```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# COMP2200/COMP6200 Week 1 Practical"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "The goal of this week's practical is to get you started using Python, Jupyter Notebooks, and Git, three tools that you will use through the semester in your work.  \n",
    "\n",
    "**Python** is our language of choice in COMP2200/COMP6200. If you do not have any experience of using Python, you need to learn basic Python coding.\n",
    "\n",
    "You are looking at a **Jupyter Notebook**, it is a document that mixes text, code and the output of the code. A lot of your work will be creating notebooks like this to present your analysis.  \n",
    "\n",
    "**Git** is a distributed version control system (DVCS), you will use it to keep track of your work and ensure that you have a backup copy of what you are doing. You should have checked this notebook out of **Github** using Git. Your task this week is to complete some programming work in this worksheet and commit your changes to your own Bitbucket repository."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Python Basics"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Your task this week is to complete some basic programming tasks with Python in this worksheet.  There are questions below with a space for you to write code to achieve the given outcomes. Write the code, test it, and when you are done, submit your work as described at the end of the notebook. \n",
    "\n",
    "The tasks aren't meant to be complicated Python problems, just some simple tasks to get you started with this process.  "
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Examples - basics\n",
    "- How to define a variable\n",
    "- How to perform some basic arithmatic operation\n",
    "- How to use a constant"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 77,
   "metadata": {},
   "outputs": [
    {
```

```
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "The area of the circle is:  78.53981633974483\n"
     ]
    }
   ],
   "source": [
    "# Let's show an example of calculating the area of a circle\n",
    "\n",
    "# Define the radius\n",
    "radius = 5\n",
    "\n",
    "# Then we can calculate the area. \n",
    "# We need to make use of a constant PI (around 3.1415926), which is defined in the
module math\n",
    "import math\n",
    "\n",
    "area = math.pi * radius * radius\n",
    "\n",
    "print(\"The area of the circle is: \", area)\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "- How to use a built-in function\n",
    "\n",
    "To calculate the the squared radius \"raius*radius\", we could also use the built-in
function pow() in the math module."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 75,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Task 1: Re-calculate the area of the circle above with using the funciton pow(). \n",
    "# Read the manual about how to call the function and write your code below and print the
result.\n",
    "\n",
    "# write your code here: \n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Examples - control structures\n",
    "- Braching strucutre: if-else \n",
    "- Loop structure: for, while\n",
    "- Boolean values: True/False"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "5\n",
```

```
      "It's an odd number.\n"
     ]
    }
   ],
   "source": [
    "# Let's test if a given number is even or odd, we need to use the if-else structure\n",
    "\n",
    "# The built-in input() function can let you input a string. \n",
    "# Let's input an integer number and test if it's even or odd.\n",
    "\n",
    "x = input()\n",
    "\n",
    "# Now x is a string. We need to convert it to an integer.\n",
    "x = int(x)\n",
    "\n",
    "# Now can divid x by 2 to check its parity.\n",
    "if x%2 == 0:\n",
    "    print(\"It's an even number.\")\n",
    "else:\n",
    "    print(\"It's an odd number.\")\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 39,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "8\n",
      "It's not a prime number.\n"
     ]
    }
   ],
   "source": [
    "# Now we go further to check if the given number is a prime number of not.\n",
    "# To be a prime number, it cannot be divided by the number less than it. So, simply use the loop structure to test.\n",
    "\n",
    "\n",
    "x = input()\n",
    "\n",
    "# Now x is a string. We need to convert it to an integer.\n",
    "x = int(x)\n",
    "\n",
    "# Now we use a loop to try to divide the number by the numbers less than it.\n",
    "# 0 and 1 are not prime.\n",
    "if x < 2:\n",
    "    print(\"It's not a prime number.\")\n",
    "else:\n",
    "    # 2 is a prime number\n",
    "    if x == 2:\n",
    "        print(\"It's a prime nubmer.\")\n",
    "    else:\n",
    "        \n",
    "        # Use a loop to test its factor\n",
    "        # We use the built-in function range(a, b) to generate all the integer numbers between a (included) and b (excluded)\n",
    "        is_prime = True\n",
    "        for i in range(2, x):\n",
    "            # To test if i is a factor for x\n",
    "            if x%i == 0:\n",
    "                is_prime = False\n",
    "                break\n",
```

```
    "        if is_prime:\n",
    "            print(\"It's a prime nubmer.\")\n",
    "        else:\n",
    "            print(\"It's not a prime number.\")\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Example - defining a function\n",
    "- How to define your own function\n",
    "\n",
    "Python is a dynamically typed language so we don't need to declare the type of a variable or declare the return type of a function (although Python 3 introduced optional [type hints](https://docs.python.org/3/library/typing.html)).  Apart from that the idea of writing a function in Python is the same as in Processing or (methods in) Java.\n",
    "\n",
    "Write a function that takes a single string argument and returns the number of words in the string using the code you wrote above to count words."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 42,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "4\n",
      "It's not a prime number.\n"
     ]
    }
   ],
   "source": [
    "# Now, to make things easier for the number primality test, we can make a self-defined function. \n",
    "# We can just simply resue the code above\n",
    "\n",
    "def is_prime(x):\n",
    "    # Now we use a loop to try to divide the number by the numbers less than it.\n",
    "    # 0 and 1 are not prime.\n",
    "    if x < 2:\n",
    "        return False\n",
    "    else:\n",
    "        # 2 is a prime number\n",
    "        if x == 2:\n",
    "            return True\n",
    "        else:\n",
    "            # Use a loop to test its factor\n",
    "            # We use the built-in function range(a, b) to generate all the integer numbers between a (included) and b (excluded)\n",
    "            for i in range(2, x):\n",
    "                # To test if i is a factor for x\n",
    "                if x%i == 0:\n",
    "                    return False\n",
    "            return True\n",
    "\n",
    "# Now, let's try to call this function to test a number\n",
    "\n",
    "x = input()\n",
    "\n",
    "# Now x is a string. We need to convert it to an integer.\n",
    "x = int(x)\n",
```

```
    "\n",
    "# Now we can directly call the function we defined before.\n",
    "if is_prime(x):\n",
    "    print(\"It's a prime nubmer.\")\n",
    "else:\n",
    "    print(\"It's not a prime number.\")\n",
    "        "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 74,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Task 2: Now we want to have a smarter way to perform the primality test. \n",
    "# We can improve the testing process with less computation by an optimisation strategy.\n",
    "# Strategy: You just need to test the factors less than the sqared root of number, i.e., sqrt(x). \n",
    "# The reason is that one of the factors (if any) of the number x must not be greater than sqrt(x).\n",
    "# Your task is to define a new version of the primality testing function is_prime_faster(x).\n",
    "# write your code here: \n",
    "\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## String Manipulation\n",
    "\n",
    "The next cell defines three strings that you will use for the following group of questions. Note that the first uses single quotes, the second uses double quotes and the third uses three double quotes since it includes newline characters.  These are all valid ways of writing strings in Python and are equivalent."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "#### title = 'Data Science'\n",
    "\n",
