Bangalore Institute of Technology

Department of Mathematics

Practise Question

<table>
<tr><th>Q. No.</th><th>Multiple integral</th></tr>
<tr><td>a</td><td>Python program to find the area $A = 4\int_0^a \int_0^{\frac{b\sqrt{a^2-x^2}}{a}} dy\, dx$ of the ellipse.
<pre><code>from sympy import *
a,b=symbols("a,b",real=True,positive=True)
var("x,y")
print("Area,A=")
display(4*Integral (1,(y,0,b*sqrt(a**2-x**2)/a),(x,0,a)))
A = 4*integrate (1,(y,0,b*sqrt(a**2-x**2)/a),(x,0,a))
print ("=",A)
</code></pre></td></tr>
<tr><td>b</td><td>Python program to find $\beta(3,5)$ and $\Gamma(5)$
<pre><code>from sympy import *
print("β(3,5)=%.3f"%beta(3.0,5.0))
print("Γ(5)=",gamma(5))
</code></pre></td></tr>
<tr><td></td><td></td></tr>
<tr><td>a</td><td>Python program to evaluate the integral $\int_0^1 \int_0^x (x^2+y^2)\, dy\, dx$
<pre><code>from sympy import *
var("x,y")
display(Integral (x**2+y**2,(y,0,x),(x,0,1)))
I = integrate (x**2+y**2,(y,0,x),(x,0,1))
print ("=",I)
</code></pre></td></tr>
<tr><td>b</td><td>Python program to find $\beta\left(5/2, 7/2\right)$
<pre><code>from sympy import *
print("β(5/2,7/2)=%.3f"%beta(5/2,7/2))
</code></pre></td></tr>
<tr><td></td><td></td></tr>
<tr><td>a</td><td>Python program to find the area of the curve $x^2+y^2=16$ in the positive quadrant.
<pre><code>from sympy import *
var("x,y")
print("Area,A=")
display(Integral (1,(y,0,sqrt(16-x**2)),(x,0,4)))
A = integrate (1,(y,0,sqrt(16-x**2)),(x,0,4))
print ("=",A)
</code></pre></td></tr>
<tr><td>b</td><td>Python program to find the value of $\beta\left(5/2, 9/2\right)$
<pre><code>from sympy import *
print("β(5/2,9/2)=%.3f"%beta(5/2,9/2))
</code></pre></td></tr>
</table>

a

Python program to verify relation between *Beta* and *Gamma* function for m=5 and n=7.

```
from sympy import *
m,n=5.0,7.0
Γm=gamma(m)
Γn=gamma(n)
Γmn=gamma(m+n)
βmn=beta(m,n)
RHS=(Γm*Γn)/Γmn
print("β(m,n)=%0.6f"%βmn,"RHS=%0.6f"%RHS)
if(βmn==RHS):
    print("Beta and Gamma relation verfied")
```

b

Python program to find the value of $\Gamma(13)$

```
from sympy import *
print("Γ(13)=",gamma(13))
```

a

Python program to find the volume of the tetrahedron bounded by the planes x=0, y=0 and z=0, $\frac{x}{a}+\frac{y}{b}+\frac{z}{c}=1$

```
from sympy import *
a,b,c=symbols("a,b,c")
var("x,y,z")
print("volume, V=")
display(Integral (1,(z,0,c*(1-x/a-y/b)),(y,0,b*(1-x/a)),(x,0,a)))
V = integrate (1,(z,0,c*(1-x/a-y/b)),(y,0,b*(1-x/a)),(x,0,a))
print ("=")
display(V)
```

b

Python program to evaluate $\Gamma(5)$ by using definition

```
from sympy import *
print("Γ(5)=",gamma(5))
```

a

Python program to evaluate the integral $\int_0^3\int_0^{3-x}\int_0^{3-x-y}(x\,y\,z)dz\,dy\,dx$

```
from sympy import *
var("x,y,z")
display(Integral (x*y*z,(z,0,3-x-y),(y,0,3-x),(x,0,3)))
I = integrate (x*y*z,(z,0,3-x-y),(y,0,3-x),(x,0,3))
print ("=",I)
```

b

Python program to calculate $\beta\left({}^{3}/{}_{2},\ {}^{5}/{}_{2}\right)$ and $\Gamma\left({}^{7}/{}_{2}\right)$

```
from sympy import *
print("β(3/2,5/2)=%.3f"%beta(3/2,5/2))
print("Γ(7/2)=%0.6f"%gamma(7/2))
```

