```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/home/saanvi/.local/lib/python3.10/site-packages/matplotlib/projections/__init__.py:63:
UserWarning: Unable to import Axes3D. This may be due to multiple versions of Matplotlib being
installed (e.g. as a system package and as a pip package). As a result, the 3D projection is
not available.\n",
      "  warnings.warn(\"Unable to import Axes3D. This may be due to multiple versions of
\"\n"
     ]
    },
    {
     "ename": "ModuleNotFoundError",
     "evalue": "No module named 'ganymede'",
     "output_type": "error",
     "traceback": [
      "\u001b[0;31m---------------------------------------------------------------------------
\u001b[0m",
      "\u001b[0;31mModuleNotFoundError\u001b[0m                       Traceback (most recent
call last)",
      "Cell \u001b[0;32mIn[5], line 3\u001b[0m\n\u001b[1;32m      1\u001b[0m
\u001b[38;5;28;01mfrom\u001b[39;00m \u001b[38;5;21;01m__future__\u001b[39;00m
\u001b[38;5;28;01mimport\u001b[39;00m print_function\n\u001b[1;32m      2\u001b[0m
get_ipython()\u001b[38;5;241m.
\u001b[39mrun_line_magic(\u001b[38;5;124m'\u001b[39m\u001b[38;5;124mmatplotlib\u001b[39m\u001b[38;5;124m'\u0
\u001b[38;5;124m'\u001b[39m\u001b[38;5;124minline\u001b[39m\u001b[38;5;124m'\u001b[39m)
\n\u001b[0;32m----> 3\u001b[0m \u001b[38;5;28;01mimport\u001b[39;00m
\u001b[38;5;21;01mganymede\u001b[39;00m\n\u001b[1;32m      4\u001b[0m
ganymede\u001b[38;5;241m.
\u001b[39mconfigure(\u001b[38;5;124m'\u001b[39m\u001b[38;5;124muav.beaver.works\u001b[39m\u001b[38;5;124m'\u
\n\u001b[1;32m      5\u001b[0m \u001b[38;5;28;01mimport\u001b[39;00m
\u001b[38;5;21;01mmatplotlib\u001b[39;00m\u001b[38;5;21;01m.
\u001b[39;00m\u001b[38;5;21;01mpyplot\u001b[39;00m \u001b[38;5;28;01mas\u001b[39;00m
\u001b[38;5;21;01mplt\u001b[39;00m\n",
      "\u001b[0;31mModuleNotFoundError\u001b[0m: No module named 'ganymede'"
     ]
    }
   ],
   "source": [
    "from __future__ import print_function\n",
    "%matplotlib inline\n",
    "import ganymede\n",
    "ganymede.configure('uav.beaver.works')\n",
    "import matplotlib.pyplot as plt\n",
    "import numpy as np\n",
    "import sympy as sym\n",
    "from IPython.display import YouTubeVideo, HTML\n",
    "sym.init_printing(use_latex = \"mathjax\")"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
```

```
    "### Enter your name below and run the cell:\n",
    "\n",
    "Individual cells can be run with `Ctrl` + `Enter`"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "metadata": {},
   "outputs": [],
   "source": [
    "# ganymede.name('Saanvi Chugh')\n",
    "# def check(p):\n",
    "#    # ganymede.update(p,True)\n",
    "# check(0)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {
    "scrolled": true
   },
   "outputs": [
    {
     "ename": "NameError",
     "evalue": "name 'YouTubeVideo' is not defined",
     "output_type": "error",
     "traceback": [
      "\u001b[0;31m---------------------------------------------------------------------------
\u001b[0m",
      "\u001b[0;31mNameError\u001b[0m                                 Traceback (most recent
call last)",
      "Cell \u001b[0;32mIn[3], line 1\u001b[0m\n\u001b[0;32m----> 1\u001b[0m
\u001b[43mYouTubeVideo\u001b[49m(\u001b[38;5;124m'\u001b[39m\u001b[38;5;124m9vKqVkMQHKk\u001b[39m\u001b[38;5
width\u001b[38;5;241m=\u001b[39m\u001b[38;5;241m560\u001b[39m,
height\u001b[38;5;241m=\u001b[39m\u001b[38;5;241m315\u001b[39m) \u001b[38;5;66;03m# Video by
http://www.3blue1brown.com/\u001b[39;00m\n",
      "\u001b[0;31mNameError\u001b[0m: name 'YouTubeVideo' is not defined"
     ]
    }
   ],
   "source": [
    "YouTubeVideo('9vKqVkMQHKk', width=560, height=315) # Video by http://www.
3blue1brown.com/"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "scrolled": true
   },
   "outputs": [],
   "source": [
    "YouTubeVideo('bRZmfc1YFsQ', width=560, height=315) #Note: All Khan Academy content is
available for free at khanacademy.org"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
```

```
   "source": [
    "### Power Rule\n",
    "\n",
    "The derivative of $x^n$ is $nx^{n-1}$\n",
    "\n",
    "[Read more](https://www.mathsisfun.com/calculus/power-rule.html)\n",
    "\n",
    "[Other derivative rules](https://www.mathsisfun.com/calculus/derivatives-rules.html)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {},
   "outputs": [
    {
     "ename": "NameError",
     "evalue": "name 'sym' is not defined",
     "output_type": "error",
     "traceback": [
      "\u001b[0;31m---------------------------------------------------------------------------