    "code = \"COMP2200/COMP6200\"\n",
    "\n",
    "description = \"\"\"This unit introduces students to the fundamental techniques and \n",
    "tools of data science, such as the graphical display of data, \n",
    "predictive models, evaluation methodologies, regression, \n",
    "classification and clustering. The unit provides practical \n",
    "experience applying these methods using industry-standard \n",
    "software tools to real-world data sets. Students who have \n",
    "completed this unit will be able to identify which data \n",
    "science methods are most appropriate for a real-world data \n",
    "set, apply these methods to the data set, and interpret the \n",
    "results of the analysis they have performed. \"\"\""
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Task 3.\n",
    "\n",
```

```
    "Write code to print the length of these strings, i.e., how many words they contain."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {
    "scrolled": true
   },
   "outputs": [],
   "source": [
    "#write your code here: print the lenght of these strings\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Write code to create a new string in a variable 'summary' that contains the code, title
and the first 20 characters of the description, with a ':' character between each one (ie
'COMP2200/COMP6200:Data Science:This unit...'"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {},
   "outputs": [],
   "source": [
    "# write your code here: create a new string summary and then print it\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Write code to find the number of words in the description.  Hint, this is easy in Python
since strings support the [split method]
(https://docs.python.org/3.6/library/stdtypes.html#str.split) that returns a list of strings
after splitting on whitespace (or another character if you wish).   Try split on the string,
then find out how many strings are in the resulting list."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {
    "scrolled": false
   },
   "outputs": [],
   "source": [
    "#write your code here\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Data generation and plotting\n",
    "\n",
    "Just to give you a taste of some of the capabilities of Jupyter notebooks and Python we
will look at a very simple example of data handling and plotting.  First you need to import
some libraries that will allow you to do this: `matplotlib` does the plotting. You also need
to use the random module to generate random numbers.\n",
    "\n",
    "- How to use an array\n",
```

```
    "- the random module\n",
    "- Gaussian distribution\n",
    "- Pyhton plotting"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 68,
   "metadata": {},
   "outputs": [],
   "source": [
    "import random"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "The Gaussian or Normal disbribution is often used in data science for data distribution
modelling and data analytics. The Python random module has the function gauss() that draws a
random number from a Gaussian distribition. Two parameter $\\mu$ and $\\sigma$ should be
given to determine a Gaussian distribution."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 63,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "1.4774785157818007\n",
      "[3.9382744715212734, 1.1459855794223535, -0.33466904691661964, -2.65926180162888,
2.9631745411120263, -0.6456969367532899, 8.930807213758179, 2.525706465297843,
0.7767386869308457, -1.2996036625176148, 0.44124018300775386, -1.0314096126588825,
2.9223712485567717, -5.893255572542497, 7.115349409818994, -0.9243668316855825,
2.989435913751122, 1.4702238712360973, -3.5410235744254823, 4.990670636212493,
-0.35527490911262527, -5.197406580139417, 2.528153548718955, 5.877725800821714,
-1.4889979723238103, -4.693442395992448, -0.3835970018426413, -4.556982506163179,
2.5221635701088205, 2.370544587654644, -8.665829138964614, -5.077888020374325,
-1.5889672074182757, 0.818891844821283, -4.563944028398502, -1.9591897227796526,
-7.238939266325599, 0.5434877603264603, -5.240810535759312, -0.01734238315914804,
5.34450968543907, 0.8874032233128535, 7.440994476983899, -1.3050866622182293,
9.055267121297113, -2.033934302376644, -1.4912126361437519, -3.8221673084011822,
-1.7139296107884805, 4.3375800592689835, 2.3722379688401922, -1.1983636435397202,
-4.976112082641999, 3.425614235276147, 0.8735326177399017, 2.4016845949890575,
-6.2198807760159855, 3.0259717916059605, 1.3147490083959141, -13.551461001810143,
-4.715367945858329, -5.5810910864922665, 4.092506047321316, -2.7192234513793383,
-0.9967688792618644, 5.668792238977426, 0.8461921009117521, -9.134174694593437,
8.293181364798722, -6.541375554535309, -0.08194373969300434, 0.13817092050570395,
-2.2186526104123403, 2.1582572697639253, 3.8797184534180107, -2.6599351002761358,
-0.5251508773351957, -6.901929948305811, 1.6201446417100887, -2.3528160189229195,
0.22157926862140384, 5.96686394951154, 3.757020067943821, 5.8874719339263315,
3.0484317887211203, 1.876336109032903, -6.403254243341153, -4.076765876590345,
3.7412336776189505, -8.018584627352721, 4.1465510547247595, 6.88035724961242,
-8.639422608166328, 2.330458731837612, 0.7379419471803018, -1.869918519348919,
2.579403717564668, -2.9734648767330354, 3.450209943326322, 0.3938897876272362]\n"
     ]
    }
   ],
   "source": [
    "# Random draw a number from a Gauassian distribution\n",
    "mu=0\n",
    "sigma=5\n",
```

```
    "x = random.gauss(mu, sigma)\n",
    "print(x)\n",
    "\n",
    "# Use a loop to draw a set of 100 such random numbers\n",
    "# x is an array for the generated numbers\n",
    "x=[]\n",
    "for i in range(100):\n",
    "    x.append(random.gauss(mu, sigma))\n",
    "\n",
    "print(x)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "In terms of the probability density function, we can calculate the probability density
for the genrated random number. The probability density function is defined as
$p(x)=\\frac{1}{\\sqrt{2\\pi\\sigma^2}}e^{-\\frac{(x-\\mu)^2}{2\\sigma^2}}$."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 67,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "0.07978845608028654\n",
      "[0.05850889050191842, 0.07772004273584945, 0.07960992459532215, 0.06926539056160268,
0.06693826429094868, 0.07912590695148575, 0.01618674153784448, 0.07023136248894538,
0.07883147717105986, 0.0771382633644682, 0.07947837526083304, 0.0781087992975795,
0.0672605416297029, 0.039835943528841414, 0.02898605606091475, 0.07843652920286388,
0.06672931033428986, 0.07641260494174981, 0.062091051122672755, 0.048484442115589006,
0.07958729170222176, 0.046484463112972295, 0.0702139933070419, 0.039981850816117735,
0.07632774740527708, 0.0513574907900012, 0.07955398916365398, 0.05267057982712904,
0.07025648744037069, 0.07130662215824762, 0.017768850280207072, 0.047640340813350605,
0.07585946605046656, 0.07872550299814564, 0.052603735348708616, 0.07389242696768734,
0.027975636301810654, 0.07931848903020355, 0.04606516073709386, 0.07978797614099577,
0.04506487678254191, 0.07854165678567616, 0.02636430092101426, 0.0771162334835523,
0.01547802971367457, 0.07345265569120983, 0.07631767255630403, 0.0595728286315238,
0.07523584195340964, 0.05476694496730626, 0.07129516932533704, 0.07752940819013271,
0.048625350209424524, 0.06309756476670764, 0.07858003414521292, 0.07109500317217872,
0.036805249924304125, 0.06643664107293751, 0.07707720132962893, 0.002026959582432166,
0.051146033146484096, 0.0427944011179218, 0.05707731306719348, 0.06882006258517061,
0.07821863116531705, 0.041958235234819924, 0.07865396267606886, 0.015040038619681088,
0.02016284774368332, 0.033905829133115524, 0.07977774156680117, 0.07975799674303259,
0.07230768706011077, 0.0726909828479196, 0.05904704669661706, 0.06926042938669831,
0.07934958083406775, 0.030773258831961196, 0.07570782019109804, 0.07142614429827192,
0.07971014646390187, 0.039146441546799506, 0.06016404035391825, 0.03989026534799151,
0.06625560770802802, 0.07436356390804058, 0.03514019160091845, 0.05722428854053885,
0.060306641868017236, 0.022052475095513443, 0.05657126214732264, 0.030956794783501277,
0.017931991876810528, 0.071575874962714, 0.07892418175334058, 0.07439932937066569,
0.06984736508953386, 0.06685652904571739, 0.06288450942242581, 0.07954125755864823]\n"
     ]
    }
   ],
   "source": [
    "# We can implement the probabiity density function as follows:\n",
    "def pdf(mu, sigma, x):\n",
    "    return math.exp(-0.5*math.pow(x-mu, 2)/pow(sigma, 2))/math.sqrt(2*math.pi*pow(sigma,
2))\n",
    "\n",
    "\n",
```

```
    "# The probability density example\n",
    "mu=0\n",
    "sigma=5\n",
    "print(pdf(mu, sigma, 0))\n",
    "\n",
    "\n",
    "# Now we can calculate the probablity density for the random numbers generated above.