<table>
<tr><td>a</td><td>Python program to find the volume of the tetrahedron bounded by the planes x=0, y=0 and z=0, $\frac{x}{2}+\frac{y}{3}+\frac{z}{4}=1$

```
from sympy import *
var("x,y,z")
print("volume, V=")
display(Integral (1,(z,0,4*(1-x/2-y/3)),(y,0,3*(1-x/2)),(x,0,2)))
V = integrate (1,(z,0,4*(1-x/2-y/3)),(y,0,3*(1-x/2)),(x,0,2))
print ("=")
display(V)
```
</td></tr>
<tr><td>b</td><td>Python program to evaluate Γ(21)

```
from sympy import *
print("Γ(21)=%.2f"%gamma(21))
```
</td></tr>
<tr><td></td><td></td></tr>
<tr><td>a</td><td>Python program to evaluate the triple integral $\int_{-c}^{c}\int_{-b}^{b}\int_{-a}^{a}(x^2+y^2+z^2)dx\,dy\,dz$

```
from sympy import *
a,b,c=symbols("a,b,c",real=True,positive=True)
var("x,y,z")
display(Integral (x**2+y**2+z**2,(x,-a,a),(y,-b,b),(z,-c,c)))
I = integrate (x**2+y**2+z**2,(x,-a,a),(y,-b,b),(z,-c,c))
print ("=")
display(I)
```
</td></tr>
<tr><td>b</td><td>Python program to calculate $\beta\left({}^{5}/_{2},\ {}^{7}/_{2}\right)$ and $\Gamma\left({}^{5}/_{2}\right)$

```
from sympy import *
print("β(5/2,7/2)=%.3f"%beta(5/2,7/2))
print("Γ(5/2)=%0.6f"%gamma(5/2))
```
</td></tr>
<tr><td></td><td></td></tr>
<tr><td>a</td><td>Python program to evaluate the double integral $\int_{1}^{4}\int_{0}^{\sqrt{4-x}}(x\,y)\,dy\,dx$

```
from sympy import *
var("x,y")
display(Integral (x*y,(y,0,sqrt(4-x)),(x,1,4)))
I = integrate (x*y,(y,0,sqrt(4-x)),(x,1,4))
print ("=",I)
```
</td></tr>
<tr><td>b</td><td>Python program to find the area of the Cardioid $r=a(1+cos\theta)$ in polar form

```
from sympy import *
a=symbols("a",real=True,positive=true)
var("r,θ")
print("Area,A=")
display(2*Integral (r,(r,0,a*(1+cos(θ))),(θ,0,pi)))
A = 2*integrate (r,(r,0,a*(1+cos(θ))),(θ,0,pi))
print ("=")
display(A)
```
</td></tr>
<tr><td></td><td></td></tr>
<tr><td>a</td><td>Python program to evaluate $\int_{0}^{log2}\int_{0}^{x}\int_{0}^{x+logy}(e^{x+y+z})dz\,dy\,dx$</td></tr>
</table>

```python
from sympy import *
var("x,y,z")
display(Integral (exp(x+y+z),(z,0,x+log(y)),(y,0,x),(x,0,log(2))))
I = integrate (exp(x+y+z),(z,0,x+log(y)),(y,0,x),(x,0,log(2)))
print ("=")
display(I)
```

b

Python program to find $\beta(3,5)$ and $\Gamma(10)$

```python
from sympy import *
print("β(3,5)=%.3f"%beta(3.0,5.0))
print("Γ(10)=%0.6f"%gamma(10))
```

Vector Space

1

Python program to verify the *rank-nullity* theorem for the linear transformation $T:R^3 \rightarrow R^3$ defined by $T(x,y,z)=(x+4y+7z,2x+5y+8z,3x+6y+9z)$

```python
from sympy import *
A = Matrix([[1,2,3],[4,5,6],[7,8,9]])
r = A.rank()
print('Rank of the linear transformation : r = ',r)
NullSpace =A.nullspace()
print('Null space of the linear transformation :\n ')
NullSpace = Matrix(NullSpace)
pprint(NullSpace)
n =NullSpace.shape[1]
#print('Nullity of the linear transformation : n = ',n)
dim =int(input('Enter the dimension of the vector space U :'))
if dim==r+n:
    print('Rank and Nullity theorem holds good \ndim(u)= dim(R(T))+dim(N(T))
else:
    print('dim(u)!= dim(R(T))+dim(N(T))')
```

2.