\u001b[0m",
      "\u001b[0;31mNameError\u001b[0m                                 Traceback (most recent
call last)",
      "Cell \u001b[0;32mIn[4], line 2\u001b[0m\n\u001b[1;32m      1\u001b[0m
\u001b[38;5;66;03m# Creating algebraic symbols \u001b[39;00m\n\u001b[0;32m----> 2\u001b[0m x
\u001b[38;5;241m=\u001b[39m \u001b[43msym\u001b[49m\u001b[38;5;241m.
\u001b[39msymbols(\u001b[38;5;124m'\u001b[39m\u001b[38;5;124mx\u001b[39m\u001b[38;5;124m'\u001b[39m)
\n\u001b[1;32m      3\u001b[0m x\n",
      "\u001b[0;31mNameError\u001b[0m: name 'sym' is not defined"
     ]
    }
   ],
   "source": [
    "# Creating algebraic symbols \n",
    "x = sym.symbols('x')\n",
    "x"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/latex": [
       "$$x^{2}$$"
      ],
      "text/plain": [
       " 2\n",
       "x "
      ]
     },
     "execution_count": 5,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "x = sym.symbols('x')\n",
    "expr = x ** 2\n",
```

```
   "expr"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 6,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/latex": [
      "$$\\frac{d}{d x} x^{2}$$"
     ],
     "text/plain": [
      "d ⎛ 2⎞\n",
      "──⎝x ⎠\n",
      "dx    "
     ]
    },
    "execution_count": 6,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "sym.Derivative(expr) # does not actually compute the derivative"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 7,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/latex": [
      "$$2 x$$"
     ],
     "text/plain": [
      "2·x"
     ]
    },
    "execution_count": 7,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "sym.Derivative(expr).doit()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 8,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/latex": [
      "$$2 x$$"
     ],
     "text/plain": [
```

```
      "2⋅x"
     ]
    },
    "execution_count": 8,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "sym.diff(expr) #equivalent to doit()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 9,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAXcAAAEFCAYAAAAYKqc0AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDIuMi4yLCBodHRwOi8vbWF0cGxvdGxpYi5vcmcvhp/
UCwAAIABJREFUeJzt3XlYlXX+//
HnDUd2RFA2QTZBBFwQUTSXXNLMzD3NLDE1zabJatqmmqVl0vZlWuZLY6WOaU5WOmau7mLW+KCoKCAiMiqyHY49+8P1B8ZKuo55z7L+3FdXH
+XN8cWHe/
nciqqqCCGEsC0OWgcQQghhfFLuQghhg6TchRDCBkm5CyGEDZJyF0IIGyTlLoQQNkjKXQghbJCUuxBC2CCd1gGEbVMUxQ/
oBbQGKoE0IFVVVYOmwYScYqcoSpMQVGU/sDzgA+wFzgDuADtgLbAN8A7qqqWaxZSCBsm5SMQlGUt4B/
qqp6spHHdMAMAwwFFV1f+ZPZwQdkDKXQghbJDsUBUmpSjKAkVRVRvrcDlMUZzWmYSwIox3kRG7sKkFixYwJQpU5g/fz6TJ09m6NCh7N+/
A+xpnEsLmmXKzjGzvEQBs3ryZ/v3706pVK/bu3UtAQIDWkYSwZIox3kRG7sKkFixYwJQpU5g/fz6TJ09m6NCh7N+/
X+tYQtg8GbkLkxo5ciQpKSn4+fkBsHPnTmbMmMHevXs1TiaExTLKyF3KXZhdTU0NTk50NTk50WscQwlLJZhluV577TWKi4sbfczJyYl169169bx448/
mjmVEPZDph8QJtGxY0Y0uFxccSEhIwNfXl6qqqKjIyMti3bx933HEHEHL7zwgtYxhbBZsllGmMSDDz7779GgULePPNN/
Hz8yM/Px9XV1diYmLo27cvrq6uWkcUwlIZDbOMjNyFSezevZuTJ0+yZNmdfLMbL/4ge8+7a/+xnw/Bq/8iJN3W9xbBW/mM+nzlUlgn6vFCyh1DxnXBxhqzgRmSUvaqV/1nsa/7qS1qFIbgA1yVJ4z/mNVK3V5xZFC/bfTqQW3+PVfV51fr2AU9CVCdT56eeuxA0DXGFl8aB9mMGt3ON20sewQrz2qNnM3l+JqRMcYj/
9U9kAYANvrlsaVN0SGyYVPTTABWk82lD6q7UIN+Ghcsjl5y3bHk/
1EFMBqz6NQ4ueV/mIzxOd0FyQ/Aw6APbyYjyQ6nVtjpLLdGyEBEs2QlbaPpA36wqWb1UsKMH+bFDspX4BTRCNyc+IPqsl6/
rHYKMu7Hl08ltw6dRSPw8/HLKi7ZJJx2bU6m3h8sxG+bq6NdnkiDE6Evq9hKfsUt+KRPWUQsSsfoHuuR0jbJpfsu43Tdv/
D8Bs1LH8QG/YXtM+/7TkrrP/2v9J2uTDWlvUV6Jql/n3b5zR/E+HF3eJRWR5GA0w+BiE1xf6JVrk6Ofe6UtZF8Xr4ua/
cA3tZiR1bq96+PlNb4Q3PCZYHPXZCXAG2WIoqqiq9KHmhDQaPiMu/
k9n1yBIdj9QUjtNYCGbJUHoBZahjyjLHWQY3T0xe0jmxDGEUVjS9klyeuhuc+3mMvIDZAZXs97ZRNDhHf/
tbW7oeyQBhLVxwSbW9QDaUMTswWYDmJ59lhbKJ0uAxlkvK+yQZU80rH5Y19NWk0Ei5iqwhNDj7FmAUNGHxBv92/
UgaTr24rmb/dqDUXpjd8qhUGwRhZ2u+oAjA/ieUHGB3hyA1M5n5kYg+7kZRqyOTjSmX0fWTLsyN7pFI4ez3m5VdUP+
XNyGGpvJQ7ErRVLkZYXFU0/lHJc6NTYTFYhR7ssy7VQIwPWi8ufcHvkydA+hpbL4ZaJrcIvJgyiekhaPbTKk0QzV2r7JR+dGtj9wDZ6l3dtuD3vHAZzqeQJg/
k+8A4C3r9Yz1bhIvn0gyI8Pxxg0pa2hTiuIPu/d0uBlsbw/UE+YNSn/dSY5cf0Ki7xTA1F9ndYkP6ah3/vPnCKnyDqSc1aotKU/
SK8hGfwRHNLLQkagmFyOPeIWI8rJq1FEaLXiQyzQkJ/
DZlQpdwQyQz7tVBCsBIsO4fDeD92RfLMh8PPHNAaa5PWiVeS8OFeWpEySbJZe3cu4kPOJCmdJo5CXIGtfPOWgbvPuQZmF3Z+wMTMOPgDGqxW9w5UQYndsrHdKALKTmW+l/
dkQeyQgJcAh5DWLGyhmZ2VWtZjehbBmnyoHoCm7UD0Qef+V1T9+XvXyHF1WRf/wYnMvzgFbSbGt8ZgvDkyGi7n9ZR0CkyFnffzaNO0eBcQGjrvUyuJkdGaQ
```

Bo/7ZaR2mUVZQ7wMj4YFq4NeOf6zK0jiKEsHNF56tZuOMkI+KDCG3prnWcRllNubs6OTK9bwS/
ZJxl94kSreMIIezY3M1ZVOnrNJ/W91qsptwBJiaF4uPuJKN3IYRmSi/UMH/
bCYZ2DCTSz0PrOFdlVeXu7qxjWp9wNqQXsj+nVOs4Qgg79OXWbM5X63nMgkftYGXlDjCpZ5hsexdCaOJcVS2fb85iUKw/