\n",
    "# You need to ensure that the distribution is the same, i.e., the same mu and sigma.\n",
    "y=[]\n",
    "for i in x:\n",
    "    y.append(pdf(mu, sigma, i))\n",
    "\n",
    "print(y)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Now we will create a simple plot."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 69,
   "metadata": {},
   "outputs": [],
   "source": [
    "import matplotlib.pyplot as plt\n",
    "%matplotlib inline"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 73,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "image/png":
```

"iVBORw0KGgoAAAANSUhEUgAAAkUAAAHFCAYAAAD8Jo2EAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24z
LjUuMywgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/NK7nSAAAACXBIWXMAAA9hAAAPYQGoP6dpAABksElEQVR4nO3de1yTd
f8/8NdAYEo4OSgbHhDNEkJFIBUzVJUjDx15+FWy8puTFMhvymaIZqSVmqmSOWhzDK7U0uL28Qz6rw94QHhLjUU0y0CEg
ziNK7fH/y2nBuwwcY29no+Hns85LPPde19udN7n6NIEAQBRERERHbOwdIBEBEBEREVkDJkVEREREYFJEREREBIBJERERERE
AJkVEREREAJgUEREREQFgUkREREQEgEkREREREQAmRUREREQAmBTR//fpp59CJBJpbs2aNYNMJsP48eNx9epVi8W1ePFi
iEQiiz3+g44cOL5P5P5LL5Tr3v/DCC3jooYcsEBlw48YNiEQivPfeexZ5fGMcPHgQoaGhcHV1hUgkwrfffmvpkOqUnZ2NW
bNmwd/fH66urhCLxejYsSMmTZqEw4cPwxY2B1C/C/Rj799NNGf+wHP2PEYjGkUikGDRqExMRE5Obm6hxTn/d/SUkJFi9ejC
NHjhh1nL7H6tixI55++mmjzlOXF5SWluLEiRNYtmwZDh8+jP/9EmvWrNF0gkwuLFi036eGScZpYOgKzLli1b0LVrV5SWluLEiRNYtmwZDh8+jP/
9739wd3e3dHhW5Y033kBaWpqlw7A5giDgueeewyOPPIIeeH5ITo6GsHBwXBxBxccJEeHl5ITo6GsHBwXBxccG1
a9fwzTff4Mknn8SBAwfw1FNPWTrUWslkMsjlcnTu3NliMag/YyoqKpCbxDbm4vjx49jxjx+997jxYoVeO+997Bjxw4MHjxYU/fll1/Gs
GHDjDp/SUkJEhISAABBPPPGEwcfV57Hq48svv0RGRgbmzJmjc59cLke7du3MHgPVjEkRaQkMDEDERoaCiA6g8UlUqF+Ph4fP
vtt5g6daqFo7Mew4YNw759+7B3715ERUZZOpxpGvGVVFRoWlNrI87d+7B3715ERUZOpxGVVFRoWlNrI87d+6goKAAo0ePrjOJKCkpQYsWLer1OKZy/fp1TJgwAY8
99hgOHDiAli1bau5w4bOHAgXnrpJRw5csWcQmfjS4uLigt5S4uLigT58Fo3h/s8YABg7dixiYmLjfr1LfhDjAAoF27dmZ
EtSvscZ4rLpY+rkhdp9RHdQfXr/99rGi/jOjoJiEKZqpEFCqmFjK+Fo3h/s8YABg7dixiYmLQv39/jBkzBlevXoF27dmZ
99hgOHDiAli1bau4bOHAgXnrpJRw5csWcQmfjS4uLigT58+Fo3h/s8YABg7dixiYmLQv39/jBkzBlevXoF27dmZP
EtSvscZ4rLpY+rkhdp9RHdQfXr/99pumrLS0FK+//jqCgoIgkUjg4eGBsLAwfPfddzrHi0QiREdHY2tri9AHt44
OjRA99//7lO3R9++AFBQUFwcXGBn59fjd1ApaWliIuLg5+fH5ydndG1a1dEREREREQjCsyrwVoJP7vqA1o3hz+/vvv03+/V
6ax/70008lXSG9evXC2bNnDf5/eeGFFxAQEBgQRuXQeF4vBzeV
6ax/700081XSG9evXC2bNnDf5/eeGFFxAQEIC4uDioVKpa69bUJN6xY0e88MILmr/V3QuHDh3CtGnT4OnpiZYtW2LKlCk
oLi6GUqnEc889BHhatWkEmk2Hu3LmoqKjQOW9VVRWLVuGDh06QCwWIzQ0FAcPHtSpd/XqVcQXh45/XqVUycOBFt2rSBi4sL/P39sX79
eq066u7Czz//HK+//jpatm2raSEpKSnB3Llz4efnB7FYDA8PPD4SGhmL79u01/l8sXrxY88Uzb948iEQC
fPn8eyzz8Ld3V3TomHJ53zVqlUoKSlBUlKJ0vyeeeAI9evTQ/H3t2jVMnT0QYsWLdC2bVVtERUXh8uXLWsepn/
MbN27o/X+vwsoPT0dTz/9tb58vHxwYgRI/Drr79q6z/9G7969IZFIC0tKJFC3Tq1cAkvvvui5n593WeGxqqOafv27Vi
4cCF8fHzQsmVLDB48GD/99G+16eHIebQh2HwrJDHHQPSDmhXTtfPzxvwV+m4hdCCHmVLUnrOgLa2rvcfSGNmZdLFIuUiBlOT8c
jRto3bo1ACAhIUHTVad+n9X9X2G73tqYsw3nGNjUJFXiWzDSUnuQNodRWrNnvnhscbIJfvSS4oxkpkFRtQ6UqjBKmlgyxZA1FZ62oj4FZ6LfSuoKEQX2xFs
JFwd3eHWCxGUFAQPvvsM72PY47nxt6wpYhYhqZ2dDkQB45JFHNGVIZWUoKCjA3LlzbtW5SX1+PAgQMYM2YmzZgilTpm
id44cffsCZM2ewZMkSPPTQ1i5ciVGjx6Nn376CZ06dQJQJQPcZk5MiRCAsLw1dffQWVSoWVK1dqJWNAddfLqFGjcPDgQcT
FxSE8PByXLl1CfHw85HI55HI5XFxcNPUvUtS4c4EEkJqJXgWVpiYBjQZAgJXv8WcCQ4zw5ZTuDkzJgnlCdeFgjvjEdD2LbBWjAMl8f7h9MUy8NlKbfDV8sHSmVUB8+gNHUnQW4N9BBnTVqvq6u+A+/Sit8f3lf8

NG/evM7/F0dHRyQmJmLkyJH47LPPtL54Gurll1/GmDFj8NVXXyE9PR0LFixAZWUlfvrpJ4wZMwavvPIKDhw4gBUrVsDHx
wexsbFax69btw6+vr5Ys2YNqqqqsHLlSgwfPhxHjx5FWFgYACAzMxN9+/bVfBlJpVL8+OOPmDVrFvLy8hAfH691zri4OI
SFhSE5ORkODg5o06YNYmNj8fnnn+Ptt99Gz549UVxcjIyMDOTn59d6bT169MCYMWPw2muvYeLEiVrPFwCMGTMG48ePR3R
0NIqLiy3+nKempkImk2m1btTlzp078PT0xDvvvIPWrVujoKAAn332GXr37o309HSjuwuLi4sxZMgQ+Pn5Yf369fD29oZS
qcThw4dx7949ANVdL+PGjcO4ceeOwePFiiMVi3Lx5E4cOHTJprAsWLEC/fv2wceNGFBUVd68eYiKikJWVhYcHR2Nuq77R
UZGwtHREceOHauxzo0bNzBixAiEh4dj8+bNaNWqFW7fvo19+/ahvLwcMpkM+/btw7Bhw/DSSy/h5ZdBgBNoqT24GusNh
cuXMCcOXOwePFiSKVSfPHFF5g9ezbKy8sxd+5co64xKSkJr7zyCq5fv47fv47du3fXWf+nn35C37590aZNG6xduaxaenp7Ytm0