Python program to verify the *rank-nullity* theorem for the linear transformation $T:R^3 \rightarrow R^3$ defined by $T(x,y,z)=(x+y\ ,x-y,2x-z)$

```python
from sympy import *
A = Matrix([[1,1,2],[1,-1,0],[0,0,-1]])
r = A.rank()
print('Rank of the linear transformation : r = ',r)
NullSpace =A.nullspace()
print('Null space of the linear transformation :\n ')
NullSpace = Matrix(NullSpace)
pprint(NullSpace)
n =NullSpace.shape[1]
#print('Nullity of the linear transformation : n = ',n)
dim =int(input('Enter the dimension of the vector space U :'))
if dim==r+n:
    print('Rank and Nullity theorem holds good \ndim(u)= dim(R(T))+dim(N(T)
else:
    print('dim(u)!= dim(R(T))+dim(N(T))')
```

3

Python program to verify the *rank-nullity* theorem for the linear transformation $T: R^3 \to R^3$ defined by $T(x,y,z) = (x+y\ ,y+z,\ z+x)$

```
from sympy import *
A = Matrix([[1,0,1],[1,1,0],[0,1,1]])
r = A.rank()
print('Rank of the linear transformation : r = ',r)
NullSpace =A.nullspace()
print('Null space of the linear transformation :\n ')
NullSpace = Matrix(NullSpace)
pprint(NullSpace)
n =NullSpace.shape[1]
#print('Nullity of the linear transformation : n = ',n)
dim =int(input('Enter the dimension of the vector space U :'))
if dim==r+n:
    print('Rank and Nullity theorem holds good \ndim(u)= dim(R(T))+dim(N(T))
else:
    print('dim(u)!= dim(R(T))+dim(N(T))')
```

4

Python program to verify the *rank-nullity* theorem for the linear transformation $T: R^3 \to R^3$ defined by $T(x,y,z) = (x+y+z\ ,2x+3z,\ x+2y+4z)$

```
from sympy import *
A = Matrix([[1,2,1],[1,0,2],[1,3,4]])
r = A.rank()
print('Rank of the linear transformation : r = ',r)
NullSpace =A.nullspace()
print('Null space of the linear transformation :\n ')
NullSpace = Matrix(NullSpace)
pprint(NullSpace)
n =NullSpace.shape[1]
#print('Nullity of the linear transformation : n = ',n)
dim =int(input('Enter the dimension of the vector space U :'))
if dim==r+n:
    print('Rank and Nullity theorem holds good \ndim(u)= dim(R(T))+dim(N(T))
else:
    print('dim(u)!= dim(R(T))+dim(N(T))')
```

5

Python program to find the image of the vector $(5,0)$ when it is rotated by 90^0 then stretched horizontally.

```
from pylab import *
x, y = 5, 0
X,Y = -y,x
X1,Y1 = X-Y,Y    #X1=X-a*Y with a=1
arrow(0,0,x,y,head_width =0.2,head_length=0.2,ec='r')
arrow(0,0,X,Y,head_width=0.2,head_length=0.2,ec='g')
arrow(0,0,X1,Y1,head_width=0.2,head_length=0.2,ec='b')
ylim(-2,6)
xlim(-6,6)
grid()
show()
```

6a

Python program to find the image of vector $(2,3)$ when it is stretched horizontally

```
from pylab import *
x, y = 2, 3
X = 2 * x
Y = y
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
ylim(0,4)
xlim(0,6)
grid()
show()
```

b

Python program to find the image of vector $(4,0)$ when it is rotated by 90^0.

```
from pylab import *
x, y = 4, 0
X, Y = -y,x
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
ylim(-2,5)
xlim(-2,5)
grid()
show()
```

7a

Python program to find the image of vector $(2,4)$ when it is stretched vertically.

```
from pylab import *
x, y = 2, 4
X = x
Y = x*y
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
ylim(0,10)
xlim(0,4)
grid()
show()
```

b

Python program to find the image of vector $(3,3)$ when it is reflected about y-axis.

```
from pylab import *
x, y  = 3, 3
X =-1 * x
Y = y
# Creating our arrow
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
# X and Y coordinates
ylim(-5,5)
xlim(-5,5)
axvline(x=0,color="k",ls=":")
axhline(y=0,color="k",ls=":")
grid()
show()
```

8

Python program to verify the *rank-nullity* theorem for the linear transformation $T: R^3 \to R^3$ defined by $T(x,y,z) = (x - y + 2z\,,\, y\,,\, x + 2y + z)$

```
from sympy import *
A = Matrix([[1,0,1],[-1,1,2],[2,0,1]])
r = A.rank()
print('Rank of the linear transformation : r = ',r)
NullSpace =A.nullspace()
print('Null space of the linear transformation :\n ')
NullSpace = Matrix(NullSpace)
pprint(NullSpace)
n =NullSpace.shape[1]
#print('Nullity of the Linear transformation : n = ',n)
dim =int(input('Enter the dimension of the vector space U :'))
if dim==r+n:
    print('Rank and Nullity theorem holds good \ndim(u)= dim(R(T))+dim(N(T))
else:
    print('dim(u)!= dim(R(T))+dim(N(T))')
```