MYHNtY5zTVZX7h7OOqb2CmfN4TOk5ZVpHUcIYUf+tzuPipo6/jjAskftYIXlDpDcKwxPF52M3oUQZnO+Ws/
7a48yrGMgnYK1u/B1U1lluTd3acaUXuGsOljA4fxyreMIIezAvK3ZlF6o5aHeljWHzNVYZbkDTOkVjoezjo/
WZWodRQhh485V1ZKy6TgD2vsR38byR+1gxeXu5daMybeFsSOriPTTMnoXQpjOl1uyKaus5Yk7orSO0mRWW+4AU3uHo6rw/
hrZ9i6EMI3yqlo+++U4d8T4WcW29kusuty93Z2YmBTCT2mnOXhKjpwRQhjfF5uzKa/
S88Qd7bSOckOsutwBpvaJoLmLjvdWH9U6ihDCxpRV1vLvzccZFOtPhyAvrePcEKsvdy/XZjzcJ4I1h8/
IWatCCKP6fHMW56r0VrWt/RKrL3eAh3qH4+3WjHdl9C6EMJKyC/
Vno94Z509ca+satYONlLuHs44Zt7dl49FCdp+Q+d6FELdu7ubjnKu2vm3tl9hEuQNM6hlKKw8n3vlZRu9CiFtTXFHD3pxSRnUJsvg5ZK7GZs
S+1nuETEM2We4Ag2P9SQhpwftrMqiqrdM6jhDCgu3KLmZ9eiGP9GuLl2szreMYhc2Wu6IoPDekPafLq/
hya7bWcYQQFkpVVd5amY6vpzMP3WYdV1lqCpstd4CkiJb0j/blk/
WZlF2o1TqOEMICbThayM7sYh4fEImrk6PWcYzGpssd4Nkh7TlXrefTjce0jiKEsDAGQ/
2ovY2PK+O7hWgdx6hsvtxjApszMj6IL7ZkcbqsSus4QggLsuJAPofyy3lqUDucdLZVh7b1r7mKpwa1w6CqfLBWLsdn7UpLSxk7dizt27cnJi
XZjCsUyDDOwdpHcfo7KLc2/
i4MTEplCWpORwrPK91HHELZs2axZAhQzhy5Aj79+8nJiaGOXPmMHDgQDIyMhg4cCBz5szROqawAgt3nODomfOM6RqMo4OidRyjs4tyB3hsQC
q189U6jknYTbm38nBmWp8IfjlayP4c+bXdGmVlZeHr68tDDz1Ely5dmDZtGhUVFRQUFBAYGAhAQEAABQUFjb4+JSWFxMREEhMTKSwsNGd0YW
3Cce3uQuvLT8sk4pZIb1ez549e5g5cyZ79+7F3d39d5tgFEW56n/
W6dOnk5qaSmpqKr6+tjlaE9eXW3KBL7ZkM6pLkFVePq+p7KrcPVyaMa13BLuyS2RSMSsUHBxMcHAwSUlJAIwdO5Y9e/
bg7+9Pfn4+APn5+fj5+WkZU1i4d34+igI8PTha6ygmZVflDjAuMZhof0/
mrDxCjV4mFbMmAQEBtGnThvT0+v0ma9euJTY2luHDhzNv3jwA5s2bx4gRI7SMKSxYWl4Z3+3NY2rvcFq3cNU6jknptA5gbjpHB164O4bkz3c
pEaB1J3IB//vOfTJw4kZqaGiIiIvjiiy8wGAyMGzeOuXPnEhoaypIlS7SOKSyQqqr8Y/lhfNydeMSKL5/XVHZX7gC3t/
OlbztfPlybwZiEYLzdnbSOJJooPj6e1NTU392/du1aDdIIa7I+/
Qzbjhfx8vA4mrvYxuRg12J3m2UueXFoDOer9Xy4Tk5sEsLW6esMzPnpCOGt3Lk/
ybamGbgauy336ABPxncLYcG2ExyXE5uEsGmLduWgr1N5cWgMzRzto/bs4195FU8NaoezzoE5Px3ROooQwkTKLtTy7s/
p+DV3ZmCM/
RxJZdfl7uvpzMx+bfn5UAHbjxdpHUcIYQLvrz1KWWUtfx0WZ7MnLDXGrssdYFqfCKL9PVi44wR1BjmxSQhbklFwjvnbTnBf9xBiWzfXOo5Z2
DBuTo78aVA7reOYnd2XO8CwTq3pHubDW6vS5YpNQtiI9eln2HS0kCfuaEdLD2et45idlDv185H8bXgspRdqeG/
NUa3jCCFuUY3ewKs/
HqatrzuTeoZqHUcTUu4XxbX24v6kEBZsP0H66XNaxxFC3IJ5W7PJOlvBX4bF2s2hj1eyz3/1VfxpUDQezjpe/
uGgzBophJU6e76aD9dm0D/al37R9nPo45Wk3Bvwdnfi6cHt2HqsSGaNFMJKvb0qncraOl4aFqt1FE1JuV/h/
qRQYgKb89ryw1TW1GkdRwhxA37NLSX1RDF/HBBJW18PreNoSsr9Co4OCn+/
J5a80kr+tfGY1nGEEE1UZ1D5y9I0yir1TOkdrnUczUm5NyIpoiX3dG7NvzYeI6f4gtZxhBBN8PWuHPbnlvHS3TF42sGsj9cj5X4Vf76rPT7u
vYTrDvS+AWXhRCW4c2VRzhfpeeVER3sav6Ya5Fyv4YpvSOI8vPgr8sOys5VISzUnpMlLN6Vw5Te4bTz99Q6jsWQcr8GJ50Dr47sQG5JJR+tl
HB0QS3spd6zgWScq9iRwcFF4b2YFag4GUX+TYdyG0UlGt5y9LD9LO34PpfdtqHcdiSbnfgLjWXkxMCuXj9ZlyzVUhNPLe6qPklVby+qiOOOm
v1+pqpVj34UwpwO5ZXy+JYv7k0JIDPPROo5Fk3K/
Qa5OjswZ3ZHsogu8v0aOfRfCXPR1Bv783a+0vHiAg7g2KfebcFtkK8YntuGzX46TllemdRwh7MJ/
dpwkLa+cv98Th5erTAx2PVLuN+mFu2No6e7Es9/8Sm2dQes4Qti0E0UVvLXyCA/
2CGVoxwCt41gFKfeb5OXajFdGdOBQfjkpMnOkECZjMKg8+82vOCgKj/
ZvK8e0N5GU+y0Y0iGAuzoE8MHaDI4Vntc6jhA2aeGOE+zIKualYTEEerlqHcdqSLnfopdHxOGic+D5//2KwSBTE5hDXV0dXbp0YdiwYQBkZW
++SYODvUf3aKiIlq0aIFOVz/da3BwMHl5MrmbtTtVWsk/lh+mR4QPE7uHaB3HKkm5G0mIjzt/GhzN1mNFzN+WrXUcm/
Tjjz/
i5+dH165db+r1KSkpJCYmkpiYSGFhoZHTCWO5tDmmzqDy5pjOODjI5pibIbPbG9F93dqw6uBp5qw8Qp92vrT19dA6kk3ZsmUL33//
PStWrKCqqory8nJmzZpFaWkper0enU5Hbm4uQUFBjb5+
+vTpTJ8+HYDExERzRhc34H+7czlVWsmzQ6IJaSmbY26WjNyNSFEU3hjTCWedI08t2Y9eTm4yqtmzZ5Obm0t2djaLFy9mwIABLFy4kP79+/
PNN98AMG/ePEaMGKFxUnGzThZd4G/fH8TP05nknmFax7FqUu5G5t/chddGdmB/Tin/
2nhM6zh24Y033uDdd98lMjKSoqIipk6dqnUkcRPqDCpPLdmHg6Lw5r2yOeZWKSa8bJxd71V87Ks9rDp4mqV/
6EVcay+t44grJCYmkpqaqnUM0cDH6zN5a1U6743vzKguwVrH0ZJRfqrJyN1EXh3RgRZuTjz19X6q9TI1sBDXkpZXxnurj3J3p0BGxje+z0Tc
F2d+LNMZ1ILzjHBzI1sBBXVVbxxNf76OlhxP/
GNlBTlYyEil3E+rf3o8/9I9k8a4ctmae1TqOEBZpzk9HyDxznrfv7UwLNyet49gMKXcT+0P/trRwa8aTS/
ZRXCFzngjR0C8ZhXy5NZvJt4XRJ8pX6zg2RcrdxNycdHx4XxdKKmp59pv9mHAHthBWpaSimpd/
OESknwfP3yVXVjI2KXcz6BDkxXN3tWfN4TMs2H5C6zhCaE5VVZ7+769U1Oj54L54XJo5ah3J5ki5m8mUXmH0j/
blteWHOZxfrnUcITQ1d3MWa4+cYUafCDlU2ESk3M1EURTeurczzV2a8fiivVTWyOGRwj7tyynljZVHGBzrT/
JtYVrHsVlS7mbUysOZ98Z3JuPMeV5dfkjrOEKYXVllLY99tQc/TxfeGttZDns0ISl3M+sT5cuMvhF8teMkKw/
kax1HCLNRVZXnvvmV02VV/PP+Lni5NdM6kk2TWSE18KfB0RSUV/
Pi0jRiWjcntKW71pGEMLkF20+w8uBpXhjanoQQb63j2DwZuWvASefAnwa3Q29QmfmfPVTVyvZ3YdvS8sp47cfD9I/
2ZVrvCK3j2AUpd4208XHjvfGdOZRfzt+WHdQ6jhAmc66qfju7j7sT74yLl9kezUTKXUMD2vvzWP9Ivk7NYUlqjtZxhDA6VVV5bfkhThZf4MM
Doule7iP1nHsipS7xhwdFD6c0IUWbs14dOEeyqtqtY4khFFsO1bEP1YcpmfbVkySqyqZnZS7BWjl4czH9yeQV1LJM/
+V+WeE9csrreQPX+0hrKUb74yTqyppQcrdQiSG+fDS3bHkllTy0bpMreMIcdOqaut4ZMFuavQGUiYl4ukix7NrQcrdgiTfFko7f0/
eWX2Unw+e1jqOEDdMVVVe+O4AB/
LKeG98PG19PbSOZLek3C2IoijMHt2RTsFePPn1PtJPn9M6khA3ZP62E3y7J49ZA6MYFOuvdRy7JuVuYVyaOZLyYCJuzjoenp9KiVzgQ1iJHc