bXnjhBfz222944403tOqb67mxKwKRIAhbtmwRAAinTp0SKioqhHv37gn79u0TpFKpRKioqhHv37gn79u0TpFKpMGDAAKGioqLGYysrK4WKigrhpZde
Enr27Kl1HwDB29tbKCoq0pQplUrBwcFBSExM1JT17t1b8PHxEf766y9NWVFRkeDh4SHc/zLDt2+fAEBYuXKl1uPs2LFDA
CB8/PHHmJfX1+hefPmwq+//qopu3DhggBAkMlkQnFxsab822+/FQAIe/bsqfX/6fDhwwIA4d///rcgCILQv39/oV27dp
q4n3/+ecHV1VXn/yA+Pl7nxL6vsLzzz+v+Vv9HLz22mta9UaNGiUUAEFatWqVVHhQUJAHB2v+zs7OFgdU+P84ePBgTdn
QoUOFdu3aCYWFhVrnnDlzpiAWi4WCggKt6x0wYIBO/IGBgcKoUaN0yuuijvPdd9/VKo+PjdAoAWi4WCggKt6x0wYIBO/IGBgcKoUaN0yuuijvPdd9/VKo+PjxcACG+99ZZWuaWfc7FYLPTp
00enXKVSCRUVFZqbSqWq8RyV7ZVCeXm50KVLFyEmJkJzTrn7Os7Os7Ozteqr/98PHz4sCIIgnD17VgAgfPvttzU+xnvvvScAE
O7evVtjHfX//ZYtW4yOVR1TGSkVv2vv/5aACDI5fIazykIf1/rmTNnaqzj7e0t+Pv7a/5WvybvUvvnmGwGAcOHChRrP8f
vvv9f4nqvpNabvsQSh+vUkEol0Hm/IkCFCy5YtNa8nQ59HQRCEESNGCL6+vnpjcfDu8ePHCy4uLkJOTo5WvEhDwstfTrT
QPNcNfW7ob+w+Iy19+vSBk5MT3NzcMGzYMLi7u7u/y3SuvhTNa8nQ59HQRCEESNGCL6+vnpjfzDu8ePHCy4uLkJOTo5WveHDhwstWrT
NzfN397e3mjTpg1u3rwJoPpPx8JkzZZBmzBiIxWJNPTc3N53x3Oupfvfvfvd3OwHAP/7xD7i6uup0EwUFBaFt27aav/39/QFUN
2PfP1ZFXa6OyVArVq7/+ig8++MCo42rz4GwXdWwjRozQKdcXb03/j8eOHYNKpSOHjyI0aNHo0WLFqisrNNcTcIiZ
MjUVpailOnTmmdc+zYsTqP06tXL/znP//B//B/PnzceTIEfz111l/1vubaHsvannO1MWPGwMnJSXObXNWuW5r7KykosX74cAQE
BcHZ2RrNmzeDs7IyrV6/qfgFdugcxvb6yJzvmzZuH5RkZGzvk2HZ5ARkZGm6tR5/PHHAQDPPfccv76a9y+fdugcxsb6zPPPKP1d/fu
3QHU///xfkIdM/hRdQUlKiMwPWnM+NvWBSRFq2bt2KM2fOJEcePsdP/jXv/jXv/LXp27c
dffBG1paU65/T09NQpc3Fx0XyJ/vHHH4iqqoJUKtWp92BZFn4tPqnwCRbt/9bC/YPvdsd2cJPQHDjJN0zqR5BHEOpm/LXp2Gsl
vRo0ahXfeeQd//PGHUcPf5/HH7ow2hIeUGNaGVHMaHzVu+dxIZfEz7dw1rC+Dkp5QPAgqLb7GnP4B4s9VcrH9mjpMzzWN
zcO3336LQYMGwcPDA6NGjWrwEg4PPPaaln/M0HTro/VJ5//33cebMGZw5c0bnvtjYWCxxxAijRo3C3r178d///hdnzpBj
x496pU8SiQSHD16FEFBQViwYAEe+wx+Pj4ID4+XcMCAAfj2229RWVmJWNRVmOmofWF27dggMDKx1jFd9Yn3uw/azuumoXl
xcXIz8/Hz4+PjUWKdz5844cOAA2rRpgxkzB587o3Lmz0T9K9L2ea1Lb5lJtXcWmkJ+frzdW9f9f/Rg49vrufGnnBMEWn
x9/fXjJ0YNGgQVCoVNm7ciGA++++QbPPvssAGDtbm3w8/PDjh07tAYJlpWVEaxsEsx3d3dIRKJoFQSdeelxW6Ct0xOVlZX4X4/fff
tb4kBUGAUqnU/FpuTImJiQgMDMTy5cv13u/i4qg1/8ZcH6g1/T86OzvjeoYcegspOTEXhvmNTmoTZ0YPMmqFC7Tsd8y1C7dsmaTcE
NXVlRUJCQlISEjAb7/9pmk1ioqKwv+/9796x//gY1In6OR8zZAjWr1+Ps2fPZao8rqm9amla+7Zt2zBlyhSd1oReXXh5atWql+V
vdovfg6+PBpBQAunXrhq+++gqCIODSpUv49NPPsWTJEjdMTFtPdqFoJvzE4oOBEdzbUjCe
rb6zm9sMPP0C1UtU5jT48PBzh4eF4QvQ4e/YsPvzwZm6zjbKP1P0KsYP368QY91zNpMxF6eHoy5HNIt+R3KtzmZK2HKeQSDOvU8U3712l
BwDg5eXVoPOTLrYUUa1WrlwJd3d3d3vPXWW6iqqgJQ/YHi7Oys9s9V/+/t+7qT5ueTrwsMw5NR2vqgh21R3Dr5c/c0hHEoom0KN5d7R+td+7dw979+Vqquewrl2t
zat8p07d6K4uNgi68R07doVL774IGn6yKwT7Jta0zszhmjBMgPPXNvkAWrs9QNyEqxfiwwP...

gARCLtN6cgCDplddW/v/zIkSNYtmwZkpKS0Lt3b1y7dg2zZ8+GTCbDokWL9J4zLi4OsbGxmr+LiorQvn37el0PEdGD9mU
oMH3beVhqrMJ/Mqp/aK47fB3NHICRPXyQOLYHnJtZfK4NkVWxWFLk5eUFR0dHnVah3NxcndYgNalUqrd+s2bN4OnpCQBY
tGgRJk+ejJdffhkA0K1bNxQXF+OVV17BwoUL4eCg+yHg4uICFxcXU1wWERGA6q6y09kFUBaVYun3VyyWED2osgrYmX4HO
9Pv4F8D/BAXGWDpkIishsWSImdnZ4SEhCA1NRWjR4/WlKempmLkyJF6jwkLC8PevXu1yvbv34/Q0FA4OTkBAEpKSnQSH0
dHRwiCAAuOKSciO1FeWYUFuy4j5bICJRUqS4dTq4+OZQMAEyOi/8+ibaexsbHYuHEjNm/ejKysLMTExCAnJwfR0dEAqru
1pkyZoqkfHR2NmzdvIjY2FllZWdi8eTM2bdqEuXPnaupERUVhw4YN+Oqrr5CdnY3U1FQsWrQIzzzzDBwdHRv9GonIfiSm
ZOLRRf/BN+d/tfqESO3jY9kor6yydBhEVsGiY4rGjRuH/Px8LFmyBAqFAoGBgUhJSYGvry8AQKFQICcnR1Pfz88PKSkpi
ImJwfr16+Hj44O1a9dq1igCgDfffBMikQhvvvkmbt++jdatWyMqKgrLli1r9OsjIvuRmJKpaXmxJQKAN765iHGPd0AvPw
8Owia7ZtF1iqwV1ykiIkOpF1+csvm0ycYNiQC8HN4RT3aVQllUioI/y+Dh6ow2bmJABOQWlaKguBy/3v0L/z77K/4sqzT
J48okYsRHBWBYoMwk5yNqbA39/mZSpAeTIiKqS/XYoUtIyVCipLzhXWXNnRzwiLcbnu4uw/N9/QyeGaaqEnDql3xsO3UT
x37+HcUNiEXdRrRhUjATI7JJDf3+tviUfCIiW5OYkomPj2U3uGXIw9UJo4PaYnCAtN5dV44OIvR72Av9HvbSzHjbn6nEl
hM3jD6X+noW7L6MJ7t6c8o+2R0mRURERmjo2CFPV2e8OcIfUklzk4/hcXQQIayzJ8I6e8LZUVTvOAuKK9An8SCWjw5kix
HZFf4MICIyUHllFT5Ja9hg6mWjAzE6uB3CzLyydFxkAJIm9oSrc/1m3RYUl2P6tvPcJoTsCluKiIgMUF5ZhRlfnENVPfv
MXJ0d8f5zPRq15SWyuw+GBspw6no+jl/9HZfvFOKvchXO5dw16HgBwMLdGexKI7vBgdZ6cKA1Ed0vMSUTn6Rl1yshEgEY
0V2GD8b3tIrp7qoqAf1XHIKysNTgMVEerk5YProbu9LI6jX0+5upPxFRLZb9UD2GyNiEqIWzIxZG+uOnt4dj3cRgq0iIg