9a

Python program to find the image of vector $(3,4)$ when it is reflected about y-axis.

```
from pylab import *
x, y  = 3, 4
X =-1 * x
Y = y
# Creating our arrow
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
# X and Y coordinates
ylim(-5,5)
xlim(-5,5)
axvline(x=0,color="k",ls=":")
axhline(y=0,color="k",ls=":")
grid()
show()
```

b

Python program to find the image of vector $(0,5)$ when it is rotated by 90^0.

```
from pylab import *
x, y = 0, 5
X, Y = -y,x
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
ylim(-2,6)
xlim(-6,4)
grid()
show()
```

10a

Python program to find the image of vector $(3,3)$ when it is stretched horizontally.

```python
from pylab import *
x, y = 3, 3
X = 2 * x
Y = y
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
ylim(0,4)
xlim(0,8)
grid()
show()
```

b

Python program to find the image of vector $(4,5)$ when it is reflected about y-axis.

```python
from pylab import *
x, y  = 4, 5
X =-1 * x
Y = y
# Creating our arrow
arrow(0,0,X,Y,head_width =0.2,head_length=0.2,ec='g')
arrow(0,0,x,y,head_width=0.2,head_length=0.2,ec='r')
# X and Y coordinates
ylim(-6,6)
xlim(-5,5)
axvline(x=0,color="k",ls=":")
axhline(y=0,color="k",ls=":")
grid()
show()
```

Vector Calculus

a

Python program to find gradient of $\phi = x^2y + 2xz - 4$.

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative, log

var ('x,y,z')
v= ReferenceFrame ('v')

F=x**2*y+2*x*z-4
print("The Scalar function F is")
display(F)

gradF=diff(F,x)*v.x+diff(F,y)*v.y+diff(F,z)*v.z

print ("\n Gradient of F is")
display(Derivative(F,x)*v.x+Derivative(F,y)*v.y+Derivative(F,z)*v.z)
print("=",end="")
display(gradF)
```

b

Python program for *Green's theorem* to evaluate $\oint_c (x+2y)dx + (x-2y)dy$ where c is the region bounded by the coordinate axes, the lines $x = 1$ and $y = 1$.

```python
from sympy import *
var ('x,y')
p=x+2*y
q=x-2*y
f= diff (q,x)- diff (p,y)
display(Integral (f,[x,0,1],[y,0,1]))
soln = integrate (f,[x,0,1],[y,0,1])
print ("=",soln )
```

a

Python program to find gradient of $\phi = x^2yz$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative, log
var ('x,y,z')
v= ReferenceFrame ('v')
F=x**2*y*z
print("The Scalar function F is")
display(F)
gradF=diff(F,x)*v.x+diff(F,y)*v.y+diff(F,z)*v.z
print ("\n Gradient of F is")
display(Derivative(F,x)*v.x+Derivative(F,y)*v.y+Derivative(F,z)*v.z)
print("=",end="")
display(gradF)
```

b

Python program for *Green's theorem* to evaluate

$\oint_c (xy + y^2)dx + (x^2)dy$ where c is the closed curve bounded by $y = x$ and $y = x^2$.

```python
from sympy import *
var ('x,y')
p=x*y+y ** 2
q=x ** 2
f= diff (q,x)- diff (p,y)
display(Integral (f,[y,x ** 2,x],[x,0,1]))
soln = integrate (f,[y,x ** 2,x],[x,0,1])
print ("=",soln )
```

a

Python program to find div $\vec{F}$, given that $\vec{F} = x^3\hat{\imath} + y^3\hat{\jmath} + z^3\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative

var ('x,y,z')
v= ReferenceFrame ('v')

F1=x**3
F2=y**3
F3=z**3
F=F1*v.x+F2*v.y+F3*v.z
divF=diff(F1,x)+diff(F2,y)+diff(F3,z)
print ("Given vector point function F is ")
display (F)
print ("Divergence of F is")
display(Derivative(F1,x)+Derivative(F2,y)+Derivative(F3,z))
print("=")
display (divF)
```

b

Python program to find curl $\vec{F}$ given $\vec{F} = x^2yz\hat{\imath} + y^2xz\hat{\jmath} + z^2xy\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative

var ('x,y,z')
v= ReferenceFrame ('v')

F1=x**2*y*z
F2=x*y**2*z
F3=x*y*z**2
F=F1*v.x+F2*v.y+F3*v.z

curlF=(diff(F3,y)-diff(F2,z))*v.x-(diff(F3,x)-diff(F1,z))*v.y+(diff(F2,x)-diff(F1,y))*v.z
print (" Given vector point function is ")
display (F)
print (" curl of F is ")
display((Derivative(F3,y)-Derivative(F2,z))*v.x-(Derivative(F3,x)-Derivative(F1,z))*v.y+
        (Derivative(F2,x)-Derivative(F1,y))*v.z)
print("=")
display (curlF)
```

