Python Command Sheet

MATH102 Recitation Lab

Command	Purpose	Example
import sympy as	*Import sympy library*	**import sympy as smp**
import numpy as	*Import numpy library*	**import numpy as np**
from sympy import func	*Import func or everything (*) from sympy library*	***from sympy import **
import matplotlib.pyplot as	*Import plotting library*	**import matplotlib.pyplot as plt**
print() or print("")	*Print expression or message*	**print(express) or print("message")**
display()	*Display expression*	**display(express)**
smp.sqrt() or np.sqrt()	*Square root function*	**smp.sqrt(2) or np.sqrt(2)**
smp.log() or np.log()	*Natural Logarithm function*	**smp.log(1) or np.log(1)**
smp.exp() or np.exp()	*Exponential function*	**smp.exp(2) or np.exp(2)**
smp.pi or np.pi	*Pi symbol or value*	**smp.pi or np.pi**
smp.sin() or np.sin(); **smp.cos() or np.cos();** **smp.tan() or np.tan();** **smp.cot() or np.cot();** **smp.sec() or np.sec();** **smp.atan() or np.atan();** **smp.sinh() or np.sinh();** **smp.asinh() or np.asinh();**	*Sine function;* *Cosine function;* *Tangent function;* *Cotangent function;* *Cotangent function;* *Secant function;* *Hyperbolic sine;* *Inverse Hyperbolic sine*	**smp.sin(a) or np.sin(a);** **smp.tan(a) or np.tan(a);** **smp.atan(a) or np.atan(a);** **smp.cot(a) or np.cot(a);** **smp.sec(a) or np.sec(a);** **smp.atan(a) or np.atan(a);** **smp.sinh(a) or np.sinh(a);** **smp.asinh(a) or np.asinh(a);**
smp.Abs() or np.Abs()	*Absolute value*	**smp.Abs(x) or np.Abs(x)**
xn**	*x to the power n*	**x**2**
smp.Symbol('var')	*Define variable var using smp*	**x=smp.Symbol('x')**
symbols("vars", real=True)	*Define vars as real variable*	**x, y = symbols("x y", real=True)**
smp.Function('func')()	*Define function f(x)*	**F=smp.Function('f')(x)**
func.sub(var, val)	*Substitute var with val in func*	**q=f.subs(x,1.3)**
f = lambda x: expression in terms of x	*Define f as a function of x and evaluate it over an array*	**f = lambda x: 1/(1+x**2)**
lambdify(var, func, "numpy")	*Allow func to take values of array of var values*	**f = smp.lambdify(x, 1 + 2*x**2, "numpy")**

smp.plot()	*Plot a function from sympy*	**smp.plot(f)**
np.linspace(arg1, arg2, k+1)	*Get k+1 points from arg1 to arg2*	**xvalues = np.linspace(a, b, n+1)**
plt.plot(xarg,yarg,'r')	*Plot from pyplot of yarg in terms of xarg*	**plt.plot(xvalues,yvalues,'r')**
smp.plot_implicit()	*Plot implicit relation*	**P1=plot_implicit(x-10*y**2+4*y)**
smp.plot_implicit(And(cond1,cond2))	*Plot region satisfying cond1 and cond2*	**plot_implicit(And(x<3*y-11*y**2, x>10*y**2-4*y))**
solve(f)	*Solve f= 0 and get the list of solutions*	**L=smp.solve(f)**
solve((expr1,expr2), var1, var2)	*Solve expr1= expr2 and get the list of solutions in terms of var1 and var2*	**L= smp.solve((x-10*y**2+4*y, x-3*y+11*y**2), x, y)**
smp.integrate(func,(var,a,b))	**Integrate func with respect to var over the interval [a,b]**	**smp.integrate(f,(x,0,4))**
smp.integrate(func,var)	**Indefinite integral of func with respect to var**	**smp.integrate(f,x)**
smp.diff(func,var)	*Differentiate func with respect to var*	**fx = smp.diff(f,x)**
obj.transform(expr,var)	*Substitute expr with var in obj*	**k=i.transform(3*x+5,u)**
smp.simplify(expr)	*Simplify expression*	**smp.simplify((x+x*y)/x)**
Obj.doit()	*Evaluate Obj*	**test = diff(u,x)** **test.doit()**
smp.Piecewise((val1, cond1),(val2, cond2),(…))	*Define a piecewise function*	**f = smp.Piecewise((6, (1<= x) &(x<2)), (9 - x**3 , (2<= x) & (x <= 4)))**
len(list)	*Get the length of list*	**L=len(list)**
List[p]	*Get item at position 'p' of List.*	**display(list[i])**
range(k)	*Range of values from 0 to k-1*	**range(L)**
if (): **else:**	*If conditional statement*	**if (b <=a):** **print("Error: value of b <= a")** **else:**
for i in range(L):	*For loop for i=0 to L-1*	**for i in range(len(list)):**