OpxR/FRxq1gXVBcwa40sgtMioiIavD9hTv1GkMkArDquR6YNqCTVXY7DQuUYcOkYHi4Ohl8jIDqWWlc/ZqaMut7txIRWY
F9GQrM/Crd6ONcXRxtYp2fYYEynIobDA9XZ4OPUc9KY4sRNVVMioiIHqCqErB4zxWjjhEBiOouxaX4oVafEKk5N3PA8tG
BMKZjj7PSqCljUkRE9IB1h65BWVRmcP0h/m3w09vD8eHEEKsZO2Sov7vSDG8xAoCEvZlQ1XcqHpGVYlJERHSffRkKrD7w
s8H1HUTA+n+GWOXYIUNVd6U9ZfAYIwGAorAUp7MLzBsYUSOz3XcxEZGJlVdWYcHuy0YdMy3c8H3KrFl1V1o3o7rSTlz7H
d9duA359Xy2GlGTwHWK9OA6RUT2Z1+GAgt2Z6CguNzgY6aF+2HhCOOmt1u76v+HyygorjDqOJlEjPioAJsZT0VNE9cpIi
JqoJRLCkRvO29UQrRufM8mlxAB9ZuVBgDKwlIOwCabx6SIiOxayqU7mLn9vFHHxAzugqeDfMwUkeXdPyvN0O40dZcDB2C
TLWNSRER2a1+GAq9+mW7UatUyiRgzn+xivqCshHpWmlQiNvgYDsAmW8cNYYnILqmqBMzfZdygahGA+KgAm5t2X1/DAmUY
EiDF6ewC5N4rxdXf/sS6w9fqPC73XmkjREdkemwpIiK7tO7QNdwtMXwwsYerk02sVG1qjg4ihHX2xMigtuj3sJdBx7RxM
7x1iciasKWIiOyOqkrAlhOG72nm6eoMedxTTWLqfUP08vOATCKGsrAU+nocRQCkEjF6+Xk0dmhEJmHf73Aiskunswtw9y
/DW4mWjQ60+4QIqG41io+qnnH3YAei+u/xj3fA95fucO0isklsKSIiu6KqEnDi2u8G1RUBWD/R/rrMaqMegJ2wNxOKwr/
HDrVq4QQB0FoNnGsXka3h4o16cPFGoqZpX4ZC58u8NrOfehgxQx41c1S2SVUlaAZg38grxuoDV3XqqFuP7HEsFlkGF28k
IjLAvgwFpm87b3BC5N7CCbOeesTMUdku9QDsp7v74Kszt/TW4dpFZGuYFBFRk6eqEpCwN1Pv4GB9RAASx3Szm6n3DXE6u
6DWRFO9dtGnJ7KZGJHVY1JERE3eqev5BrcQySRidvcYwdA1iZb+kIX+Kw5xGxCyakyKiKhJ25ehwIwvDdvGY+agh3F83p
NMiIxgzJpECu6PRlaOSRERNVnqcUSGTr/v97AXu8yMpF67yJg90uJ2XWZXGlklJkVE1CQZM45IhOpuMy46aLza1i6qyR8
lFVh3SHe2GpGlMSkioiaprgHAD7KnPc1MrT6bx245cYOtRWR1mBQRUZNk6ADgVs3tc08zUxsWKMPxeU9iiH8bg+rf/asC
p37JN3NURMZhUkRETZKhA4DX/5MJkak4OojgbURr0dQtpznomqyKxZOipKQk+Pn5QSwWIyQkBGlpabXWP3r0KEJCQiAWi
9GpUyckJydr3f/EE09AJBLp3EaMGGHOyyAiK1PXAGD1OKI+nTwbM6wmz8/T1eC65SoB0ZyNRlbEoknRjh07MGfOHCxcuB
Dp6ekIDw/H8OHDkZOTo7d+dnY2IiMjER4ejvT0dCxYsACzZs3Czp07NXV27doFhUKhuWVkZMDR0RH/+Mc/GuuyiMgKGLJ
5KccRmd7ksI4GD7hWm7/zEscXkVWw6N5nvXv3RnBwMDZs2KAp8/f3x6hRo5CYmKhTf968edizZw+ysrI0ZdHR0bh48SLk
crnex1izZg3eeustKBQKuLoa9guGe58R2abyyip8Lr+BmwUl8PVogclhHXHof7/p7HfGjUrNKzElEx8dyzbqmJjBj2D24
C5miojsRUO/v5uZISaDlJeX49y5c5g/f75WeUREBE6ePKn3GLlcjoiICK2yoUOHYtOmTaioqICTk5POMZs2bcL48eNrTY
jKyspQVlam+buoqMiYSyEiK5CYkolP0rJxf4PDspQsTAv3w/F5T2o2L23jVj31ni1E5hMXWd1CZ0xitOVkNmY++TCfF7I
oiyVFeXl5UKlU8Pb21ir39vaGUqnUe4xSqdRbv7KyEnl5eZDJtH/1nT59GhkZGdi0aVOtsSQmJiIhIaEeV0FE1qCmlokq
4e8vZvUXNTWOuMgAeLg6I/E/PxlU/25JBU5nFyCsM8d4keVYfKC1SKT9q0AQBJ2yuurrKweqW4kCAwPRq1evWmOIi4tDY
WGh5nbrlv4dn4nI+pRXVuHjtNpbJD5Jy0Z5ZVUjRURqU/t1gjENP4Yuo0BkLhZLiry8vODo6KjTKpSbm6vTGqQmlUr11m
/WrBk8PbV/XZSUlOCrr77Cyy+/XGcsLi4uaNmypdaNiGzDgl2XUdfIyCoB+Fx+o1Hiob85N3PAtHA/g+sbs48akTlYLCl
ydnZGSEgIUlNTtcpTU1PRt29fvceEhYXp1N+/fz9CQ0N1xhN9/fXXKCsrw6RJk0wbOBFZDVWVgJTLhk3nvllQYuZoSJ+4
yABMC+9Yax1us0LWwqLdZ7Gxsdi4cSM2b96MrKwsxMTEICcnB9HR0QCqu7WmTJmiqR8dHY2bN28iNjYWWVlZ2Lx5MzZt2
oS5c+fqnHvTpk0YNWqUTgsSETUdp7MLUFKhMqiur0cLM0dDNVk44jF8OKGn3vu4PAJZE4sNtAaAcePGIT8/H0uWLIFCoU
BgYCBSUlLg6+sLAFAoFFprFvn5+SElJQUxMTFYv349fHx8sHbtWowdO1brvD///DOOHz+O/fv3N+r1EFHjMnQMigjV6+e
Q5UT18IGTo0hneQQpl0cgK2LRdYqsFdcpIrIN8uv5mPDJqTrrPd1dhnUTgxshIqqLqkrg8ghkNja7ThERUUOpt/JQFpai
pl93rs6O+GC8/q4banyODiK90+6ZLJE1YFJERDZLvZXH9G3nIQL0JkbvP9eDX65Wbl+GQqdbzcPVGW+PDERkd3arUeOx+
DpFRETGUlUJkF/Px3cXbkPS3BnrJ/aE9IHd2WUSMZInBXOsipXbl6HA9G3ntRIiACgoLserX57Hsh+uWCgyskdsKSIim6
KvVUEmEWPRiAC4uzqz+8WGqKoEJOzNrLHrEwA+SbsBQISFI7giOZkfW4qIyGbU1KqgLCzFjC/Po/CvcowMaouwzp5MiGz
A6ewCnedSn0/SspFyybD1qIgagkkREdmE2loV1GUJezOhquKEWlthzLYeb+y8xOeWzI5JERHZhLpaFQQAisJSnM4uaLyg
qEGM2dbjz7JKrDt0zYzREDEpIiIbYWirAjcVtR29/Dzg4epscP0tJ7LZWkRmxaSIiGyCoa0K3FTUdjg6iPD2yECD69/9q
4ItgWRWTIqIyCaoF2qsafg0NxW1TZHdZRj0qJfB9dkSSObEpIiIbIJ6oUYAOokRNxW1ba8MeNjgumwJJHNiUkRENmNYoA
wbJgXrLNQolYixgQs12qxefh6QtnSpsx5bAsncuHgjEdmUYYEyDAmQcp+sJsTRQYTFzzyG6G3na63HlkAyN5EgCBzK/4C
G7rJLRETG25ehwPxdl3G3pEKr3L2FExLHdGNLINWpod/fbCkiIiKroG4FPHU9H/Jf8gCIENbZE306cYVyahxMioiIyGo4
OojQr4sX+nUxfEYakalwoDURERERmBQRERERAWD3GRER2RBVlcCZh2Q2TIqIiMgm7MtQIGFvptbGwB6uTnh7ZCAiu/tYM
DJqKth9RkREVm9fhgLTt53XSogAoKC4Aq9+mY7ElEwLRUZNCZMiIiKyaqoqAQl7M1HbonofHctGyiVFo8VETROTIiKyOF
WVAPn1fHx34Tbk1/OhquKasvS309kFOi1E+iz6LoOvHWoQjikiIovSN05EJhEjPiqAKxgTACD3Xt0JEQDkF5fjdHYBwjp