a

Python program to find curl $\vec{F}$, given that $\vec{F} = y^2x\hat{\imath} + 2x^2yz\hat{\jmath} - 3yz^2\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x*y**2
F2=2*x**2*y*z
F3=-3*y*z**2
F=F1*v.x+F2*v.y+F3*v.z
curlF=(diff(F3,y)-diff(F2,z))*v.x-(diff(F3,x)-diff(F1,z))*v.y+(diff(F2,x)-diff(F1,y))*v.z
print (" Given vector point function is ")
display (F)
print (" curl of F is ")
display((Derivative(F3,y)-Derivative(F2,z))*v.x-(Derivative(F3,x)-Derivative(F1,z))*v.y+
        (Derivative(F2,x)-Derivative(F1,y))*v.z)
print("=")
display (curlF)
```

b

Python program to find divergence of $\vec{F} = x^2yz\hat{\imath} + y^2xz\hat{\jmath} + z^2xy\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x**2*y*z
F2=x*y**2*z
F3=x*y*z**2
F=F1*v.x+F2*v.y+F3*v.z
divF=diff(F1,x)+diff(F2,y)+diff(F3,z)
print ("Given vector point function F is ")
display (F)
print ("Divergence of F is")
display(Derivative(F1,x)+Derivative(F2,y)+Derivative(F3,z))
print("=")
display (divF)
```

a

Python program to find curl $\vec{F}$, given that $\vec{F} = x^3\hat{\imath} + y^3\hat{\jmath} + z^3\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x**3
F2=y**3
F3=z**3
F=F1*v.x+F2*v.y+F3*v.z
curlF=(diff(F3,y)-diff(F2,z))*v.x-(diff(F3,x)-diff(F1,z))*v.y+(diff(F2,x)-diff(F1,y))*v.z
print (" Given vector point function is ")
display (F)
print (" curl of F is ")
display((Derivative(F3,y)-Derivative(F2,z))*v.x-(Derivative(F3,x)-Derivative(F1,z))*v.y+
        (Derivative(F2,x)-Derivative(F1,y))*v.z)
print("=")
display (curlF)
```

b

Python program to find div $\vec{F}$, given $\vec{F} = x^2\hat{\imath} + 3y\hat{\jmath} + x^3\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative

var ('x,y,z')
v= ReferenceFrame ('v')

F1=x**2
F2=3*y
F3=x**3
F=F1*v.x+F2*v.y+F3*v.z
divF=diff(F1,x)+diff(F2,y)+diff(F3,z)
print ("Given vector point function F is ")
display (F)
print ("Divergence of F is")
display(Derivative(F1,x)+Derivative(F2,y)+Derivative(F3,z))
print("=")
display (divF)
```

a

Python program to find curl $\vec{F}$, given $\vec{F} = (x + y + 1)\hat{\imath} + \hat{\jmath} - (x + y)\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x+y+1
F2=1
F3=-(x+y)
F=F1*v.x+F2*v.y+F3*v.z
curlF=(diff(F3,y)-diff(F2,z))*v.x-(diff(F3,x)-diff(F1,z))*v.y+(diff(F2,x)-diff(F1,y))*v.z
print (" Given vector point function is ")
display (F)
print (" curl of F is ")
display((Derivative(F3,y)-Derivative(F2,z))*v.x-(Derivative(F3,x)-Derivative(F1,z))*v.y+
        (Derivative(F2,x)-Derivative(F1,y))*v.z)
print("=")
display (curlF)
```

b

Python program to find gradient of $\phi = x^2 - 2y^2 + 4z^2$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative, log
var ('x,y,z')
v= ReferenceFrame ('v')
F=x**2-2*y**2+4*z**2
print("The Scalar function F is")
display(F)
gradF=diff(F,x)*v.x+diff(F,y)*v.y+diff(F,z)*v.z
print ("\n Gradient of F is")
display(Derivative(F,x)*v.x+Derivative(F,y)*v.y+Derivative(F,z)*v.z)
print("=",end="")
display(gradF)
```