ezheWMGnE7vS1s9T60jiIil3C/fi0BjCW7rzwndpbDpaqHUcIS5TVZWXvktjc+ZZZo/

uyG2RrbSOJBqQcrdwOkcH3rsvnnb+njy6cA9HTssl+oRl+GTDMb5OzeGPAyK5N7GN1nHEFaTcrYCHs47PJyfi7uzIQ1/
soqC8SutIws79sP8Ub61KZ0R8a54a1E7rOKIRUu5WItDLlc8nd608sn7nVUW1XutIwk6lZhfzp//
up1uYN2+O7SSXyrNQUu5WJK61Fx/dn8Dh/
HJeX3GYWjlEUphZ9tkKHp6fSlALV1IeTMRZ56h1JHEVUu5Wpn97P965tzMLd5zkuf/9isEgk4wJ8yiuqOGZb/
YD8PnkbnjL3OwWTa6haoVGJQSTU1LJu6uP4uPmxIt3x8ivxsKkzlfreeiLnZw5V81nkxIJbyXX/
bV0MnK3Un8cEMnk28L49+YsPt14TOs4ZpGTk0P//
v2JjY0lLi6ODz74AIDi4mIGDRpEVFQUgwYNoqSkROOktqVaX8eMBamknSrnlREdSAyTi25YAyl3K6UoCn8dFsuI+Na8uTKdxTtt/
yQnnU7HO++8w6FDh9i+fTsff/
wxhw4dYs6cOQwcOJCMjAwGDhzInDlztI5qM+oMKk8s3seWzCLeHNNJJJgOzIlLuVszBQeGtsZ25vZ0vL3x3gBUH8rWOZFKBgYEkJCQA4OnpSU
pZ3mL8NiGdM1WOtI4gYoJrzqj+zpM5MLNXpmLdrL4dPneGVEHAPa2/7oKjs7m759+5KWlkZISAilpaVAfSF5e3tfvt1QSkoKKSkpABQWFnLi
aw0canKTh//jxjxozh/
fffp3nz5r95TFGUq+5cnj6JynNZAAASOUlEQVR9OqmpqaSmpuLr62uOqFZJVVVeX3GYL7dm0zuqlUzfa6Wk3G2El2szFkztTpS/
Bw/
PT2VzxlmtI5lEbW0tY8aMYeLEiYwePROAf39/8vPrN0nl5+fj5+enZUSrpqoqb61K57NfspjUM5SX5EgsqyXlbkNauDnxn6lJRLRyZ+q8XWz
uHDhzNv3jwA5s2bx4gRI7SKaPU+WJvBvzPmCH+/
J06K3YrJNncbVHS+mgmfbSenuJIvHupGj4iWWkcyis2bN9OnTx86duyIg0P9uOT1118nKSmJcePGcfLkSUJDQ1myZAk+Ptc+XC8xMZHU1FRz
7lOK+vOMzohGDeHNNJLrihHaOseCl3G1V4rr7gHRWFvw+PpWdbmY61ISn3/09VVd7+OZ2P1x/
jkb4RPDOkPY5S7FqSHari6nw9nfnq4SQ8XXQkf7GLtYcLtI4kLJCqqrzy4yE+Xl+/KeZZKXabIeVuw/
w8XUiZlEi0vyczFuzm+/2ntI4kLIjBoPLCd2l8sSWbh3qF8fqoDrIpxoZIuds4H3cnvno4iYRQb2tXO2z/
TFZxfbV1Bl5alsainSd5tF9b/
josVnae2hgpdzvg6dKMeQ91v3wma8om+5iLRjTuQo2e6RcPl312SDTPDmkvxW6DpNzthKuTIykPJnJ3x0D+/
UsWb69Kl+mC7VBxRQ0TPtvBxqOFzLg9gkf7RWodSZiIlLsdcdI58OGELoxOCOKj9ZnM+nofVbV1WscSZpJTfIGxn27lSH45nz7QlYlJoVpHE
HNCF/
bnlDHm062cKKrQOpYwkZ8O5HPvv7bRoXVzvnmkpxS7nZByt2P3dG7NwoeTKL5Qw4TPtrMru1jrSMKIVFXlo3UZzFy4h5jA5rw+uiNR/
p5axxJmIuVu57qF+fDtzNuZ9tZ5EdXPTDHlTV1vHE1/t4+
+ejjIxvzaKHe+Dn6aJ1LGFGUu6CCF8P3h7bmdvatuLP3x7ghe8ONUKM3aB1L3KT8skpmLd7Lsn2neOb0aN4bH49L4wEJMyMv0gWRkDvLqjc
uJn00+f4vwe68of+kXIMu52ScheXXTqS5qP7u3C88DzT5++2uWmDbVWdQeW0W91/
mInvh7OzJ3cjTs7BGgdS2hIyl38zrBOrVk8owfnqmqZO0HcH7/ycjr5ONtNYqqLz1SR/vpMP1mYwukswS//
Qi7a+HlrHEhqTcheNauffnB/
+2JuxcCH8c10mEz7e7bzqnSSq1jiStsO1bE8I+2sD07mDfGd0OTtezvh6iTb14XM5y6aYOnePF787gA6RwfEaVsP17E9/
+u+tc/
nXlVbx9ur0pm7JYshcQE8NiCSSuNZeWscSxiEX6xDmk3W2gj9+tQcViAlszl+GxeLl2kzrWDfNmsv9QG4ZTy3ZR8aZ8zzYI5Q/
D22Pm5OcbG5D5GIdwnzCW7nz7czb6B/
tx3d787jzvU2sTz+jdSy7Ultn4P01Rxn1yRbKq2qZN6k7s0t5en/
7udowXnu7RrMS3fH4OVmXXPTWNvIPS2vjI/XZ/
JT2mlGxrfm5eEd8HKz3t+cxDXJZhmhnWp9HR+syWDvyRIyzpznxbtjGBkfZDXHVFtLuZdV1vLuz+ks3s
h2gyKXdhGQwGlcW7cnhj5RHaB3gSHeDJ4wOja0XhrHW0q7Lkcj94qozXlx9my7EiOgd78drIjnQMliNh7IiUu7AsReer+Xh9JvO2ncBF58CM
Ldvjxub+fLHTH+TOgegqNctNreSLkLy3Ss8DxvrUxn5cHT+Ho688QdUYxPbIPO0XIOzrKkci86X81H6zNZuP0kigKTe4Xx6O2RssPUfkm5C8
RnQ3o9RCUE467Q/
g9ISyr3wXDVfbDnOzwfPcPzsee7t2oYnBkUR6OWqaS6hOSl3YflUVWXtkQI+WX+MPSdL8fN0ZkrvcO5PCqG5Yj3YjUy3L/
WTRBT775ThLUnOoqTMwpVc4E7q3IdLP8ndEC7OQchfWQ1VVtmQW8a+Nx9iceRZPZx33J7XhgR5htPFxM3sec5e7qqrsyCrmpwP5LNh+AkcHh
GvTMQrKqth9soTeka0Y360Ng2L9zbbJxlzlXnahlv/tyWXhjhMcK6yghauOB3qE8UCPUAK85MpIolFS7sK65RZf4L4+7c/
lvag6nyqrwdmvGqC7BjE5oTVxrL5OeEGXKcq+qrWND+hl++DWf/
NIq9pwsIb5NCyYmhTCsU2vcnXXmuL95CdVX1Ie6VNXVsP17E9/
tPsfpQAeer9bR0d+KBHqEMivWnQ5Acpy6aTMpd2J6Sihp+PnSaVQcL2Jx5lhq9AR93J8YmBBEd0JzbIls2ejTJypUrmTVrFnV1dUybNNo3nn3
7RspdVVXOlFdzML+MX3PLOFVaycq8+UeZO1O9h7FiiNOvgSlLpm2O3Oz1zW1nVzN1d2D+dVbx/pWH6OmkAZkTKPTAvMkORNDpOfMTJY6vWMZ
2kfYNCl3Yb9UVeV0WRWH8sv5Yd1Wdh89QcdufUg7Vc7J4gu0qCsmIjwCnYMDjg4KeoMBB0VBUQAV1syewsA/
f06N3oCTzoEqfR06BwVHBwf0dQb255Ti7+VCr8hW+Hk608rDmbjWXsS1bo67s+VNpyBsimWXe1xcnOrqavln2hUWFuLr66t1jOuSnFdXUlJC
APQG1QMqorBoHLufAUVFRUA1FaUERQehaJwufQdHZTLBe/
k6IAlzF4s33fjsYaMALt37z6oqmqHW34jVVVN8tW1a1fVGkhO49Ii59atW9XBgwdfvv3666+rr7/+
+jVfI+vTuKwpVkVFVBVJVI3Sw7PkRVq9bt25kZGSQQlZVFTU0NixcvTsdHH33EnXfeSV1dHVOmTCEuLk7rWEJo
PysoiKSmJyMhIxo8fT01Nza3EaZLx48THx9PfHw/q8sLAw0nbsSHx8PImJiY0+x5T+/ve/
ExQUdDnrihUrTn3eypUriY6OJjIykjlz5pg55JTzzD00b9+eTp06MWrUKEpLSxt9bVhYGB07diQ+Pp7ERp0mPHjxxMZGUlSUhLZ2dlmywaQk5ND//
79iY2NJS4ujg8++OB3z9mwYQNeXl6Xfw9eeeUVDZI27v+ejvvbEP9m0b9+eTp06MWrUKEpLSxt9nlbr83rrp7q6mvHjxxMZGUlSUhLZ2dlmywaQk5ND//
79iY2NJS4ujg8++OB3z9mwYQNeXl6XPwuvvPKWTNecr3voaqqPP7440RGRtKpUyf27Nljjpi/8+fDzNmzfn/
fff/81ztFqfU6ZMwc/Pjw4d/v9RjsXFxQwaNIioqCgGDRpESUlJo69NPX8cAGILHB/
bGdOnVSq6qq1OPHj6sRERGqXq//3SE/9957r7po0dnoob/
97W/qW2+9dc3n6PV6NSIiQj127JhaXV2tdurUST148KA5Yl62evVqtbb20vO//vWv6l//+tdGnxcaGqoWFhaaM9plTVmfH3/8sfrggw+qqqqqixYtUseNG2fWjKdOnVL37NmjqqqqlpWVqVFRUb/LuH79enXYsGFmzXU11/t5/vjjj+qdd96pGgwGddu2bWr37t3NmO63rrYuM4v1bxu17FEr7Q+SXdRP36T0kDTkc9h6Dj/ppiOxsVM8wlAFNNzVAmV0ws7OHNxbWCKlSJ2aq2dI5mp2g3+s/xUqy4Gu7nm0rQoa08bTm2b3IMwmUwd4wmGnrKU7zj3NZOmcn4jHcWwaHwt6bgHMA1wM18vUzq8Wm4NcnKz5o6CjhMGX5nmeAoaq9f+9YVrzFWSS5rOeW9qVFl08/i0p/0MTZxOAY6aWK/
