Complex Analysis Worksheet

1740256

1. Construct a menu driven calculator for the following operations to be performed on complex numbers: ¶

(a): Sum(2 numbers)

(b): Difference(2 numbers)

(c): Conjugate

(d): Polar form

(e): Plot the number entered on X-Y plane

(f): Plot the number entered on Argand plane

(g): Modulus

(h): Amplitude

(i): Real Part

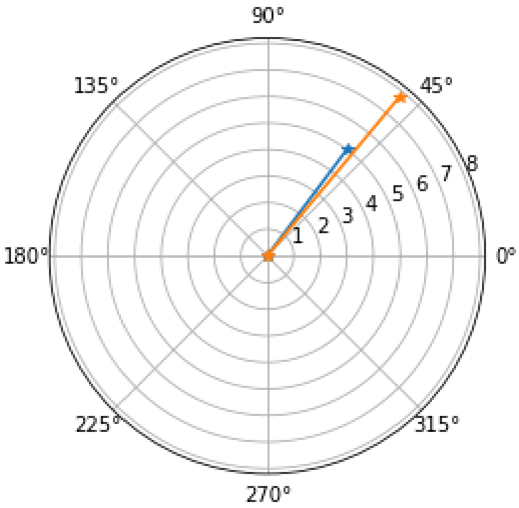
(j): Imaginary Part

In [2]:

```
import matplotlib.pyplot as plt
import numpy as np
from cmath import *

x1=int(input("Enter the real part:"))
y1=int(input("Enter the imaginary part: "))

x2=int(input("Enter the real part:"))
y2=int(input("Enter the imaginary part: "))

z1=complex(x1,y1)
z2=complex(x2,y2)

print("The entered complex numbers are:",z1, "and", z2)

print("1.Sum\n2.Difference (two numbers)\n3.Conjugate\n4.Polar form\n5.Plot the number ente

ch=int(input(("Enter Choice:")))
if ch==1:
    print("Sum=", z1+z2)
if ch==2:
    print("Difference=", z1-z2)
if ch==3:
    print("Conjugate of",z1,":", np.conjugate(z1))
    print("Conjugate of",z2,":", np.conjugate(z2))
if ch==4:
    print("Polar form of z1:", polar(z1))
    print("\nPolar form of z2:", polar(z2))
if ch==5:
    print("The plot on XY plane is:")
    plt.axhline(y=0, color="black")
    plt.axvline(x=0, color="black")
    plt.plot(x1,y1, x2, y2, color="green", linestyle='dashed', linewidth = 3, marker='o', m
    plt.xlabel('Real Axis')
    plt.ylabel('Imaginary Axis')
    plt.grid()
    plt.show()
if ch==6:
    plt.polar([0,np.angle(z1)], [0,np.abs(z1)], marker='*')
    plt.polar([0,np.angle(z2)],[0,np.abs(z2)], marker='*')
if ch==7:
    print("Modulus of",z1,":", abs(z1))
    print("\nModulus of",z2,":", abs(z2))
if ch==8:
    print("Amplitude of",z1,":", anngle(z1))
    print("\nAmplitude of",z2,":", angle(z2))
if ch==9:
    print("Real part of",z1,":", np.real(z1))
    print("\nReal part of",z2,":", np.real(z2))
if ch==10:
    print("Imaginary part of",z1,":", np.imag(z1))
    print("\nImaginary part of",z2,":", np.imag(z2))
```

```
Enter the real part:3
Enter the imaginary part: 4
Enter the real part:5
Enter the imaginary part: 6
```

```
The entered complex numbers are: (3+4j) and (5+6j)
1.Sum
2.Difference (two numbers)
3.Conjugate
4.Polar form
5.Plot the number entered on the X-Y plane
6.Plot the number entered on the Argand plane
7.Modulus
8.Amplitude
9.Real part
10.Imaginary part
Enter Choice:6
```