a

Python program to find gradient of $\phi = x^4 + y^4 + z^4$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative, log
var ('x,y,z')
v= ReferenceFrame ('v')
F=x**4+y**4+z**4
print("The Scalar function F is")
display(F)
gradF=diff(F,x)*v.x+diff(F,y)*v.y+diff(F,z)*v.z
print ("\n Gradient of F is")
display(Derivative(F,x)*v.x+Derivative(F,y)*v.y+Derivative(F,z)*v.z)
print("=",end="")
display(gradF)
```

b

Python program to find div $\vec{F} = x^2yz\hat{\imath} + y^2xz\hat{\jmath} + z^2xy\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x**2*y*z
F2=x*y**2*z
F3=x*y*z**2
F=F1*v.x+F2*v.y+F3*v.z
divF=diff(F1,x)+diff(F2,y)+diff(F3,z)
print ("Given vector point function F is ")
display (F)
print ("Divergence of F is")
display(Derivative(F1,x)+Derivative(F2,y)+Derivative(F3,z))
print("=")
display (divF)
```

a

Python program to find gradient of $\phi = x^2 - y^2 + 2z^2$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative, log
var ('x,y,z')
v= ReferenceFrame ('v')
F=x**2-y**2+2*z**2
print("The Scalar function F is")
display(F)
gradF=diff(F,x)*v.x+diff(F,y)*v.y+diff(F,z)*v.z
print ("\n Gradient of F is")
display(Derivative(F,x)*v.x+Derivative(F,y)*v.y+Derivative(F,z)*v.z)
print("=",end="")
display(gradF)
```

b

Python program to find curl $\vec{F}$, given that $\vec{F} = x^2\hat{\imath} + 3y\hat{\jmath} + x^3\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x**2
F2=3*y
F3=x**3
F=F1*v.x+F2*v.y+F3*v.z
curlF=(diff(F3,y)-diff(F2,z))*v.x-(diff(F3,x)-diff(F1,z))*v.y+(diff(F2,x)-diff(F1,y))*v.z
print (" Given vector point function is ")
display (F)
print (" curl of F is ")
display((Derivative(F3,y)-Derivative(F2,z))*v.x-(Derivative(F3,x)-Derivative(F1,z))*v.y+
        (Derivative(F2,x)-Derivative(F1,y))*v.z)
print("=")
display (curlF)
```

a

Python program to find div $\vec{F} = (x + 3y)\hat{\imath} + (y - 3z)\hat{\jmath} + (x - 2y)\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x+3*y
F2=y-3*z
F3=x-2*y
F=F1*v.x+F2*v.y+F3*v.z
divF=diff(F1,x)+diff(F2,y)+diff(F3,z)
print ("Given vector point function F is ")
display (F)
print ("Divergence of F is")
display(Derivative(F1,x)+Derivative(F2,y)+Derivative(F3,z))
print("=")
display (divF)
```

b

Python program to find gradient of $\phi = xy^2 + yz$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative, log
var ('x,y,z')
v= ReferenceFrame ('v')
F=x*y**2+y*z
print("The Scalar function F is")
display(F)
gradF=diff(F,x)*v.x+diff(F,y)*v.y+diff(F,z)*v.z
print ("\n Gradient of F is")
display(Derivative(F,x)*v.x+Derivative(F,y)*v.y+Derivative(F,z)*v.z)
print("=",end="")
display(gradF)
```

a

Python program to find gradient of $\phi = x^2y^2 + y^2z^3$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative, log
var ('x,y,z')
v= ReferenceFrame ('v')
F=x**2*y**2+y**2*z**3
print("The Scalar function F is")
display(F)
gradF=diff(F,x)*v.x+diff(F,y)*v.y+diff(F,z)*v.z
print ("\n Gradient of F is")
display(Derivative(F,x)*v.x+Derivative(F,y)*v.y+Derivative(F,z)*v.z)
print("=",end="")
display(gradF)
```

b

Python program to find curl $\vec{F}$, given that $\vec{F} = x^3\hat{\imath} + y^3\hat{\jmath} + z^3\hat{k}$

```python
from sympy . physics . vector import *
from sympy import var, diff, Derivative
var ('x,y,z')
v= ReferenceFrame ('v')
F1=x**3
F2=y**3
F3=z**3
F=F1*v.x+F2*v.y+F3*v.z
curlF=(diff(F3,y)-diff(F2,z))*v.x-(diff(F3,x)-diff(F1,z))*v.y+(diff(F2,x)-diff(F1,y))*v.z
print (" Given vector point function is ")
display (F)
print (" curl of F is ")
display((Derivative(F3,y)-Derivative(F2,z))*v.x-(Derivative(F3,x)-Derivative(F1,z))*v.y+
        (Derivative(F2,x)-Derivative(F1,y))*v.z)
print("=")
display (curlF)
```

Numerical Methods for differential equation

1

Python program to find $y(0.1)$ for $\frac{dy}{dx} = y^2 + x^2$, $y(0) = 1$ using *Taylor's series* method considering up to third degree terms.