VBZt24dY8eOBSA5OZmlS5e5+qo5nFhz8mcVmKIPKAeLg9I7eYJ5m8aQ9wT0LwWHKyZ8D8/nn7Yz8jeoDj+8ELOzKWGfKtYQrcdkmx6W8LcUz
ua/j5u0YH3gmsVlW1WFXVEmA1MOS7KEB0+zfli9+P3D8CHmhwey4w9orXmaUjZ/m28FDCf/

```
flGicLAFnAHmA3MN1cuRos/+9ANvAr8DmN/KQGxgL/bnD7QeAjc2dtsPwfGn7ftV6fTVk/QBoQ3OD2MaCVRusvDDgJNL/
i/n5AEbAf+AmI0yjfNb+HwI9A7wa31zbsBQ3yfg481sj9mq3Pi9/
jtAa3Sxv8XWl4u8H9TwMvNbj9F+Dp6y3ruodCKoqyBgho5KEXVVU17zCxiZqYeQKw6Bpv01tV1TxFUfyA1YqiHFFVdZO5cgKfAq9Sf9TRq8A
yNiZfn9ZMURQP4H/AE6qqll/x8B4gVFXV84qiDAWWAlHmzogVfQ8VRXEChgN/
buRhS1mfv6GqqqooitF+bbxuuauqesdNvG8e9SPxS4Iv3tdQEdBCURSdqqr6qzznplwvs6IoOmA00PUa75F38c8ziqJ8B3QHjPpBbuq6VRTl
Fx4Uf/
ZNBtFUZpRX+wLVVX99srHG5a9qqorFEX5RFGUVqqqnjVnziZ8D83yeWyiu4A9qqoWXPmApazPiwoURQlUVTVfUZRA4Ewjz8mj/
reNS4Kp31pyTaY6FPJ74D5FUZwVRQmn/qfibza6XyyB9dT/6gyQDJjrN4E7gCOqqja6gVhRFHdFUTwv/
R0YTP2v72Zz8Rt9yairLH8XEKUoSvjFkcp91K97s1EUZQjwLDBcVdULV3mOVuuzKevne+o/e1D/
WVx3tR9QpqDUz0E8Fzisquq7V3lOwMXnoShKd+r/
35r7B1BTvoffA5OUej2AMlVVtdroftXfzC1hfTbQ8PN3tQ5cBQxWFMVbURRv6td944cgNnSL249GAblANVAArGrw2IvUb79MB+5qcP8KoPXF
qZwH8BZzNt9/oSeOSK+1oDKxrk2n/x6yD1mx/
Mvb1wAXCA+m3u3wOBV+a8eHsocPTiutYiZyaQA+y7+PUvS1qfja0f4BXqfxgBuFz87GVe/
CxGmHn99aZ+09uvDdbhUOCRS59R4LGL620/sB24TYPvc6PfwytyKsDHF9f1ATTa3g64U1/WXg3u03x9Uv/
DJh+ovdibU4GW1O+byADWAD4Xn5vIb/
cXTbn4Gc0EHmrK8kx5hqoQQgiNyBmqQghhg6TchRDCBkm5CyGEDZJyF0IIGyTlLoQQNkjKXQghbJCUuxBC2CApdyGEsACKonRTFOVXRVFcLp
qR/
A3TUbuQghhIRRF+R5YDIRTP1ngYzf7Xtedz10IIYTpKYoyCahVVfUrRVEcga2KogxQVXXdTb2fjNyFEML2yDZ3IYSwQVLuQghhg6TchRDCBk
B1i/MoDH0RCwAAAAAElFTkSuQmCC\n",
      "text/plain": [
       "<Figure size 432x288 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "sym.plot(expr);"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 10,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAXcAAADzCAYAAAB9llaEAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/
AAAADl0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uIDIuMi4yLCBodHRwOi8vbWF0cGxvdGxpYi5vcmcvhp/
UCwAAIABJREFUeJzt3XlcVPX+P/
DXQQRRAUFkEZR9V0QEpVJMBRM1yDKXLDW9YYvWzcrbpqJp2nrzWmc8rrcyp91K8hdVFIrJcklQZFVAYcdFJF15vP7w+KriYg4M2eW1/
Px4JGcc+bMuzPDi8Nnzud9JCEEiIjIsJjIXQAREakfw52IyAAx3ImIDBDDDnYjIADHciYgMEMOdiMgAMdyJiAwQw52IyACZyl0AGR5JkuwB3A
fj23btmm5KiLjxPYDpDb9+/
fHgw8+iC5duiAkJAS9evVCfX09srKycOLECURGRuL1l1+Xu0wio8BhGVKbJ554Aps3b8a7774Le3t7KBQKWFhYwN/
fHxEREbCwsJC7RCJ9oJZhGZ65k9qkpaXh4sWL+OKLL3DgwIEb1tXV1d11uBcUFGD69OkoKSkyePBn5+flwc3PD1q1
0La9euRW5uLpydnVuWCyEgSRJyc3Pvav8KhQIhSGoqanBoEGD8P3332PDhg2wtbXFq6+
+ipUrV6KqqgrvvPPO3f7vEMmFV8uQbnrmmWewdu1ajT9PbGws5s5s6di7lz5yIlJQVOTk5QKBS4//77kZmQfnJ1KnCxVXsXLXGayZNojhTsY
DB69OhxQ5jb2NigqqqqzX2Ehobi2LFjmi6VRKs8qIVcVvzdsmjUYET695C7rJgx3I
Pg52SJ18f6Yea97jAz1c2J/
gx3IqI2CCGw7ZQCb+84A8Wlejw80BmvRvvB3qqL3KW1ieFORHQLmcWXsTgpHUdxtal6ha6guFORPQXl+qa8M+957D5yHmEu
uykTl1UY8NrgvXh7tC5tuZnKXdscY7kREAE4UVOM/h/
OQePIiQl1tsDFmMPo5W8tdVocx3InIqJXVNODdXWfxdVohfBy648NJAzBhoLVss0vVgeFOREyapqVmJzUcu4J97z6Gu6gGuSYk5ER6YN8ob3c0NIx
+Dm7HONXH8Z/j5xHcN8e2PX3CLw21t9ggh3gmTvpkZkzZZzZ2Lu3LmYPn36DctffPFPzyyzJVRfqkqLoOb28/g+2/
```

K+BiY4H4mECM8rPX+yGY1jDcSW9EREQgPz9f7jJID9U3KfGfn/Kxat85CAG8GOmDOcM90KWz/
A2+NIXDMqT3Pv74YwQFBWHWrFmoqqq65XYJCQkIDQ1FaGgoysrKtFghyUUIgeSMEoz+50EcOFuK+33skTx/
OF6I9DboYAd4D1XSM/n5+Rg/
fjxOnz4NACgpKYGdnR0kScLChQuhUCiwfv362+6H91A1fHnltVjyQzpSMsvgZd8di8cHYJgONvhqBe+hSuTg4NDy76eeegrjx4+XsRrSBbUN
iyU1zTiwQG98Y9oX9hb6naDL01huJPemDp1KlJSUlBeXg4XFxcsWbIEKSkpOHHiBCRJgpubGz799F8yyQZnFFca/
CVmleJ/s7WWP/kAIT0tZG7LFlxzJ2MEsfcDUPV1Ub8c885/PfoeVhbdMaCMX6YFNpHrxp8tYJj7kRknJQqgf/
3awHe230WXvbd8Xi4K+ZH+aBHV/1r8KUpDHci0itp56sQn5SO34suYbCbLeJjAhDQW38bfGkKw52I9EJpTT3WHMjBhp/
z4WBljlVTghEzoLdBzi5VB4Y7Eem0JqUKG3/
Ox0fJWfCy74anh3ti3kgvdDOgPjCawKNDRDrrcFY54n9IR3bpFdzv2wuLxgfAo1d3ucvSCwx3ItI5hVVXsXz7Gew8XYy+tl3x2fRQjPI3zAZ
5mJNShb6O/fAy6N98Ldhht3gS1MY7kQkOyEE9maUYOm2DBRW1WFckBNei/
aDi01XuUvTWwx3IpJVTtkVLPkhAw1NSnQzM8WXTw3BvZ52cpel9xjuRCSLKw3NWL0vC+t/
ykOXzp0wP8oHj4e7Gl2DL01huBORVgkh8P2JImw9VohfciowKdQFC8b4wa67udylGRSGOxFpzemiS4hPSsex81UY5WeP75+7D8F9eshdlkFi
rARL8bfOk0hjsRaYxSJfBV6gV8k1aA9IuXMf0eN7wY5QNri85yl2bwGO5EpBHH8iuxOCkd6RcvY4i7LbbNGwpfRyu5yzIaDHciUqvSy
VY+2MO/vNTPpysu2D11IEYH+TE2aVaxmuOSG/
MmjUL9vb2N9xKr7KyEvFRUBSqqqpkrNC4NTarkHHAwByPeT8HR3IEo8N8IT+14ajgfZuVEWDHfSGzNnzsSuXbtuuWMUUqMGjUJKWVl
vOItwj55YMy0Erzzgh65mHByQC8Od9EZERARsbW1vWJaYmIgZM2YAAAGbMmIHvv/
9ejtKMVkHlVcRtOobp61OhUgn8Z2YYPp8ZBje7bnKXZvT4a5X0WklJCZycnAAjo6OKCkpueW2CQkJSEhIAACUlZVppT5DVd+kxNqUHKTmVe
84XDqYSF3afQXDHfSaw4ODlAoFHBycoJCoYC9vb3cJRms7NIaxCdl4HB2OfwcLbElLhzhHj3lLotugWPupNdiYmKwceNGAMDGjRsRGxsrc0W
D555/