2. Verify the following for 2 complex numbers $z_1 = 5 - 7i$ and $z_2 = 4 + i$:

(a) $|z_1 z_2| = |z_1||z_2|$

(b) $|z_1 + z_2| \leq |z_1| + |z_2|$

(c) $amp(z_1 z_2) = amp(z_1) + amp(z_2)$

(d) $amp(\frac{z_1}{z_2}) = amp(z_1) - amp(z_2)$

In [4]:

```
import cmath
import math
z1 = 5 - 7j
z2 = 4 + 1j

propa1 = abs(z1*z2)
propa2 = abs(z1)*abs(z2)

if propa1 == propa2:
    print("A.\n\t\t As |z1 z2| == {0} and |z1||z2|== {1}\n\t\t We can say that |z1 z2| == |

propb1 = abs(z1 + z2)
propb2 = abs(z1) + abs(z2)

if propb1 <= propb2:
    print("B.\n\t\t As |z1 + z2| == {0} and |z1|+ |z2|== {1}\n\t\t We can say that |z1 z2|

propc1 = math.floor(cmath.phase(z1 * z2))
propc2 = math.floor(cmath.phase(z1) + cmath.phase(z2))

if propc1 == propc2:
    print("C.\n\t\t As amp(z1 + z2) == {0} and amp(z1)+ amp(z2)== {1}\n\t\t We can say that

propd1 = math.floor(cmath.phase(z1 / z2))
propd2 = math.floor(cmath.phase(z1) - cmath.phase(z2))

if propd1 == propd2:
    print("C.\n\t\t As amp(z1 + z2) == {0} and amp(z1)+ amp(z2)== {1}\n\t\t We can say that
```

```
A.
                 As |z1 z2| == 35.4682957019364 and |z1||z2|== 35.4682957019
364
                 We can say that |z1 z2| == |z1||z2|
B.
                 As |z1 + z2| == 10.816653826391969 and |z1|+ |z2|== 12.7254
30892660288
                 We can say that |z1 z2| <= |z1||z2|
C.
                 As amp(z1 + z2) == -0.705568177685211 and amp(z1)+ amp(z2)=
= -0.7055681776852111
                 We can say that |z1 z2| == |z1||z2|
C.
                 As amp(z1 + z2) == -1.1955255039389394 and amp(z1)+ amp(z2)
== -1.1955255039389394
                 We can say that |z1 z2| == |z1||z2|
```

3. Evaluate $e^{2n\pi i}$ for any 3 values of n.

In [6]:

```
import sympy as sy
val=[-1,4,23]
n=sy.symbols("n")
eq=sy.exp(2 * n * np.pi * complex(0,1))
for j in val:
    deq=eq.subs(n,j)
    print(deq.evalf())
```

```
1.0 + 2.0e-16*I
1.0 - 1.0e-15*I
1.0 - 1.3e-14*I
```

4. Find the locus such that $|z-1|^2 + |z+1|^2 = 4$.

In [7]:

```
from sympy import *
z = symbols('z')
x, y = symbols('x, y', real = True)

z = x + I*y

def roots(expr):
    expr = simplify(expr)
    print("The equation is:")
    eq = Eq(expr, 4)
    print("{0} = 4".format(expr))
    print("\nSolving w.r.t real axis, we obtain the following equation: ")
    eq1 = eq.subs(y, 0)
    print(eq1)
    print("\nRoots obtained are: ")
    root1 = solve(eq1, x)
    print("(0, {0}), (0, {1})".format(2, -2))
    print("\nSolving w.r.t imaginary axis, we obtain the following equation: ")
    eq2 = eq.subs(x, 0)
    print(eq2)
    root2 = solve(eq2, y)
    print("\nRoots obtained are: ")
    print("(0, {0}), (0, {1})".format(root2[0], root2[1]))

roots(abs(z-1) + abs(z+1))
```

```
The equation is:
sqrt(x**2 - 2*x + y**2 + 1) + sqrt(x**2 + 2*x + y**2 + 1) = 4

Solving w.r.t real axis, we obtain the following equation:
Eq(sqrt(x**2 - 2*x + 1) + sqrt(x**2 + 2*x + 1), 4)

Roots obtained are:
(0, 2), (0, -2)

Solving w.r.t imaginary axis, we obtain the following equation:
Eq(2*sqrt(y**2 + 1), 4)

Roots obtained are:
(0, -sqrt(3)), (0, sqrt(3))
```

