me')
print(g.greeting).
```

3-

```
class car:
    def __init__(self, model, year):
        self.model = model
        self.year = year
c = car("Toyota", "2013")
print(c.model).
print(c.year).
```

(53) np = __import__('numpy', globals(), locals(), [], 0)
a = np.array([1, 2, 3])
print(type(a))

(54) x = compile('print(55)', 'test', 'eval')
exec(x)

(55) complex() → complex number
x = complex(3, 5)

(56) delattr() is a Attribute from the object.

class person:
    name = 'John'
    age = 36
delattr(person, 'age')

(57) dict() → create a dictionary.

d = dict(('a': 2, 'b': 3, 'c': 4))

print(d)

(26) class Ticket:

def __init__(self, movie_name, seat_number, price):

self.movie_name = movie_name

self.seat_number = seat_number

self.price = price

def display_ticket_details(self):

print(f"movie Name: {self.movie_name}")

print(f"seat Number: {self.seat_number}")

print(f"price: {self.price}")

def apply_discount(self, discount_percentage):

self.price = self.price * (1 -

discount_percentage/100).

(27)

ticket = Ticket("Avengers: Englishi", "A pro", 55.99).

ticket.display_ticket_details().

ticket.apply_discount(10).

4- user_input = input("Enter a string:")
ascii_representation = ascii(user_input)
print("ascii represent:", ascii_representation)

5- num = int(input("Enter the integer:"))
binarry = bin(num).
print("the binarry is:", binarry)

6- value = input("Enter a value:")
usingbool = bool(value)
print("Boolean value:", boolean_value).

7- numbers = [65, 66, 67, 68]
byte_array = byte array(numbers).
print("Byte array is:", byte_array).

8- number = [65, 66, 67, 68]
bytes_obj = bytes(numberes)
print("Bytes is:", bytes_obj)
we make a list of integere and use from buit-in code for calculator

1- python abs()

```
num = float(input("Enter the number"))
print(f"the abso-value of {num is {absolute of}")
```

2- my-list = [True, True, False, True]

```
# List are True
result = all(my_list)
    if result:
    print ("All element in the list are True")
  else:
      print("At least one element in the list False")
```

3- my_list = [1, 2, 3, 4]

```
result = any(my_list)
    if result:
    print("At least one element in list]
    else:
      print("All element in the list is not]
```

(16) class team:
def __init__(self, Teacher, student):
self.Teacher = Teacher
self.student = student
class Football(school):
def __init__(self, teacher, studen, *name, *name):
super().__init__(teacher, steadent)
self.football = None
self.name = name
self.__name = __name
def remove(self):
pass
class remove(library):
def __init__(self, book, price, name, age):
super().__init__(booke, price)
self.name = name
d.remove(name, 'name')
d.name().

تاریخ: / /

موضوع:

(16) class libraray:
def --init-- (self, bookk, price):
self.bookk = book
self.price = price
def --add-- (self, other):
list = [
list.append('math')
def --repr-- (self):
def remove (bookk):
remove('book')
d = libraray ('math', '20 Afghani')
print (d.price)
print (b.bookk)

(17) class schools
def --init-- (self, teacher, student, fname, lname):
super().--init-- (Teacher, student)
self.neme = name
self.lneme = lname
class removee (library):
def --init-- (self, book, price, name, age):
supper().--init-- (book, price)
self.neme = name
self.age = age

```
def remove_anime(self,
    for a in self.anime
        if a.name == anime
            self.anime.remove
            print(f"{anime}")
dog = Animal('cow', 'four', 200)
dog.walk()
```

(12)
```
import datetime
class log:
    def __init__(self, filename):
        self.filename = filename
    def write_error(self, message):
        with open(self.filename, 'a') as file:
            timestamp = datetime.datetime.now().strftime
                ("%Y-%m-%d %H:%M:%S")
log = log('error.log')
log.write_error("something is wrong in here")
```

(28) class Restaurant:

def __init__(self, name):

self.name = name

self.menu = ~~menu~~ []

def add_item(self, item):

self.menu.append(item)

def remove_item(self, item_name):

for i in self.menu:

if i.name = item_name

self.menu.remove(i)

return

restaurant = Restaurant("my Restaurant")

item1 = Item("pizza", 10, 99)

item2 = Item("Burger", 8, 88).

restaurant.add_item(item1).

restaurant.add_item(item2)

(40)