7mjkiaqrYUkREFlPTOBFlYSmmbzuPfRnsDiGgjZu47kr/n6EJFJE+TIqIyCJqGyeiLkvYm8nuEEIvPw94uDoZVNeYBIro
QUyKiMgi6honIgBQFJbidHZB4wVFVsnRQYS3RwbWWU8mqV63iKi+mBQRkUUY2s3B7hACgMjuPvjXAL8a7xcBiI8K4EKO1
CBMiojIIgzt5mB3CKnFRQYgaWIwPFydtcplEjE2TArmwHxqMM4+IyKL6OXnAZlEDGVhqd5xRSIAUnaH0AMiu8swNFDKrT
7ILJgUEZFFODqIEB8VgOnbzkMEaCVG6q83doeQPo4OIr3T7rkvGjUUkyIisphhgTJsmBSss06RlOsUkZH2ZSiweE8mlEX
3vY5airH4Gb6OyHAWH1OUlJQEPz8/iMVihISEIC0trdb6R48eRUhICMRiMTp16oTk5GSdOnfv3sWwaItRTt27MCcOXOQlJSEf
v364aOPPsLw4cORmZmJDh066NTPzs5GZGQkpk2bhm3btHEiRN49dVX0bp1a4wdOxYAUF5ejiFDhqBNmzb45ptv0K5dO9
y6dQtubm6NfXlEpEdNXRxchZjqQ1UlYP6uy7XWidt1GUMCpOxKozqJBEGw2MpovXv3RnBwMDZs2KAp8/f3x6hRo5CYmKh
Tf968edizZw+ysrI0ZdHR0bh48SLkcjjkAIDk5Ge+++7w97+97//wcnJsMW+HlRUVASJRILCwkKwkK0bNmyXucgIl3c0oNM7cS1

PPxz43/rrPfFy73R72GvRoiILKmh398W6z4rLy/HuXPnEBERoVUeERGBkydP6j1GLpfr1B86dCjOnj2LiooKAMCePXsQF
haGGTNmwNvbG4GBgVi+fDlUKlWNsZSVlaGoqEjrRkSmxS09yBzk1/NNWo/sm8WSory8PKhUKnh7e2uVe3t7Q6lU6j1GqV
TqrV9ZWYm8vDwAwC+//IJvvvkGKpUKKSkpePPNN/H+++9j2bJlNcaSmJgIiUSiubVv376BV0dE9+OWHmQ+hr1mBAPrkX2
z+EBrkUi7j1cQBJ2yuurfX15VVYU2bdrg448/RkhICMaPH4+FCxdqddE9/QKC4uDoWFhZrbrVu36ns5RKQHt/QgcwnrZFiX
2Jf/vcnWSKqTxZIiLy8vODo66rQK5ebm6rQGqUmlUr31mzVrBk/P6kGaMpkMjzzyCBwdHTV1/P39oVqUV5erve8Li4ua
NmypdaNiEyHW3qQufTp7IlWLeoeP/pHSSW7aalOFkuKnJ2dERISgtTUVK3y1NRU9O3bV+8xYWFhOvX379+P0NBQzaDqfv
364dq1a6iqqtLU+fnnnnyGTyeDsrL00PBE1Dm7pQebi6CDCO2O6GVRXQPVMNHbTUk0s2n0WGxuLjRs3YvPmzcjKykJMTAx
ycnIQHR0NoLpba8qUKZr60dHRuHnzJmJjY5GVlYXNmzdj06ZNmDt3rqbO9OnTkZ+fj9mzZ+Pnn3/GDz/8gO0XL2PGjBmN
fn1EVE29pUdNHeMicIdzqr9hgTIkTwqGu0EtRhVYd+hqI0RFtsiiSdG4ceOwZs0aLFmyBEFBQTh27BhSUlLg6+sLAFAoF
MjJydHU9/PzQ0pKCo4cOYKgoCAsXboADt27fH/v37cebMGXTv3h2zZs3C7NmzMX/+/Ea/PiKqpt7SA4BOYs
QtPcgUhgXK8NbTAQbV3XLiBluLSC+LrlNkrbhOEZF5cJ0iMif59XxM+OSUQXW3T+vDBUOboIZ+f3PvMyIymwdXrx4SIMW
QAO5wTubRy88DrZo74e5fFXXW5aB+0odJERGZBVuFqLE5OogwtZ8fVh/4uc66Xq4ujRAR2RqLr1NERE0PV68mS5n55MMG
TdF//d8X+TokHUyKiMikuHo1WZKhU/SVRaWIZoJOD2BSREQmxdWrydLUU/SlLevuIpvPdYvoPkyKiMikuHo1WYNhgTK8/
1xQnfXullRg3aFr5g+IbAKTIiIyKa5eTdYi788eTtA08b5b5UsUsHuAUsHuXALApIiITIyrV5O1O1UK
9bZAh25xLApIiIzGBYoAwbJgVDKtH+pS6ViLFhUjDXKaJGUb1uUUeD6rI7lwAu3khEZjIsUMbvQ8niZj7ZBVtO3sDdEv2
rXItQnayzO5cAthQRkRk5OogQ1tkTI4PaIqyzBgdfLOwdZkdFJ05MgRM4RBRERkHuruXNkD
3bmi/58pbT5xAxM+OYX+Kw5xMUc7JxIEwajUWCwWo23btpg6dg6dSqef/55tG/f3lyxdd9klIiLro96g+ECmEptO3NC5X
92axHFvtquh399GtxTduXMHs2fPxq5du+Dn54ehQ4fi66+/+/Rnl5udEPTkRE1FgcHUTo5eeBlAyl3vu5DQ0ZnRR5eHhg1q
xZOH/+PM6ePYtHH30UM2fOhM2bMgEwmw6xZs3Dx4kVzEzEsWSq3AIsFBERRRg3IaGatOggdZBQUGYP38+ZsyYgeLiYmzevBkhISEIDw/HlSt
XTBUjERGRSXAbGqpNvZZKiiooKfPPPN4iMjISvry9+sIry9+/PFHrFu17n17ERERA3CbWioNkavU/Ta
a69h+/btAIBJkyZh5cqVCAwM1nzv6uqKKag3oZGWVgKfaOGuG6RfTM6KcrMzMSHH36IsWPHwtnZW
W8dHx8fHD58uMHBERERmZJ6G5rp285r1vi8QhGJyrwjDVOwk89smaJSSYPMZqfC
6/gZsFJfD1aIHJYR3h3IxrwZLt2JehQMLeTK1B1zKJGPFRAZyOb8Ma+v1tdFLk6OgIhUKBNm3aaJXn5+ejTZs2UKlURgdvblnpjXg2FPL
hbZgUEdUsMSUTn6Rl4/4ZywiYFq4H+IiAywXGJGR1OsWcRuapqOh399Gd58JggCRSPdFk5+fD1dXVoHpFtGjp1DDLLxQ+JtAFr
ZeuUVwnQlDMxIluh3oaGSM3gpGjMmDEAAJFIhOeff15rppVKpcLFixfR19+tocH0QFS5duoS+ffuaPkIsgrllVX4JE3JgE03IbrfJ2nZeD2iK
7vSiMgmGZwUSSUBSQNANUtRW5ubmjmjxtVKlcnZ2Rp8+fTADM5KFBntlXvAgsGfD04HhAnKX9JCnsd19XZkmSSkAAB
07dsTcuXPZVUkZkZ24WlJi0HhGRtTF6TFF8fLw54iAiK+fr0cKk9YiIrI1BSVFwcDAOHjwjwId3d39OzZU+9Aa7Xz58bLDg
ish6tXfWvS3Y/BxEwOayj+YMhIjIDg5KikSNHagzZWjxo1ypzxEJEVqmnW2YOmhftxkDUR2Syj1ymyB1ynipOhvKZcUePXL
uluAp4X7YeEITscnIstp6Pe30T/pbt26hV9//VXz9+nTpzFnzhx8/PHHHRj84EVk3VZWAN7/LMKjuk129zRwNEZF5GZ0UT
Zw4UbOvmVKpxODBg3H69GksWLAAS5YsMXmARGQ5p7MLUFBcblDd3HuldV3ciIrJiRidfGRkZ6NWrFwDg66+/Rrdu3XDy5E
l8+eWX+PTTT40OICkpCCkp0CX5+fhCLxQgJCUFaWlqt9Y9Y8ePYqQkBCJxWJ06tQJycnJWvd/+umnEIlEORfSUn5gExnLmESnjZv
YjJEQEZmf0UlRRUWFtZtD1gQMH8MwwzzcYd5duzYgTlz5mDhwoVIT09HeHe3Hd4i9s7OzERkZifDw8KSm8PxPtzGzYGTlz5mDhwoVIT09HeHe3Hg4hg8fjpycL31s7OzERkZifDw
cKSnp2PBggZgWYNWsWdu7cqVm/WTXvWxFSRd6kcHMjemdWwauUjcbQuFfPDRqrFmvcWYNtcD8b1ljdm8i
HDnzh14ehq3h8yqvavw0ksv4W+4WXX4a/vz/WrFmD9u3bY8OGDXrrJycnw8fHB2vWrIG/vzwneVrmzNe3x8ssv48