D1dUVW7dulbtMg6FSCXx7vAgrd55FRW0DnhnuidlD3dGTDb70giSE0NS+NbZjorsVGhqKY8eOyV2Gzvq98BIWJZ3G8QvVCO7TA0tjAxHkwgZ
OBefpOSgZzczvDcxCI+EuLDBlx5iuBMRmpUqfJl6Ae/
vzsTVxmY8d78X4oZ7wKoLG3zpK4Y7kZFLzavEosTTOFtcg3s9eyI+JhA+DpZyl0V3ieFOZKSKL9Vhxc6zSDxxEc49LLBmWgii+zmyD4yBYLg
ikdPySWwF/
JyusmjoQYW62t38gGRSGO5GBuFTXhFXJ57DzdDGuNirxVmwgHhviik6cXWqUGO5Eek6lEvgmrRDv7DqLyquNeHa4J2YP84BtNzO5SyMZMdyJ
JgZmqCF0Z5Y8a9bmzwZZx4g2yiPzk7OwMA7O3tMWHCBKSmprYZ7rpCCIEfTinw9vYM2HQzw/
ggJ7wa7Qd7qy5yl0Z6juFOeq+2thYqlQqWlpaora3Fnj17sGjRIrnLuq0zisuIT0rH0bxKBPa2wtLYQAxyZYMvUg+GO+m9kpISTJgwAQDQ3N
TDlLC+bPBFasVwJ73n4eGBkydPyl3GbSlVAoknirBs+xlUX23EtCGueGm0D3p0ZYMvUj+GO5EW/HahCosT03FGcQkj/
RzwQqQ3AnuzwRdpDsOdSIPKahrwzq6z+CatEPaW5nj/0WDEBvfm7FLSOIY7kQY0KVXY+HM+M+ViVnob5ZiTnDPTBvpDe6m/
NHjrSD7zQiNfs5uxzrfszBwaxyDPfphUUPBsCzV3e5yyIjw3AnUpOi6jos356BHb8XY5hXT/x7eigi/
e05BEOyYLgT3aX6JiUSxUSDuZiTUo2AGB+lA/
iIjzY4IxvXAn6iAhBPaeKcXSbRkorKpDdJAjFjzkDydrC7lLI2K4E92JnLIaxP+QgZ+yyuHv1g1R/7kVbs4g1OKNfpqtVWxQyvFu0hp6MV+zzCVCVuyaveDS1yi9WTmVtRIn6pvWOnGaoN1JM1d2Cy6lKH/ZhE8yV
L0ncaFM2m3hRJNa0VvJ66jXc17OShu/2+SWWSsPL9FEFL+CYQWbVc3rxo2P/X72Nlvi2HFw1jbWtF1HPJsJsl0Ra2lzqgmxiqxVJ+lMRuhHp9RwZ2lR6wfR+gW2V8/6dJRMB3O1ZmpHx4zMw6fwz+cKp2QbjJ1wRMSnssBPcCV0UuMsLszvMgc8dQ9yJ8pw7vlh/W
q6n4lLkClsQTqPlM0T1qbQ4w37SWFl0J5Kw4lQ06ciKMj07jBXuyTcTMq8/6E5dOv5MQ5B8CQXp4pAIKbcxILE4Yd/
0NtZAp5LMSvqzr7OqXIWvUkQ1/Z8XNSUJNWnKqjhDf4cvc+4LEhWS6KX/Nfd3MVlc5y7+8r67fyt0j1nr5n+V+OEh7GX/
oSX+YHXkGlQA7s1rFvz8YsHLn9Ntv2DiyP4VAMhTB63MyP/VxHJnuZCdBeY2tpSi8WVFqxb1HZEO2FVpwtwZ/
A4yFvUD4GrNDTlJGN8+rp+oz81AP7jZcvB3EDfI9S4ivpDh+0Lu2K5VE8p/6RHrGM1Vs9/lv/
4yZJ1FmHtE5ixQEWmEnAHTrIlnZaW9h/+hkcIzQcA2ZsP8PeYTRVNR5zG3kXY3jX4bT1XVrEWZIPCNOMjVd6eTTGE2W8rMU7ng5YDXk5Smm3vhO+c1afQ/Q1Wdn1DL3ka0WK0rSxTT4N23fsUmg9n5WGKMCnQ71PCSHqf2zmxEtMQd/
+NqnjU8HyIrvQOyyJoNXB5VjhAHpQVIn77mx0DR5NEyGkS6XCGeq/
kO+8nEf1VpRdXFjHnmFDA3q4LhFe3iUH0HbHCMrt/
zjGIZ9ltdyqmRmYgu0dZnhyhN9F2x8p3WSl3/
pUvJ2XZbIkTUa0WDALbR4Zv+63zZ6KhZAlF6fyhXnAt8Ou/
L1EOqZ+Ff5o+qQwjctIhycgFdFfZMzf/W6M7OQTc1I2tEQKNFiK3yNe9tLE1jcLFxXuDlb+XiWaEFqvqaRJc4m3GWcIXzPJUDdWvG/
XeMBVCjVtzW39z4C7+fdCFUo7mrXZa4cRLRGrZtCApFLqaZbkFNMd/NdB6L5GfAgvhEHfqQpfkAt3YcLcfY4DmiPJ4IoG1j5hDfAvpT/
Kq8tVrJt4bnQ0LQEUIkVFyT2OP8PU0TTbhKH7VGnA/
qeJGZ4kIgtGT1ZvhFt7zcxPoWgrtaTeZPN5Ztso8zdPFTG1Fbz06u/
Ba9rCevTuqxSrVeXmQzEQKoF/fhY7T2IQe3UOq7kn8dC4mkTkpYXRcJwxDjAPWg3Xz5rYdfbK8hvn3umyxYHbQnYoQ3eXAKH8nWqgpMxFqSmYv
/GKpfPhFZoxnz/7Jg1HPr0wjMBfEKz7ARMjVUl5ErPuJ2DhD5TAHdiCUH8pUBlnOgJjbD2nfD4FmIVrH/
zS3fnEjdvIg2C36bn9NcXqDaBTHNXsBuMCzs8yTgA+DaptEYsoERJEJXK5nd/
CfGbV3HiJAPuhnh8w+iVvuobi2hJGR81KNnLvpyWSl+lEOyRpCQpnaQi3fNDs4VQRrL0iJfvfV1u/sGZPKH+cA2hhG/
m6ryXi13GsqWIeNpfSvVxxnnJFVa+xjoTrp/
cQDn8+vLbISyOmSp6VxjX7vs6OrhlXGmWBn3MSWj/tm4NjGGVTvGmrZkyVgpwPfd09eS0MOMZVKyOLvI4FhKH2zlGm9/1nOpmUIOaOrvuO/d6hWKzrEuDkLjj2YRr4Ee/
wjx+w5LqbPGtZIVi0IjZPdqZZuHPg7mjOYy5uKqzlMtl1K3GT+SSeVHLmsKMaJrNg1OXoDuZxa7uvIBYmgS/
+rVfPiRIGkjYHW1qU4PHhyF6hJObqLUOEFNh+D/
Y8CbcdQeZ1jF4Ctun1Pds3IfjL1vtDebMBmNB/
AQJxMmfNJx/

qzF3t4eEyZMQGpqqtrfyLer809PPfUUxo8f32qNtzvO6nC7Ojds2IBt27Zh3759t3xztud4CiGw83Qxvv2tEMlnShEb3BuvRfvD0bpLh+puz
H5cxsXFxc0Nzfj0qVL6NmzZ4eer6OamprwyCOPYNq0aXj44YdvWn992I8dOxbPPvssysvLtT4R63avobbej+2xc+dOhISEwMHh5lbOunI8Ac
O6771p9/uun3Tc2NmLLli2IiYnRZpnYtWsX3n33XSQlJaFr166tbtOe45lVUoPHPz+KZ7/4DUVVdfh6zj1YNWVgh4MdaN/
xiYmJwcaNGwEA33zzDUaOHKnVW+sJITB79mz4+/tj/
vz5rW5TXFzc8jlAamoqVCq1n8Btec1jImJwaZNmyCEwJEjR2Btbd0y5KBtbf1lrgvH80/Xv//
ayMDdAEZLkmQjSZINgNF/LGvb7T5xbesLwARcG/
9pAFACYPef65YtWyY8PDyEj4+P2LFjR8unw9HR0aKoqEgIIUROTo4ICwsTnp6eYuLEiaK+vv7uP5JuhxkzZoi1a9fesKyoqEhER0e31BUUFC
r37y8efPBBcfHixZvqFOLaFQre3t7Cw8NDljo9PT2Fi4uLGDBggBgwYEDLlSftPZ6X6hrFkqR04fHadhEUv1ts+jlPNDUr1VZfa8dn4cKFwt
v1bjuH27dvF2rVrW96jq1evFgEBASIoKEgMGTJE/PTTT1qtUYhbv4bX16lSqcSzzz4rPDw8RL9/cSvv/
6q9TqFEOLKlSvC1tZWVFdXtyzTheM5ZcoU4ejoKExNTYWzs7P47LPPRHl5uRg5cqTw8vISo0aNEhUVFUJcy9ZQAJ+J/
8vaWQCy//
h6UrQjn9l+gGShUgn877dCJJ4owk85FZgS1hevPOAL225mWnl+th8gHcb2A6SfThVWY3FSOo5fqMYIv15Iem4o+rtYy10WkUFhuJPWVFxpwP
C13NTPG3oe54fpQ3LLuwwReRpjDcSaO05FYgPikdZ4trMNTLDosfDIC3Axt8EWka7zlGGqG4VId/
fHMSUxKOoKa+GeseD8Hm2YPVHuztbbBGZGx45k5q1dCsxGeH8vDx/mwM6GONF0Z54+nhnrAw09zs0hdffBEvv/
yyxvZPpI8Y7qQ2+8+WYOkPGcivuIrRAQ5YOD4AfWxbn9hERJrFYRm6a/nltZi94VesS8mFiYmETbMGI2F6qNaC/
XYN1v6UkJCA0NBQhIaGoqysTCu1EbXXr7/+iqCgEiS1EWSpG5/3Cujw1PjOYmJOuxqYzM+OZCNfx/
MQ+dOEl6I9MHMe91gZqrec4a2GpeFh4fDzs6upcGaQqFoV4M1TmIiXfTmm29ilLHwCwAFAohLi581k7MdzpjgkhsO2UAm/
vOAPFpXo8PNAZr0b7wd6q431g1CE/Px/jx4/
H6dOnb7stw510UWNjI8zNzU8BqAdwrxBC2dF9ccyd7sjZ4stYlZyFvRkl8HGwxOqpB2rInx31gFs3WCPSF3/
cS6A7gM4AugCo7ei+GO7ULpfqmvDPeveew+ch5WHYxxcpHgjBhoDM6yTy7dMGCBThx4gQkSYKbmxs+/
fRTWeshuhtz5swBgIUA3AG8UN3AG8A2BuR/fFcKc2qVQCX6cV4N1dmai62ojHhvTFS1G+sNFSg6/