```
import tkinter as tk
from tkinter import ttk
import requests
class WeatherApp:
    def __init__(self, master):
        master.title("Weather App")
        self.api_key = "your_Api_Key"
        city_label = tk.Label(master, text="Enter city:")
        city_label.pack(pady=10)
        self.city.pack(padx=10)
        self.city_entry = tk.Entry(master)
        self.city_entry.pack()
text="Fetch weather", command=self.fetch_weather)
        fetch_button.pack(pady=10)
        self.weather_info_label =
tk.Label(master, text="", font=("Century", 16))
        self.weather_info_label.pack()
        :
        :
root = tk.Tk()
app = WeatherApp(root)
root.mainloop()
```

(13) class laptop:

def __init__(self, model, size):

self.model = model

self.size = size.

def f(self):

print(self.size).

printl(self.model

~~class f(self):~~

Class properti(laptop):

def __init__(self, model, size)

super().__init__(model, size)

self.__size = None.

def h(self):

print(self.__size)

d = properti('del', '16 inch')

d.h().

d.f().

(29)

```
import tkinter as tk
from tkinter import ttk
class LoginApp:
    def __init__(self, master):
        self.master = master
        master.title("Login Form")
        username_label = tk.Label(master, text="username:")
        username_label.pack(pady=10)
        self.username_entry = tk.Entry(master)
        self.username_entry.pack()
        password_label = tk.Label(master, text="password:")
        password_label.pack(pady=10)
    def login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        if username == "user" and password == "password":
            tk.messagebox.showinfo("login successful")
            self.master.destroy()
        else:
            tk.messagebox.showerror("login Error",
            "Invalid username or password")
root = tk.Tk()
app = LoginApp(root)
root.mainloop()
```

تاریخ / /

(38)

```
Class calculatorApp:
        def __init__(self, master):
        master.title("simple calculator")
        self.display = tk.Entry(master, width=25,
        borderwidth=5, state='readonly')
        self.display.pack(pady=10)

# Buttons
button_frame = tk.Frame(master)
button_frame.pack()
buttons = ['7', '8', '9', '/',
        '4', '5', '6', '*',
        '1', '2', '3', '-',
        '0', '.', '=', '+',]
        row=0
        col=0
        for button_text in buttons:
        button = ttk.Button(button_frame, text=button_text,
        width=5, command=lambda text=button_text:
        self.button_click(text))
        button.grid(row=row, column=col,
        padx=5, pady=5)
        col += 1
        if col>3:
        col = 0
        row += 1
```

(29)
```
class Flight:
    def __init__(self, flight_number, dest):
        self.flight_n = flight_n
        self.des = des
        self.passenger = passenger
    def add_passenger(self, passenger)
        self.passenger.append(passenger):
    def remove_passenger(self, passenger):
        if passenger in self.passenger:
            self.passenger.remove(passenger)
        else:
            print(f"{passenger} is not a pa-")
flight = Flight("ABC123", "new york")
person = person("john")
person2 = person("johne")
flight.add_passenger(person)
flight.add_passenger(person2)
flight.remove_passenger(person)
```

clear_button = tk.Button(button_frame, text="C", width=5,

command = self.clear_display)
clear_button.grid(row=4, columnspan=2, padx=5, pady=5)
#equal_button.
equals_button = tk.Button(button_frame, text="=",
width=5, command=self.calculate)
clear_button.grid(row=4, column=0, columnspan=2,
padx=5, pady=5).

self.current_expression = ""

def button_click(self, text):
if text == "=":
self.calculate()
else:

(14)
```
class student:
    def __init__(self, grade, firstp):
        self.grade = grade
        self.firstp = firstp
def f(self):
    print(self.firstp)
    print(self.grade)
class properti(student):
    def __init__(self, grade, firstp):
    super().__init__(grade, firstp)
        self.size = None
    def h(self):
        print(self.__size)
d = properti('12th at the school', 'first position the class')
d.h()
d.f()
```