```python
from sympy import *
var("x,x1")
y=Function("y")(x)
f=x**2+y**2
x0,y0=0,1
fi=f
der=[f]
for i in range(3):
    der.append(diff(f,x).subs(diff(y,x),fi))
    f=der[i+1]
yx1=y0
for i in range(3):
    f1=lambdify([x,y],der[i])
    yx1=yx1+(((x1-x0)**(i+1))/factorial(i+1))*f1(x0,y0)
print("The Taylor's series expansion upto 4th degree term is")
display(yx1)
print(f"y({x1})=%0.4f"%yx1.subs(x1,float(input("enter the value of x at which you need y: "))))
```

2

Python program to find y at $x=0.3$ for $\frac{dy}{dx}-2y=3e^{x}$ and $y(0)=0$ using *Taylor's series* method considering up to third degree terms.

```python
from sympy import *
var("x,x1")
y=Function("y")(x)
f=2*y+3*exp(x)
x0,y0=0,0
fi=f
der=[f]
for i in range(3):
    der.append(diff(f,x).subs(diff(y,x),fi))
    f=der[i+1]
yx1=y0
for i in range(3):
    f1=lambdify([x,y],der[i])
    yx1=yx1+(((x1-x0)**(i+1))/factorial(i+1))*f1(x0,y0)
print("The Taylor's series expansion upto 4th degree term is")
display(yx1)
print(f"y({x1})=%0.4f"%yx1.subs(x1,float(input("enter the value of x at which you need y: "))))
```

3

Python program to find $y(0.1)$ by *Taylor's series* method when $y'+4y=x^{2}$, $y(0)=1$

```python
from sympy import *
var("x,x1")
y=Function("y")(x)
f=-4*y+x**2
x0,y0=0,1
fi=f
der=[f]
for i in range(3):
    der.append(diff(f,x).subs(diff(y,x),fi))
    f=der[i+1]
yx1=y0
for i in range(3):
    f1=lambdify([x,y],der[i])
    yx1=yx1+(((x1-x0)**(i+1))/factorial(i+1))*f1(x0,y0)
print("The Taylor's series expansion upto 4th degree term is")
display(yx1)
print(f"y({x1})=%0.4f"%yx1.subs(x1,float(input("enter the value of x at which you need y: "))))
```

4

Python program to solve by *Modified Euler's* method: $y'=e^{-x}$ with $y(0)=-1$, at $x=0.2$.

<table>
<tr><td></td><td>

```python
f=lambda x,y:exp(-x)
x0=0
y0=-1
h=0.2
n=int(input("Enter the maximum number iterations needs to be performed: "))
x1=x0+h
y1E=y0+h*f(x0,y0)
print("\nInitial guess by Euler's method is x=%0.2f y=%0.4f"%(x1,y1E))
print("By Modified Euler's Method")
print(f"Iteration\t\ty1({x1})")
for i in range(n):
    y1=y0+(h/2)*(f(x0,y0)+f(x1,y1E))
    print(i+1,"\t\t\t%.4f"%y1)
    if abs(y1-y1E)<0.0001:
        break
    else:
        y1E=y1
print(f"\ny({x1})=%.4f"%y1)
```

</td></tr>
<tr><td>5</td><td>

Python program to solve by *Modified Euler's* method: $y' = x + y$, $y(0) = 1$, at $x = 0.1$.

```python
f=lambda x,y:x+y
x0=0
y0=1
h=0.1
n=int(input("Enter the maximum number iterations needs to be performed: "))
x1=x0+h
y1E=y0+h*f(x0,y0)
print("\nInitial guess by Euler's method is x=%0.2f y=%0.4f"%(x1,y1E))
print("By Modified Euler's Method")
print(f"Iteration\t\ty1({x1})")
for i in range(n):
    y1=y0+(h/2)*(f(x0,y0)+f(x1,y1E))
    print(i+1,"\t\t\t%.4f"%y1)
    if abs(y1-y1E)<0.0001:
        break
    else:
        y1E=y1
print(f"\ny({x1})=%.4f"%y1)
```

</td></tr>
<tr><td></td><td></td></tr>
<tr><td>6</td><td>

Python program to find $y(0.1)$ by *Runge Kutta* method when $y' = x - y^2$, $y(0) = 1$