5. Check whether $f(z) = logz$ is analytic. If yes, then find $f'(z)$

In [8]:

```
def analytic(u,v):
    print("Given expression f(z):",(u+1j*v))
    diff_u_x=sy.diff(u,x)
    print("\nDerivative of u wrt x:",diff_u_x)
    diff_u_y=sy.diff(u,y)
    print("Derivative of u wrt y:",diff_u_y)
    diff_v_x=sy.diff(v,x)
    print("Derivative of v wrt x:",diff_v_x)
    diff_v_y=sy.diff(v,y)
    print("Derivative of v wrt y:",diff_v_y)
    if(diff_u_x == diff_v_y and diff_u_y == -diff_v_x):
        print("\nf(z) is an analytic function.")
        print(sy.diff(u+1j*v))
        return True
    else:
        print("\nf(z) is not an analytic function.")
        return False


z=sy.symbols("z")
x=sy.symbols("x")
y=sy.symbols("y")
eq=sy.log(z)
eq=eq.subs(z,(x+1j*y))
im=sy.sympify(eq.subs(y,0))
rl=eq-im
print("imaginary part : ",im)
print("real part :",eq-im)
analytic(rl,im)
```

```
imaginary part :  log(x)
real part : -log(x) + log(x + 1.0*I*y)
Given expression f(z): -log(x) + 1.0*I*log(x) + log(x + 1.0*I*y)

Derivative of u wrt x: 1/(x + 1.0*I*y) - 1/x
Derivative of u wrt y: 1.0*I/(x + 1.0*I*y)
Derivative of v wrt x: 1/x
Derivative of v wrt y: 0

f(z) is not an analytic function.
```

Out[8]:

```
False
```

6. Verify whether $f(z) = z - \overline{z}$ is differentiable using the $C - R$ equations.

In [9]:

```
import math
import cmath
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
x,y = symbols('x,y', real = True)
z = (x+I*y)
zbar = (x-I*y)
fz = z-zbar
expr =fz.as_real_imag()
u = expr[0]
print("u=",u)
v = expr[1]
print("v=", v)
pderiv_ux= diff(u,x)
print("\n du/dx:", pderiv_ux)
pderiv_uy=diff(u,y)
print("\n du/dy:", pderiv_uy)

pderiv_vx= diff(v,x)
print("\n dv/dx:", pderiv_vx)
pderiv_vy=diff(v,y)
print("\n dv/dy:", pderiv_vy)

if(pderiv_ux==pderiv_vy) and (pderiv_uy== -(pderiv_vx)):
    print("\n Conclusion: Function is differenciable")
else:
    print("\n Conclusion: Function is not differentiable")
```

```
u= 0
v= 2*y

 du/dx: 0

 du/dy: 0

 dv/dx: 0

 dv/dy: 2

 Conclusion: Function is not differentiable
```

7. For the complex number $z = 3 + 3i$, plot the following in the same $X - Y$ plane:

(a) Reflection with respect to Y-axis

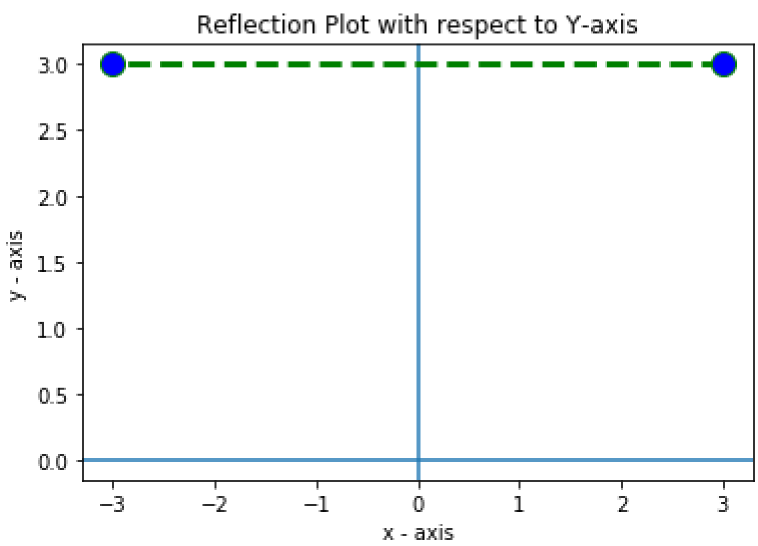
(b) Translation by $c = 1 + 2i$

(a)

