

B1 - SET - B

PART A(10 X 01=10)

1) Select the correct syntax to draw line and arc in canvas component.

- a) `arc1 = C.create_arc(coord, start = 10, extent = 160, fill = "red")`
`line1 = C.create_line(10,10,200,200,fill = 'white')`
- b) `arc1 = C.create(coord, start = 0, extent = 150, fill = "red")`
`line1 = C.create(20,20,200,200,fill = 'white') (ans)`
- c) `arc1 = C.create_arc(coord, fill = "red")`
`line1 = C.create_line(20,20,fill = 'white')`
- d) `arc1 = C.create_arc(start = 0, extent = 150, fill = "red")`
`line1 = C.create_line(200,200,fill = 'white')`

2) Which of the following property of Geometry manager pack allows the widget to fill any space not otherwise used in widget's parent?

- a. `fill`
- b. `span`
- c. **`expand`**
- d. `pad`

3) `execute_query(a,b)`, which one is the correct representation of a and b

- a. a is the connection object, b is the query string**
- b. a will return the tuples, b is the connection object
- c. a is the query string, b is the object for mysql
- d. a is the mysql object, b is the insert query string

4) If we want to store the .jpg image in database, what type of field is needed?

- a) VARCHAR
- b) TEXT
- c) **BLOB (ans)**
- d) CHAR

5) What are the required parameters to connect any database in python?

- a) Names of the database, host and number of the used port
- b) Password, username, port number and the database name
- c) Password, username, socket number, database name
- d) **Username, Password, names of the host and database (ans)**

6) The below code prints ____ numbers(s) of starts

```
i=7
while i>0:
    i-=3+1
    print('*')
    if i<=2:
        break
else:
    print('*')
```

- a) 2 (ans)**
- b) 3
- c) 4
- d) 5

7) What does the Thread.join() method do?

- (a) Adds the thread to a pool**
- (b) Merges two threads into one
- (c) Waits for the thread to finish
- (d) Restricted access to a resource

8) Which method returns the number of thread object in the caller's thread control?

- a) Threading.activeCount()
- b) Threading.currentThread() (ans)**
- c) Threading.enumerate()
- d) Threading.thread()

9) Function Programming Paradigms which primarily focus on

- a) expressions and stateless
- b) stateful and expression
- c) statements and expressions (ans)**
- d) stateful and statements

10) The following code is an example for

```
small = []  
for i in range(20):  
    if i<5:  
        small.append(i)
```

- a) Imperative programming (ans)**
- b) Declarative programming
- c) Both Imperative and declarative
- d) Iterative programming

11)

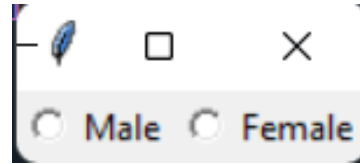
a)

Geometry manager – pack()

```

from tkinter import *
r=Tk()
var1=IntVar()
var2=IntVar()
r1=Radiobutton(r,text="Male",variable=var1,value=1)
r1.pack(side=LEFT)
r2=Radiobutton(r,text="Female",variable=var2,value=1)
r2.pack(side=LEFT)
r.mainloop()

```



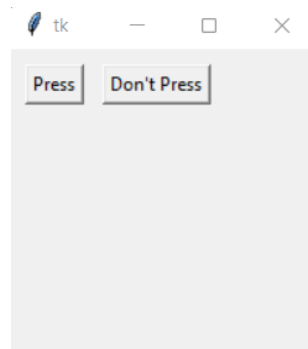
b)

Geometry manager -place()

```

from tkinter import *
r=Tk()
button1=Button(r,text="Press")
button1.place(x=10,y=10)
button2=Button(r,text="Don't Press")
button2.place(x=60,y=10)
r.mainloop()

```



12)

Event handlers is a type of function or method that run a specific action when a specific event is triggered. Making the user interface more interactive.

Code snippet:-

```

from tkinter import *
r=Tk()
def pressed():
    label1.configure(text="Welcome")
label1=Label(r,text="")
label1.pack()
button1=Button(r,text="Press",command=pressed)
button1.pack()
button2=Button(r,text="Don't Press",command=r.destroy)
button2.pack()
r.mainloop()

```



13)

```
#suppose two trending topics are topic1 topic2 and bunch of tweets are tweets=[]
import functools
def fun1(x):
    if x.topic==topic1:
        return True
    else:
        return False
def fun2(x):
    if x.topic==topic2:
        return True
    else:
        return False
t1_count=0
t2_count=0
def func1(x):
    global t1_count
    t1_count+=t1_count
def func2(x):
    global t2_count
    t2_count+=t2_count
tweet1=filter(fun1,tweets)
tweet2=filter(fun2,tweets)
map(func1,tweet1)
functools.reduce(func2,tweet2)
print(t1_count)
print(t2_count)
```