b2bx5s9wlEKnFpk2b0LlzZwghvpQvpQkqROAnyVJGimE2N+R/XHMnW7pREE11h/
OQ9LJiwh1tcGS2EAE9jaMBl8ccycdxq6QpBnlVxrw7q6z2HqsEH6OlvjpnPRFAFAERcrCEBCt7wEp0LIbxb5VVAy/
9VggHK3N8NCUYMQN6c3YpETHc9VGTUoWNP+fjQGYpUvMq8cz9npg7wgvd2OCLiP7AT9n0zOGsckSvOoRl28+gs4mE3X+PwD/
G+Bl8sH/99dcIDAyEiYnJTX3YV6xYAS8vL/
j6+mL37t0yVUikWww7EQxIYdVVfLj3HL79rQh9bbis+mhGOVvbzRDMP369c033377523IWmRkZGDLli1IT0/
HxYsXERkZiXPnqFTJ/bIIePGcNdx9U1KfPpjLtakZCOkbw+8PNoHfxtmfA2+/
P39W12emJiIKVOmwNzcHO7/u7vDy8ueeLVdIpFsY7jpKCIE9GSV4a1sGCqvqMW/Wp+CEBS0fF9YWAhnzcHO740WQJAmLHwzAE+GuMDWS2aXJycl3/BhnZ2cFBS0fF9YWAhnzGGfdXHS/
EyPdT8OnBXAz37YUf5t2HJ+9zN5pg76iYmBhs2bIFDQ0NyMvLQ1ZWFgYPHix3WUSy45m7zE4XXUJ8UjqOna/
CABdrJEwPRXCfHnKXpXO+++47zJs3D2VlZRg3bhyCg4Oxe/duBAYGYtKkSQgICICpqSk+
+eQTXilDBEASQmhq3xrbsSGoqm3Eh8mZ2H6qGBKABWN88eigPjAxMY5LG+UWGhp60/XyRDpCLSHAM3ctU6oEvky9gA/
2ZKKmvhnPj/
TCzPvcYW3RWe7SiMiAMNy16Fh+JRYlpiNDcRnhHraIjwmEnyMbfBGR+jHctaD0cj3e3aeL0c3cFKunDsT4ICEjmV1KRNrHcNegxmYV/
vNTHv61LwtNSoGXRvvgiXtc0dWMh52INIspoyGHsuxKPE0ZRozhfhey
K7DkhwxU1DZGSlgfvDzaFz27m8tdGhHRDRju7fR74SUsTjqNc8U1GOhqg1ceCEWQCxt8EZFuYrjfRsWVBry/
JxNbfi1Az25mWBwTiEdCXNjgi4h0GsP9FpqVKnxx9AJ2nlbg1/
wqzLrPHS9EesOqCxt8EZHuY7i34mhuBRYnpeNscQ3u8bDFzheGwseBDb6ISH8w3K9TfKkO0b+84i6STF+HcwwJrpoUgup8jG3wRkd7htXsAGp
Iq9GSV4fqQXkucPx9j+7NyoK77+
+msEBgbCxMTKhpts5Ofnw8LCAsHBwQgDsbTTz8tY5VEusPoz9wPZJZi6Q8ZyCuvRVSAAz59fBD69mSDL13Tr18/
fPvtt5gzZ85N6zw9PXHixAkZqiLSXUYb7ucravHWk5YmFh103bHgyDPf7ssGXrvL395e7BCK9YnThXtd4bQgmOaMEFyqv4rVoPzx5nzvM
y8vIwcOBAWFlZYdymyZRg2bFir2yUkJCAhIQEdvxdj+fYMXLxUjvc5YmN
3w2OC+6MTZpQajrKwMtra26NSpE3Jzc5GVlVlQUPDw+5yyKSnUUGu0ol8M1vhdh28iIOZZfjscF98fJoX9h0M5O7NOqg7777DvPmzUNZWRnGjR
PDuWh5HI9nhrmgXkjvWDJBl9EZMT00tx/yalAfFI6MktqcvLDFI6MktqMCWsD/
42zB1e9pZyl0VEJDu9DPeL1XVYuvMmMtp9SwLmHBdY9PggBDpwCIaI6A96Fe4NzZupsSS3Ayp1noRICf4/0xtPDPdGlcye5SyMi0il6E+77zp
62HaVuywiIp2k8+GeV16LpPk40BmGTx7dcOn00MxzLuX3GUREek0nQ332oZmfHIwG2G58fyoOZqQneGOuPGfe6scEXEVE76Fy4CyHwwykFNv2
BwiDNeHeMHezb4IiJqN50K9zOKy4hPSsfRvEqMDnTA/565B4NcOZWci0hO6US4X7rahA/
3ZmLzkfOwtuiM5RP6YUoYG3wREXWUrOGuVAl8f7wIy3ecQfRkwb4oqXRvugR1c2+CIiuhuyhftvF6qwODEdSpWAZ69uiI8ZjMDebPBFRKQ
30pp6vLMzE//7rRD2luZ4faw/aw/YgY4wcSEV8EQEamL1sK9SanCf4+cx4d7zqG+WYk5wz0wb6c6Q3upvrxLA/EZFB0crp8k/
Z5YhedQi7ThcjxNUGu/4egdei/Rns1G6vvPIK/Pz8EBQUhIkTJ6K6urqVbaOL/Px1l/t/
Pz8EBQUhAkTJqC6urpl3YoVK+Dl5QVfX1/
s3r1bxiqJdIdGw72w6iqe2/gbpn12BHWNSaz
9MccRH74Iw5kluKlKB/
sfXE4ogId2bmROmT06NEwNb32l154eDgKCwsBAImJiZgyZQrMzc3h7u4OLy8vpKamylkqkU7Q2LhIbaMSI/
3s8ca4ADj3sNDU05ARWr9+PSZPngwAKCoqQnh4eMs6FxcXFBUVtfq4hIQEJCQkAADq6uo0XyiRjDQW7n8f5Q0TTkKiOxAZGYni4uKbli9fvh
zY1NcW0adPueP9xcXGIi4u76zJ9IHGwp3BTncqOTm5zfUbNmxAtm3bsG/
fvpbhPWdnZxQUFLRsU1hYCGdnZ43WSaQPeHE56YVdu3bh3Xff
jExMdiyZQsaGhqQl5eHrKwsDB48WMZKiXQDr0UkvTB37lw0NDQgKioKwLUPVdetW4fAwEBMmjQJAQEBMDU1xSeffIJOnXhnLiJJCKGpfWtsx

```
wPpyr74crLuGwAAAABJRU5ErkJggg==\n",
      "text/plain": [
       "<Figure size 432x288 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "sym.plot(sym.diff(expr));"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "x = sym.symbols('x')\n",
    "expr = -x ** 3 + 2\n",
    "\n",
    "sym.plot(expr, xlim=(-2, 2), ylim=(-10, 10));"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/latex": [
       "$$\\frac{d}{d x} \\left(- x^{3} + 2\\right)$$"
      ],
      "text/plain": [
       "d ⎛   3    ⎞\n",
       "──⎝- x  + 2⎠\n",
       "dx          "
      ]
     },
     "execution_count": 12,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "sym.Derivative(expr)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 13,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/latex": [
       "$$- 3 x^{2}$$"
      ],
      "text/plain": [
       "    2\n",
```

```
      "-3·x "
     ]
    },
    "execution_count": 13,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "sym.Derivative(expr).doit()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "sym.plot(sym.diff(expr));"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Now, let's generate a fake one-dimensional signal:"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 9,
  "metadata": {},
  "outputs": [
   {
    "ename": "NameError",
    "evalue": "name 'np' is not defined",
    "output_type": "error",
    "traceback": [
     "\u001b[0;31m---------------------------------------------------------------------------
\u001b[0m",
     "\u001b[0;31mNameError\u001b[0m                                 Traceback (most recent
call last)",
     "Cell \u001b[0;32mIn[9], line 1\u001b[0m\n\u001b[0;32m----> 1\u001b[0m ys
\u001b[38;5;241m=\u001b[39m \u001b[43mnp\u001b[49m\u001b[38;5;241m.