In [10]:

```
def ref(a,b):
    import matplotlib.pyplot as plt
    import cmath
    import sympy as sp
    z = complex(a,b)
    zl = [a,-a]
    print("The entered complex number is: ",z)
    w = z.conjugate()
    wl = [b,b]
    print("The reflection of ",z," is: ",w)
    plt.axhline()
    plt.axvline()
    plt.plot(zl,wl,color='green', linestyle='dashed', linewidth = 3, marker='o', markerface
    plt.xlabel('x - axis')
    plt.ylabel('y - axis')
    plt.title('Reflection Plot with respect to Y-axis')
    plt.show()

x = int(input("Enter the real part: "))
y = int(input("Enter the imaginary part: "))
ref(x,y)
```

```
Enter the real part: 3
Enter the imaginary part: 3
The entered complex number is:  (3+3j)
The reflection of  (3+3j)  is:  (3-3j)
```

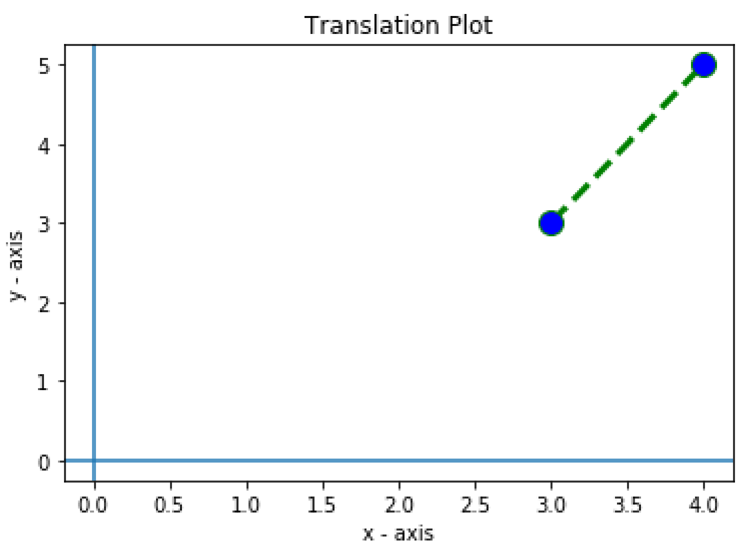


(b)

In [11]:

```
def trans(a,b,c,d):
    import cmath
    import matplotlib.pyplot as plt
    z = complex(a,b)
    print("The entered complex number is: ",z)
    c = complex(c,d)
    print("The entered complex constant is: ",c)
    w = z + c
    print("The translation of ",z," is: ",w)
    wr = w.real
    wi = w.imag
    zl = [a,wr]
    cl = [b,wi]
    plt.axhline()
    plt.axvline()
    plt.plot(zl,cl,color='green', linestyle='dashed', linewidth = 3, marker='o', markerface
    plt.xlabel('x - axis')
    plt.ylabel('y - axis')
    plt.title('Translation Plot')
    plt.show()

a = int(input("Enter the real part of the complex number: "))
b = int(input("Enter the imaginary part of the complex number: "))
c = int(input("Enter the real part of the complex constant: "))
d = int(input("Enter the imaginary part of the complex constant: "))
trans(a,b,c,d)
```

```
Enter the real part of the complex number: 3
Enter the imaginary part of the complex number: 3
Enter the real part of the complex constant: 1
Enter the imaginary part of the complex constant: 2
The entered complex number is:  (3+3j)
The entered complex constant is:  (1+2j)
The translation of  (3+3j)  is:  (4+5j)
```