14)

Imperative Programming	Declarative Programming
In this, programs specify how it is to be done.	In this, programs specify what is to be done.
It simply describes the control flow of computation.	It simply expresses the logic of computation.
Its main goal is to describe how to get it or accomplish it.	Its main goal is to describe the desired result without direct dictation on how to get it.
Its advantages include ease to learn and read, the notional model is simple to understand, etc.	Its advantages include effective code, which can be applied by using ways, easy extension, high level of abstraction, etc.
Its type includes procedural programming, object-oriented programming, parallel processing approach.	Its type includes logic programming and functional programming.

#Declarative example :-

```
small_nums = [x for x in range(20) if x<5]
print(small_nums)
```

```
[0, 1, 2, 3, 4]
```

#Imperative example:-

```
small_nums=[]
for i in range(20):
    if i<5:
        small_nums.append(i)
print(small_nums)
```

```
[0, 1, 2, 3, 4]
```

15)

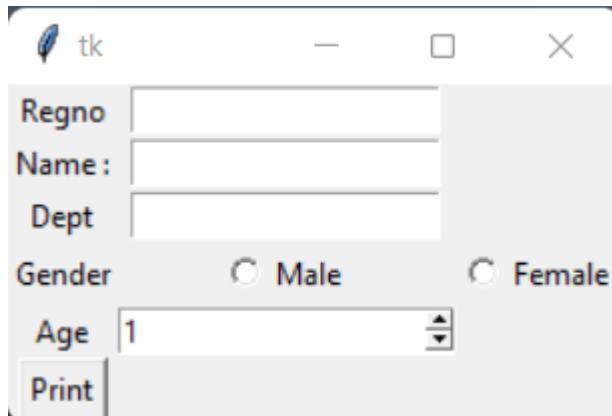
Closure is a function object that remembers values in enclosing even if they are not present in memory

```
def product(a,b):
    a=a
    b=b
    def multiply():
        print("Product is : ",a*b)
    return multiply
mul=product(10,5)
print(mul)
mul()
```

```
<function product.<locals>.multiply at 0x000001A4653848B0>
Product is : 50
```

16)

```
from tkinter import *
r=Tk()
var=IntVar()
label1=Label(r, text="Regno")
label1.grid(row=0,column=0)
entry1=Entry(r)
entry1.grid(row=0,column=1)
label2=Label(r, text="Name :")
label2.grid(row=1,column=0)
entry2=Entry(r)
entry2.grid(row=1,column=1)
label3=Label(r, text="Dept")
label3.grid(row=2,column=0)
entry3=Entry(r)
entry3.grid(row=2,column=1)
label4=Label(r, text="Gender")
label4.grid(row=3,column=0)
R1 = Radiobutton(r, text="Male", variable=var, value=1)
R1.grid(row=3,column=1)
R2 = Radiobutton(r, text="Female", variable=var, value=2)
R2.grid(row=3,column=2)
label5=Label(r, text="Age")
```

A screenshot of a Tkinter window titled "tk". The window contains a form with the following elements: a label "Regno" followed by an empty text entry field; a label "Name :" followed by an empty text entry field; a label "Dept" followed by an empty text entry field; a label "Gender" followed by two radio buttons labeled "Male" and "Female"; a label "Age" followed by a spin box showing the value "1"; and a "Print" button at the bottom left.

```

label5.grid(row=4,column=0)
s1 = Spinbox(r, from_=1, to=200)
s1.grid(row=4,column=1)
button1 = Button(r, text='Print')
button1.grid(row=5,column=0)
r.mainloop()