EqlWrdiMh4vfw8IJPUnRi9PTKQG2kSkc0zOilasWIFWIFpvroIzzxBOYMGECEvToAQDYs2ePMLw1b37+bhwiIiK0yiM
iInDy5Em9x8jlcp36Q4cOxdmzZ1FRUaEAEp+/PPP+Hr64t27drh66RD7HeyRjrHDNHcPMLMEQjsHP
ZC5Gr2j9xBNPPIC8vD0VFRXB3d9eUv/LKKwCAhw4eCYHBwcLmP7rFt0gmh4vbZPbJXBBUjPwuu34r82FUSlpa
PTTT9GtWzcUFRXhgw8QL9+/XDx4kV06dJF73nLysyppQSQNVlam+buoqMYPqEnrZIwrIqfIqubqYi
4yC0ZHRGQ6RidFAODo6qjvEVAHVe6LVx4rYwuCUOk2rfq31hQtOo81+1KpkU7Hd97MmR2DVMTFMM8tPGCnYZRH3eWh+r3p2QlO9Ms
UK34iezAsUIYhAVKcz5i7r1StHETo5efB7vMiKhJMbr77LfffpszPkz2awd
BqlJpVK99Zslai9bjeCYHBwc8/vjju1Hr1aozxxMXFobCwCwUHO7deuWwddBlBBSpqgTIr+fjuwu3IbTBIeeD1WVAEcHEc
FuEdfZkQkR2Sd97g5oOoluKXnjhBeTk5GDRokWQyWR6Z6IZKi8vDyqVCt7e3lrl3t7eUCqVeo9RKpV661dWViIvLw8yWe0Vx57o5ampqRg
379yM0NBROTk56jxEEARcxESvOkiycsHbYTYS5ZAUykh7b3H45a2v6C+yg8Pd3vNtHMfDcR/3Q6NRAQ2Nh7ubG9m3nv81kc4k5bvWq
gB9+xYwcmT56M5ORkR1W4OeMWfkYpL4J8fMfc/mP9XZ3X3b2H3h2dPIm6HOa1OGvWz4xl0s1Ad2Zn7
7FgAQEJCAvr06YMuXbqqgKgIa9euxuxeeff44TJ04YPBcc0ChPafhnB/PAnu8T3hmlo6Pe30S1F7
du3h6m2Sxs3bhzy8/OxZMkSKBXXpuWkKBQNAJCo/gQXfGWhxwKbtNQh+RgcMnSUs+S1r3ZJJHKZHvRVA5dc3dNVEJSo7Ykmmgdo7qGjV8Q7mnEaA1GGWEQrFabZLd6w
tQKpWQSCTo2bMnjh07ZttZMOCJ7pKoSkLA3U+dDHwAEVH/4J+zNxZAKbvOyK7wvWE/jG4p2r9/P95P2ctSH9R5cbe3
YUkT2SH49HxM+OVVnve3T+iCss3FrkhHZMr43bEejtxSNGzcOJSuHJ0cOJSUl6Ny5M1q0Aq0KEz1qegoMDoIIjI8g5k6l8K40H6c44zs
jaGveb43bJ/RSdGaNWvMEAYRkR4GQsA1nhsbFpGS5IPbWSTbWw7QPTkhKmDHBiHg0ii06ZYyTEheOvWZEpvndF0BKByZZFNWMENmOTS4V8zaGu9XWD8JgrcndRZRWjbmIJaWDONo
ER1qm0mp4DqGZWJezMxJEDKrjQbYFBSdPjwYc2/V61aBTc3N3Wdwd3cHAPzxxxxx+YOnUqwsPDzRMlEZmUetuC6dvOA4
DWBzqn4RMZrq6ZnAIARWEpTmcXcH0jG2D07LP3338fiYmJmoQIcfj222/H2222Wdwd3cHAPzxxx+YOnUqwsPDzRMlEZmUetuC6dvOA4
yjB5oXVRUhN9++w2PPfaYVnlubi7sbuzuSaxxxF/xtbosbO3FAv0gfvfPnwN4v3330efPnOAAKdOncL/
/d//YcyyMSYPkIjMi9sWENWNfenyesrBU77gibpNjW4zuPktOTsaIESMwadIk+Pr6P5z8xfPhwJCUlmSNGIiIiq
6Qenwf8PR5PjePzbI9IEIS6tj/Sq7i4GNevX4Fh6upq6tgspiqoCBKJBIWFhWwyhHRWjZsqWlwyEiIivHdYqsQ0O/v+
udFDVlTIqoqeFKu0Tmx/eZ5TX0+7veK1oTkW3gL1iixsHxebbP6DFFRGQ71CvtPriOinql3X0ZCgtFRkRkfZgUETVRda2
0C1SvtKuqYg86kTmpqgTIr+fjuwu3Ib+ez/ecFTO6+6y4uLhJDaomaqq40i6R5bH72rYY3VLk7e2Nf198EcPehPERk
Ilxpl8iy2H1te4xOirZv347CwkI89d+RTeOSRR/DOO+/gzp075oiNR7P10+oiNiBqAK+0SWQ67r22T0U1RVFQUdu7ciTt37mD69OVv
n07fH198fTTT2PXrl2orKw0R5xEZCTlQBbsnjK6b1EEo7C7IL1Srs1TQgWoboZnyvtEpmeMd3XZD3qPdDa09MTMTExuHjwIlatoWoUDBw7g2efy

+PD9566y2UlJSYMk4iMhJX2iWyHHZf26Z6J0VKpRIrV66Ev78/5s+fj2effRYHDx7E6tWrsXv3bowaNcqEYRJRfQwLlGH
DpGBIJdpdZFKJGBsmBXOgJ5GZsPvaNhk9+2zXrl3YsmULfvzxRwQEBGDGjBmYNGkSWrVqpakTFBSEnj17mjJOIqqnYYEy
DAmQcqVdokbEjWJtk9FJ0dSpUzF+/HicOHECjz/+uN46nTp1wsKFCxscHBGZBlfaJWpc6u7r6dvOQwRoJUbsvrZeRu99V
lJSghYtWpgrHqvAvc+IiMgUalqn6LnQ9lBVVQGo/sHSp5MnEyQTaOj3t9Fjitzc3JCbm6tTnp+fD0dHR6MDSEpKgp+fH8
RiMUJCQpCWllZr/aNHjyIkJARisRidOnVCcnJyjXW/+uoriEQijm8iu8GVc4msy7BAGY7PexLbp/XBB+ODEDO4C/4qr8Q
HB69i3eHrWHf4Gv658b8IeTuV6xZZAaOTopoalsrKyuDs7GzUuXbs2IE5c+Zg4cKFSE9PR3h4OIYPH46cnBy99bOzsxEZ
GYnw8HCkp6djwYIFmDVrFnbu3KlT9+bNm5g7dy7Cw8ONionIVu3LUKD/ikOY8MkpzP7qAiZ8cgr9VxziBy2Rham7r12aO
WD1gau4+5fu0jV3SyoQzQUdLc7g7rO1a9cCAGJiYrB06VI89NBDmvtUKhWOHTuGGzduID093eAH7927N4KDg7FhwwZNmb
+/P0aNGoXExESd+vPmzcOePXuQlZWlKYuOjsbFixchl8u14hk4cCCmTp2KtLQ03L17F99++63BcbH7jGyNeuXcB9/M6sZ
4zjQjsixVlYB+7xyEsqis1noyiRjH5z3JrrR6auj3t8EDrVevXg2guqUoOTlZq6vM2dkZHTt2rLUr60Hl5eU4d+4c5s+f
r1UeERGBkydP6j1GLpcjIiJCq2zo0KHYtGkTKioq4OTkBABYsmQJWrdujZdeeqnO7jigupWrrOzvF2pRUZHB10FkaXWtn
CtC9cq5QwKk/KAlspDT2QV1JkQA9yO0NIOTouzsbADAoEGDsGvXLri7uzfogfPy8qBSqeDt7a1V7u3tDaVSqfcYpVKpt3
5lZSXy8vIgk8lw4sQJbNq0CRcuXDA4lsTERCQkJBh9DUTWgBu/Elk/YxZp5IKOlmP0mKLDhw83OCG6n0ik/ctVEASdsrr
qq8vv3buHSZMm4ZNPPoGXl5fBMcTFxaGwsFBzu3XrlhFXQGRZXDmXyPoZs0jjjbxiM0ZCtTGopSg2NhZLly6Fq6srYmNj
a627atUqgx7Yy8sLjo6OOq1Cubm5Oq1BalKpVG/9Zs2awdPTE1euXMGNGzcQFRWlub+qqgoA0KxZM/z000/o3LmzznldX
Fzg4uJiUNxE1oYr5xJZv15+HpC2dDGoC2376RzMfLILu7stwKCkKD09HRUVFZp/16S2Fp4HOTs7IyQkBKmpqRg9erSmPD
U1FSNHjtR7TFhYGPbu3atVtn//foSGhsLJyQldu3bF5cuXte5/8803ce/ePXzwwQdo3769wfER2QqunEtk/RwdRFj8zGO
I3na+zrrKojJ2d1uIQUnR4cOH9f67oWJjYzF58mSEhoYiLCwMH3/8MXJychAdHQ2gulvr9u3b2Lp1K4K4DqmWbr1q1DbGws