8. For the complex number $z = 5 - 2i$, plot the following in the same argand plane:

(a) Magnification by $A = 4e^i$ with respect to Y-axis

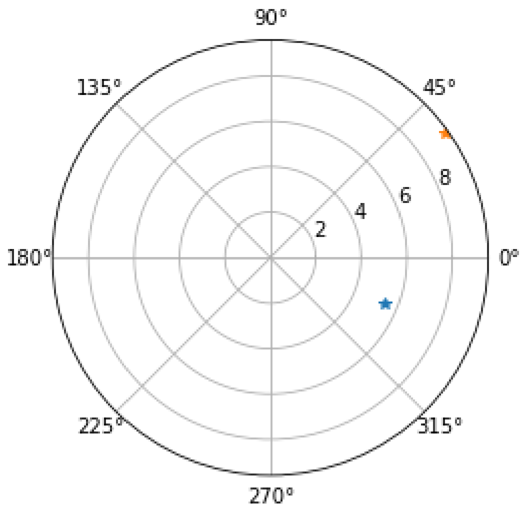
(b) Inversion by $A = 2$

In [12]:

```
import cmath
import matplotlib.pyplot as plt
import numpy as np
def MR(z, a):
    r, phi = cmath.polar(z)
    r1, phi1 = cmath.polar(a)
    absolute = r + r1
    angle = phi + phi1
    plt.polar(np.angle(z), abs(z), marker = '*')
    plt.polar(angle, absolute, marker = '*')
    return absolute, angle

MR(5-2j,4*exp(1j))
```

Out[12]:

(9.385164807134505, 0.6194936228876351)

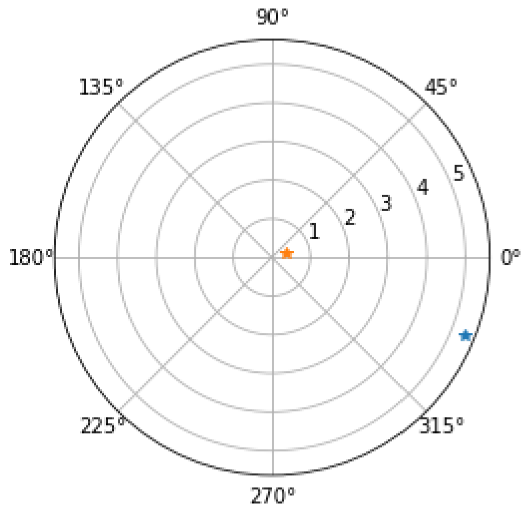


In [13]:

```
from sympy import *
def inversion(z,a):
    inversion = a/z
    plt.polar([np.angle(z)], [abs(z)], marker = '*')
    plt.polar([np.angle(inversion)], [abs(inversion)], marker = '*')
    return inversion

inversion(5-2j,2)
```

Out[13]:

(0.3448275862068966+0.13793103448275862j)



9. Find the points at which the functions $w_1 = cosz$ and $w_2 = \frac{1}{2}(z + \frac{1}{z})$ is not conformal.

In [14]:

```
def conformal(eq):
    a=[]
    for i in np.arange(-5,5,0.5):
        z=sy.symbols("z")
        difeq=sy.diff(eq,z)
        difeq=difeq.subs(z,i)
        if(difeq==0.0):
            a.append(i)

    if(len(a)==0):
        print("Conformal at all points")
    else:
        print("Conformal at all points except : \n",a)

w1=sy.cos(sy.symbols("z"))
w2=(1/2)*(sy.symbols("z")+1/(sy.symbols("z")))
conformal(w1)
conformal(w2)
```

```
Conformal at all points except :
 [0.0]
Conformal at all points except :
 [-1.0, 1.0]
```

10. Find the fixed points of the transformation $w = \frac{3z-4}{z}$

In [15]:

```
import sympy as sy
z=sy.symbols("z")
w=(3*z-4)/z
eq=sy.Eq(w,z)
eq1=sy.solve(eq)
sy.pprint(eq1)
```

$$\left[\frac{3}{2} - \frac{\sqrt{7}\cdot i}{2}, \frac{3}{2} + \frac{\sqrt{7}\cdot i}{2}\right]$$