\u001b[39marray([\u001b[38;5;241m0\u001b[39m, \u001b[38;5;241m1\u001b[39m,
\u001b[38;5;241m0\u001b[39m, \u001b[38;5;241m1\u001b[39m, \u001b[38;5;241m2\u001b[39m,
\u001b[38;5;241m0\u001b[39m, \u001b[38;5;241m2\u001b[39m, \u001b[38;5;241m3\u001b[39m,
\u001b[38;5;241m2\u001b[39m, \u001b[38;5;241m1\u001b[39m, \u001b[38;5;241m2\u001b[39m,
\u001b[38;5;241m102\u001b[39m, \u001b[38;5;241m108\u001b[39m,  \u001b[38;5;241m95\u001b[39m,
\u001b[38;5;241m100\u001b[39m,  \u001b[38;5;241m98\u001b[39m,  \u001b[38;5;241m99\u001b[39m,
\u001b[38;5;241m104\u001b[39m, \u001b[38;5;241m110\u001b[39m, \u001b[38;5;241m103\u001b[39m,
\u001b[38;5;241m100\u001b[39m,  \u001b[38;5;241m96\u001b[39m, \u001b[38;5;241m102\u001b[39m,
\u001b[38;5;241m101\u001b[39m])\n\u001b[1;32m      3\u001b[0m fig,ax
\u001b[38;5;241m=\u001b[39m plt\u001b[38;5;241m.\u001b[39msubplots()\n\u001b[1;32m
4\u001b[0m ax\u001b[38;5;241m.\u001b[39mplot([i \u001b[38;5;28;01mfor\u001b[39;00m i
\u001b[38;5;129;01min\u001b[39;00m
\u001b[38;5;28mrange\u001b[39m(\u001b[38;5;28mlen\u001b[39m(ys))], ys);\n",
     "\u001b[0;31mNameError\u001b[0m: name 'np' is not defined"
    ]
   }
  ],
  "source": [
```

```
    "ys = np.array([0, 1, 0, 1, 2, 0, 2, 3, 2, 1, 2, 102, 108,  95, 100,  98,  99, 104, 110,
103, 100,  96, 102, 101])\n",
    "\n",
    "fig,ax = plt.subplots()\n",
    "ax.plot([i for i in range(len(ys))], ys);\n",
    "# check(1)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Next, let's look at small chunks of our fake signal:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "chunks = np.split(ys, len(ys)//2)\n",
    "print(chunks)\n",
    "check(2)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "**Question:** Which one of these chunks would you say is the most \"interesting\"?\n",
    "Between 2 and 102 is the most interesting because this is where it spikes up.\n",
    "\n",
    "**Question** If we always divide up the signal as we did above, will we always find
something \"interesting\"?\n",
    "Not necessarily, it could just be a straight line."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Convolutions\n",
    "\n",
    "Derivatives and convolutions are one technique to help us tackle the above problem. \n",
    "\n",
    "First, you'll need to generate windows into the signal. Write a function that can
generate windows with a user-supplied windowsize, and print them out.\n",
    "\n",
    "An example signal with 3 window sizes is shown below. Your output does not need to
replicate the formatting shown, but they should produce the same windows. E.g., given an input
signal of `[10,20,30]` and a `windowsize=2`, your function should return `[[10,20],
[20,30]]`."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### A windowsize of 1:\n",
    "\n",
    "```\n",
```

```
"signal:\n",
"        0   1   0   2   1   0   1 101 100  98 102 101\n",
" 0:     0\n",
" 1: ____    1\n",
" 2: ________    0\n",
" 3: ____________    2\n",
" 4: ________________    1\n",
" 5: ____________________    0\n",
" 6: ________________________    1\n",
" 7: ____________________________  101\n",
" 8: ________________________________  100\n",
" 9: ____________________________________   98\n",
"10: ________________________________________  102\n",
"11: ____________________________________________  101\n",
"\n",
"\t :::::::::::::::::::::::::::::::::::::::::::::::: \n",
"\n",
"i:    0 |  i + windowsize:     1  |  window:  [   0]\n",
"i:    1 |  i + windowsize:     2  |  window:  [   1]\n",
"i:    2 |  i + windowsize:     3  |  window:  [   0]\n",
"i:    3 |  i + windowsize:     4  |  window:  [   2]\n",
"i:    4 |  i + windowsize:     5  |  window:  [   1]\n",
"i:    5 |  i + windowsize:     6  |  window:  [   0]\n",
"i:    6 |  i + windowsize:     7  |  window:  [   1]\n",
"i:    7 |  i + windowsize:     8  |  window:  [ 101]\n",
"i:    8 |  i + windowsize:     9  |  window:  [ 100]\n",
"i:    9 |  i + windowsize:    10  |  window:  [  98]\n",
"i:   10 |  i + windowsize:    11  |  window:  [ 102]\n",
"i:   11 |  i + windowsize:    12  |  window:  [ 101]\n",
"```\n",
"\n",
"### A windowsize of 2:\n",
"\n",
"```\n",
"signal:\n",
"        0   1   0   2   1   0   1 101 100  98 102 101\n",
" 0:     0   1\n",
" 1: ____    1   0\n",
" 2: ________    0   2\n",
" 3: ____________    2   1\n",
" 4: ________________    1   0\n",
" 5: ____________________    0   1\n",
" 6: ________________________    1 101\n",
" 7: ____________________________  101 100\n",
" 8: ________________________________  100  98\n",
" 9: ____________________________________   98 102\n",
"10: ________________________________________  102 101\n",
"\n",
"\t :::::::::::::::::::::::::::::::::::::::::::::::: \n",
"\n",
"i:    0 |  i + windowsize:     2  |  window:  [   0,   1]\n",
"i:    1 |  i + windowsize:     3  |  window:  [   1,   0]\n",
"i:    2 |  i + windowsize:     4  |  window:  [   0,   2]\n",
"i:    3 |  i + windowsize:     5  |  window:  [   2,   1]\n",
"i:    4 |  i + windowsize:     6  |  window:  [   1,   0]\n",
"i:    5 |  i + windowsize:     7  |  window:  [   0,   1]\n",
"i:    6 |  i + windowsize:     8  |  window:  [   1, 101]\n",
"i:    7 |  i + windowsize:     9  |  window:  [ 101, 100]\n",
"i:    8 |  i + windowsize:    10  |  window:  [ 100,  98]\n",
"i:    9 |  i + windowsize:    11  |  window:  [  98, 102]\n",
"i:   10 |  i + windowsize:    12  |  window:  [ 102, 101]\n",
"```\n",
```

```
    "\n",
    "\n",
    "### A windowsize of 3\n",
    "\n",
    "```\n",
    "signal:\n",
    "        0   1   0   2   1   0   1 101 100  98 102 101\n",
    " 0:     0   1   0\n",
    " 1: ____    1   0   2\n",
    " 2: ________    0   2   1\n",
    " 3: ____________    2   1   0\n",
    " 4: ________________    1   0   1\n",
    " 5: ____________________    0   1 101\n",
    " 6: ________________________    1 101 100\n",
    " 7: ____________________________  101 100  98\n",
    " 8: ________________________________  100  98 102\n",
    " 9: ____________________________________   98 102 101\n",
    "\n",
    "\t :::::::::::::::::::::::::::::::::::::::::::::::: \n",
    "\n",
    "i:    0 |  i + windowsize:     3  |  window:  [   0,   1,   0]\n",
    "i:    1 |  i + windowsize:     4  |  window:  [   1,   0,   2]\n",
    "i:    2 |  i + windowsize:     5  |  window:  [   0,   2,   1]\n",
    "i:    3 |  i + windowsize:     6  |  window:  [   2,   1,   0]\n",
    "i:    4 |  i + windowsize:     7  |  window:  [   1,   0,   1]\n",
    "i:    5 |  i + windowsize:     8  |  window:  [   0,   1, 101]\n",
    "i:    6 |  i + windowsize:     9  |  window:  [   1, 101, 100]\n",
    "i:    7 |  i + windowsize:    10  |  window:  [ 101, 100,  98]\n",
    "i:    8 |  i + windowsize:    11  |  window:  [ 100,  98, 102]\n",
    "i:    9 |  i + windowsize:    12  |  window:  [  98, 102, 101]\n",
    "```"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "The below resources may be helpful::\n",
    "\n",
    "## List Comprehensions\n",
    "\n",
    "https://www.pythonlikeyoumeanit.com/Module2_EssentialsOfPython/
Generators_and_Comprehensions.html#List-&-Tuple-Comprehensions\n",
    "    \n",
    "## Numpy indexing with slices\n",
    "\n",
    "http://www.pythonlikeyoumeanit.com/Module3_IntroducingNumpy/
AccessingDataAlongMultipleDimensions.html#Slice-Indexing"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Formatting numbers in python\n",
    "\n",
    "https://pyformat.info/#number\n",
    "\n",
    "**input:** `'{:4d}'.format(42)`\n",
    "\n",
    "**output:** `_` `_` `4` `2`\n",
    "\n",
```

```
   "**input:**  `'{:06.2f}'.format(3.141592653589793)`\n",
   "\n",
   "**output:**  `003.14`"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## String concatenation\n",
   "\n",
   "```python\n",
   ">>> print('a' + 'b' + 'c')\n",
   "abc\n",
   ">>> print(''.join(['a', 'b', 'c']))\n",
   "abc\n",
   ">>> print(''.join(['a', 'b', 'c']))      \n",
   "a,b,c\n",
   "```"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "def make_windows(sequence, windowsize):\n",
   "    window = []\n",
   "    for i in range(len(sequence) - windowsize + 1):\n",
   "        new_list = sequence[i:i + windowsize]\n",
   "        window.append(new_list)\n",
   "    print(window)\n",
   "\n",
   "make_windows([10,20,30], 2)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "series = [0, 1, 0, 2, 1, 0, 1, 101, 100, 98, 102, 101]\n",
   "\n",
   "\n",
   "make_windows(sequence=series, windowsize=1)\n",
   "make_windows(sequence=series, windowsize=2)\n",
   "make_windows(sequence=series, windowsize=3)\n",
   "\n",
   "check(3)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## When you are done:\n",
   "\n",
   "Generate some example outputs in this notebook.\n",
   "\n",
   "1. Double-check that you filled in your name at the top of the notebook!\n",
```

```
    "2. Click `File` -> `Export Notebook As` -> `PDF`\n",
    "3. Email the PDF to `YOURTEAMNAME@beaver.works` "
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.10.12"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 4
}
```