```python
f = lambda x,y: x-y**2
x0, y0 =0, 1
h = 0.1
x1 = x0+h
k1 = h*f(x0,y0)
k2 = h*f(x0+h/2,y0+k1/2)
k3 = h*f(x0+h/2,y0+k2/2)
k4 = h*f(x0+h,y0+k3)
y1 = y0+(1/6)*(k1+2*k2+2*k3+k4)
print("\nk1=%0.4f\tk2=%0.4f\tk3=%0.4f\tk4=%0.4f"%(k1,k2,k3,k4))
print("y(%0.2f)=%.4f"%(x1,y1))
```

</td></tr>
<tr><td></td><td></td></tr>
</table>

7	Python program to evaluate by *Runge Kutta* method: $\frac{dy}{dx}=3x+\frac{y}{2},\ y(0)=1$ at $x=0.2$

```
f = lambda x,y: 3*x+y/2
x0, y0 =0, 1
h = 0.2
x1 = x0+h
k1 = h*f(x0,y0)
k2 = h*f(x0+h/2,y0+k1/2)
k3 = h*f(x0+h/2,y0+k2/2)
k4 = h*f(x0+h,y0+k3)
y1 = y0+(1/6)*(k1+2*k2+2*k3+k4)
print("\nk1=%0.4f\tk2=%0.4f\tk3=%0.4f\tk4=%0.4f"%(k1,k2,k3,k4))
print("y(%0.2f)=%.4f"%(x1,y1))
```

8	Python program to find $y(1.2)$ by *Runge Kutta* method when $\frac{dy}{dx}=1+\frac{y}{x},\ y(1)=2$

```
f = lambda x,y: 1+y/x
x0, y0 =1, 2
h = 0.2
x1 = x0+h
k1 = h*f(x0,y0)
k2 = h*f(x0+h/2,y0+k1/2)
k3 = h*f(x0+h/2,y0+k2/2)
k4 = h*f(x0+h,y0+k3)
y1 = y0+(1/6)*(k1+2*k2+2*k3+k4)
print("\nk1=%0.4f\tk2=%0.4f\tk3=%0.4f\tk4=%0.4f"%(k1,k2,k3,k4))
print("y(%0.2f)=%.4f"%(x1,y1))
```

9	Python program to solve by *Milne's predictor and corrector* method: $\frac{dy}{dx}=x^2+\frac{y}{2}$ at $y(1.4)$ Given that $y(1)=2$, $y(1.1)=2.2156$, $y(1.2)=2.4649$, $y(1.3)=2.7514$. Use *corrector* formula thrice

```
func = lambda x, y: x**2+(y/2)
x = [1, 1.1, 1.2, 1.3]
y = [2, 2.2156, 2.4649, 2.7514]
h = 0.1
f = []
for i in range(4):
    f.append(func(x[i],y[i]))
    print(f"x{i}=%.4f\ty{i}=%.4f\tf{i}=%0.4f"%(x[i],y[i],f[i]))
#predict y4=y(x4)
x4 = x[3]+h
y4p = y[0]+(4*h/3)*(2*f[1]-f[2]+2*f[3])
print("\nThe predicted value at x4=%.4f is y4p=%.5f\n"%(x4,y4p))
#correction
for i in range(3):
    f4p=func(x4,y4p)
    y4c=y[2]+(h/3)*(f[2]+4*f[3]+f4p)
    y4p=y4c
    print(f"The corrected value at x4=%.4f in iteration {i+1} is y4c{i+1}=%.5f"%(x4,y4c))
```

10

Python program to solve by *Milne's predictor and corrector* method: $\frac{dy}{dx} = x^2 + y^2$ at $y(0.4)$ Given that $y(0)=1$, $y(0.1)=1.1113$, $y(0.2)=1.2507$, $y(0.3)=1.426$. Use *corrector* formula thrice

```python
func = lambda x, y: x**2+y**2
x = [0, 0.1, 0.2, 0.3]
y = [1, 1.1113, 1.2507, 1.426]
h = 0.1
f = []
for i in range(4):
    f.append(func(x[i],y[i]))
    print(f"x{i}=%.4f\ty{i}=%.4f\tf{i}=%0.4f"%(x[i],y[i],f[i]))
#predict y4=y(x4)
x4 = x[3]+h
y4p = y[0]+(4*h/3)*(2*f[1]-f[2]+2*f[3])
print("\nThe predicted value at x4=%.4f is y4p=%.5f\n"%(x4,y4p))
#correction
for i in range(3):
    f4p=func(x4,y4p)
    y4c=y[2]+(h/3)*(f[2]+4*f[3]+f4p)
    y4p=y4c
    print(f"The corrected value at x4=%.4f in iteration {i+1} is y4c{i+1}=%.5f"%(x4,y4c))
```

HOD