```

17)

```

import sqlite3
conn=sqlite3.connect('APP.db')
print("Opened database successfully")
conn.execute("CREATE TABLE product(product_id INT PRIMARY KEY NOT NULL,product_name VARCHAR,category
text,price INT,manufacturer text,quantity INT);")
print("Table created successfully")
conn.execute("INSERT INTO product VALUES(101,'Television','Electronics',30000,'ABC',10);")
conn.execute("INSERT INTO product VALUES(102,'Chair','Furniture',3000,'XYZ','56');")
conn.execute("INSERT INTO product VALUES(103,'Bulb','Electronics',50,'ABC',30);")
conn.execute("INSERT INTO product VALUES(104,'Table','Furniture',100,'XYZ',20);")
conn.commit()
print("Records inserted successfully");
print("\nDisplaying the table \n")
for rows in conn.execute("SELECT * from product;"):
    print(rows)
print("\nPART A\n")
for rows in conn.execute("SELECT product_name,price from product ORDER BY price DESC LIMIT 3;"):
    print(rows)
print("\nPART B\n")
for rows in conn.execute("SELECT * from product WHERE manufacturer='ABC' AND price<200;"):
    print(rows)
print("\nPART C\n")
for rows in conn.execute("SELECT category from product GROUP BY category;"):
    print(rows)
print("\nPART D\n")
for rows in conn.execute("SELECT product_name from product WHERE manufacturer LIKE 'A%' OR manufacturer LIKE
'C%';"):
    print(rows)
print("\nPART E\n")
for rows in conn.execute("SELECT category,SUM(quantity) from product GROUP BY category;"):
    print(rows)
conn.close()

```

```
Opened database successfully
Table created successfully
Records inserted successfully
```

Displaying the table

```
(101, 'Television', 'Electronics', 30000, 'ABC', 10)
(102, 'Chair', 'Furniture', 3000, 'XYZ', 56)
(103, 'Bulb', 'Electronics', 50, 'ABC', 30)
(104, 'Table', 'Furniture', 100, 'XYZ', 20)
```

PART A

```
('Television', 30000)
('Chair', 3000)
('Table', 100)
```

PART B

```
(103, 'Bulb', 'Electronics', 50, 'ABC', 30)
```

PART C

```
('Electronics',)
('Furniture',)
```

PART D

```
('Television',)
('Bulb',)
```

PART E

```
('Electronics', 40)
('Furniture', 76)
```

18)

```
import threading
def printodd(ls):
    print("Inside Printodd")
    for i in ls:
        if i%2!=0:
            print(i)
def panildrome():
    print("Inside Panildrome")
    for i in range(1,201):
        j=str(i)
        if j==j[::-1]:
            print(i)
ls=[2,4,9,1,3,4]
thread1=threading.Thread(target=printodd,args=(ls,))
thread2=threading.Thread(target=panildrome)
thread1.start()
thread2.start()
```

```
thread1.join()
thread2.join()
print("Done")
```

```
Inside Printodd
9
1
3
Inside Panildrome
1
2
3
4
5
6
7
8
9
11
22
33
44
55
66
77
88
99
101
111
121
131
141
151
161
171
181
191
Done
```

19)

a)

A function is called pure function if it always returns the same result for same argument values and it has no side effects like modifying an argument (or global variable) or outputting something

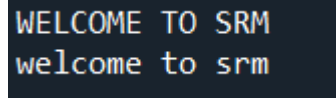
Examples of pure functions are `strlen()`, `pow()`, `sqrt()` etc.

b)

Since functions are objects we can pass them as arguments to other functions. Functions that can accept other functions as arguments are also called higher-order functions.

Example-

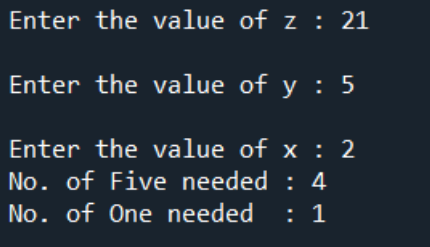
```
def shout(text):
    return text.upper()
def whisper(text):
    return text.lower()
def greet(func):
    greeting=func("Welcome to srm")
    print(greeting)
greet(shout)
greet(whisper)
```



```
WELCOME TO SRM
welcome to srm
```

c)

```
def make_amount(rupees_to_make,no_of_five,no_of_one):
    five_needed=0
    one_needed=0
    five_needed = rupees_to_make // 5
    one_needed = rupees_to_make-(five_needed*5)
    if five_needed <= no_of_five and one_needed <= no_of_one:
        print("No. of Five needed :", five_needed)
        print("No. of One needed :", one_needed)
    elif ((no_of_five - five_needed)*5) <= (no_of_one - one_needed):
        one_needed = one_needed + ((five_needed - no_of_five)*5)
        five_needed = no_of_five
    if five_needed <= no_of_five and one_needed <= no_of_one:
        print("No. of Five needed :", five_needed)
        print("No. of One needed :", one_needed)
    else:
        print(-1)
    else:
        print(-1)
z=int(input("Enter the value of z : "))
y=int(input("Enter the value of y : "))
x=int(input("Enter the value of x : "))
make_amount(z,y,x)
```



```
Enter the value of z : 21
Enter the value of y : 5
Enter the value of x : 2
No. of Five needed : 4
No. of One needed : 1
```