pk2bBrlcjk2bNmH79u0AALFYjMDAQK3HUG8/8mA5UVPBlXOJbMOwQBle6tcRm07cqLMuu7sto94bwprCuHHjsGbNGixZs
gRBQUE4duwYUlJS4OvrCwBQKBRaaxb5+fkhJSUFR44cQVBQEJYuXYq1a9di7NixlroEIqvAjV+JbMPgAKlB9djdbRkGrV
M0ZswYg0+4a9euBgVkDbhOEdkqVZXAjV+JrJiqSkD/FYfq7O7mWkX10yjrFEkkEqNPTESNjxu/Elk3dndbN6M3hLUHbCk
iIiJzqmmj2PioAHZ3N0CjrWhNRNaBXWREtm9YoAxDAqR8L1sZg5Ki4OBgHDx4EO7u7ujZs2etU+/Pn697uiER1Q9/XRI1
Hezutj4GJUUjR47ULG44atQoc8ZDRDWoadNXZWEppm87z1lmREQNxDFFenBMEVkb9YyVmvY444wVIiILjik6e/YssrKyI
BKJ4O/vj5CQkPqeiojqwE1fiYjMz+ik6Ndff8WECRNw4sQJzWrRd+/eRd++fbF9+3ZupUFkBtz0lYjI/Ixe0frFF19ERU
UFsrKyUFBQgIKCAmRlZUEQBLz00kvmiJHI7nHTVyIi8zO6pSgtLQ0nT57Eo48+qil79NFH8eGHH6Jfv34mDY6IqscTVQk
CWjV3wt2/KvTW4aavREQNZ3RS1KFDB1RU6H4wV1ZWom3btiYJioiq6ZuC/yCugktEZBpGd5+tXLkSr732Gs6ePQv1xLWz
Z89i9uzZeO+990weIJG9Uk/Bry0hArjpKxGRqRg0Jd/d3V1rwcbi4mJUVlaiWbPqhib1v11dXVFQUGC+aBsJp+STpdU1B
R8AWrVwwvoJwejT2ZMtREREaKQp+WvWrDH6xERUf3VNwQeAuyUVcHAQMSEiIjIRg5Ki559/3txxENF9OAWfiKjxNWhD2L
/++ktn0DW7m4ga7kZesUH1OAWfiMh0jB5oXVxcjJkzZ6JNmzZ46KGH4O7urnUjoobZl6HA6gNXa60jQvVGsJyCT0RkOkY
nRW+88QYOHTqEpKQkuLi4YOPGjUhISICPjw+2bt1qjhiJ7IaqSkDC3kyD6nIKPhGRaRndfbZ3715s3boVTzzxBF588UWE
h4fj4Ycfhq+vL7744gv885//NEecRHbBkAHWADBn8COcgk9EZGJGtxQVFBTAz88PQPX4IfUU/P79++PYsWOmjY7Izhg6c
LqjVwszR0JEZH+MToo6deqEGzduAAACAgLw9ddfA6huQVJvEEtE9cM9zoiILMfopGjq1Km4ePEiACAuLk4ztigmJgb/93
//Z/IAiexJLz8PyCRi1DRSiAOsiYjMx6AVrWtz8+ZNnDt3Dp07d0aPHj1MFZdFcUVrsiT19h4AcP+bU50ocUsPIiL9Gvr
93eCkqCliUkSWpm8jWJlEjPioACZEREQ1aJRtPh508OBBrF69GllZWRCJROjatSvmzJmDwYMH1+d0RPSAYYEyDAmQ4nR2
AXLvlaKNW3WXGafgExGZj9FjitatW4dhw4bBzc0Ns2fPxqxZs9CyZUtERkZi3bp15oiRyC45OogQ1tkTI4PaIoybvhIRm
Z3R3Wdt27ZFXFwcZs6cqVW+fv16LFu2DHfu3DFpgJbA7jMiIiLb09Dvb6NbioqKijBs2DCd8oiICBQVFRkdABEREdWfqk
qA/Ho+vrtwG/Lr+VBVcahwfRk9puiZZ57B7t27dabff/fdd4iKijJZYERNnapK4JghImoQTsowLYOSorVr12r+7e/vj2X
LluHIkSMICwsDAJw6dQonTpzA66+/bp4oiZoYfpARUUOpl+94sF1IWViK6dvOc/mOejBoTJF6W486TyYS4ZdffmlwUJbG
MUVkTjV9kHEdIiIylKpKQP8Vh2rcK1EEQCoR4/i8J+2qBbpRpuRnZ2cbfWIi0qWqEpCwN1MnIQKqF2oUAUjYm4khAVK7+
iAjIuPUtXm0AEBRWIrT2QUI6+zZeIHZOKMHWt9PEAQ0dO3HpKQk+Pn5QSwWIyQkBGlpabXWP3r0KEJCQiAWi9GpUyckJy
dr3b9r1y6EhoaiVatWcHV1RVBQED7//PMGxUhkKsZ8kBER1cTQzaMNrUfV6pUUbd26Fd26dUPz5s3RvHlzdO/evV6Jx44
dOzBnzhwsXLgQ6enpCA8Px/Dhw5GTk6O3fnZ2NiIjIxEeHo709HQsWLAAs2bNws6dOzV1PDw8sHDhQsjlcly6dAlTp07F
1KlT8eOPP9bnUolMih9kRGQK3DzaPIyefbZq1SosWrQIM2fORL9+/SAIAk6cOIHo6Gjk5eUhJibGqHO99NJLePnllwEAa
9aswY8//ogNGzYgMTFRp35ycjI6dOiANWvWAKge9H327Fm89957GDt2LADgiSee0Dpm9uzZ+Oyzz3D8+HEMHTrU2MslMi
l+kBGRKag3j1YWlurtjlePKeLm0cYxuqXoww8/xIYNG7BixQo888wzGDlyJFauXImkpCStWWp1KS8vx7lz5xAREaFVHhE
RgZMnT+o9Ri6X69QfOnQozp49i4qKCp36giDg4MGD+OmnnzBgwIAaYykrK0NRUZHWjcgc1B9kNY0WEqF6Fho/yIioNo4O
IsRHBQCAzueJ+u/xj3fA95fucO0iIxidFCkUCvTt21envG/fvlAoFAafJy8vDyqVCt7e3lrl3t7eUCqVeo9RKpV661dWV
iIvL09TVlhYiIceegjOzs4YMWIEPvzwQwwZMqTGWBITEyGRSDS39u3bG3wdRMYw5IMsPiqAg6yJqE7DAmXYMCkYUol2y3
KrFk6QtHDC6gM/Y/ZXFzDhk1Pov+IQ9mUY/h1tr4xOih5++GF8/fXXOuU7duxAly5djA5AJNL+8BcEQaesrvoPlru5ueH
ChQs4c+YMli1bhtjYWBw5cqTGc8bFxaGwsFBzu3XrltHXQWSomj7IpBIxp+MTkVGGBcpwfN6T2D6tDz4YH4SYwV3wR0kF
7pZo956o1y5iYlQ7o8cUJSQkYNy4cTh27Bj69esHkUiE48eP4+DBg3qTpZp4eXnB0dFRp1UoNzdXpzVITSqV6q3frFkze
Hr+PeXQwcEBDz/8MAAgKCgIWVlZSExM1BlvpObi4gIXFxeDYydqqGGBMgwJkHJFayJqMPXm0eq1i/Thkh+GMbqlaOzYsT
h9+jS8vLzw7bffYteuXfDy8sLp06cxevRog8/j7OyMkJAQpKamapWnpqbq7Z4DgLCwMJ36+/fvR2hoKJycnGp8LEEQUFZ
WZnBsRI1B/UE2Mqgtwjp78kOKiBqES340nFEtRRUVFXjllVewaNEibNu2rcEPHhsbi8mTJyM0NBRhYWH4+OOPkZOTg+jo
aADV3Vq3b9/G1q1bAQDR0dFYt24dYmNjMW3aNMjlcmzatAnbt2/XnDMxMRGhoaHo3LkzysvLkZKSgq1bt2LDhg0NjpeoP
rjHGRE1Bi750XBGJUVOTk7YvXs3Fi1aZJIHHzduHPLz87FkyRIoFAoEBgYiJSUFvr6+AKoHdd+/ZpGfnx9SUlIQExOD9e
vXw8fHB2vXrtVMxweA4uJivPrqq/j111/RvHlzdO3aFdu2bcO4ceNMEjORMbjHGRE1Fi750XAG7X12v6lTp6Jbt26IjY0
1V0wWx73PyBS4xxkRNSb1mKK61i5qyvuhNcreZ/d7+OGHsXTpUpw8eRIhISFwdXXVun/WrFlGB0HU1HCPMyJqbOolP6Zv
Ow8RoPX5wyU/DGN0S5Gfn1/NJxOJ8MsvvzQ4KEtjSxE1lPx6PiZ8cqrOetun9eFmjURkUvbcbd/oLUXZ2dlGPwiRveGAR
yKyFC75UX9GJ0X307dwIhFxwCMRWZZ6yQ8yjtHrFAHApk2bEBgYCLFYDLFYjMDAQGzcuNHUsRHZLO5xRkTWSlUlQH49H9
9duM190R5gdEvRokWLsHr1arz22msICwsDUL1Ra0xMDG7cuIG3337b5EES2RoOeCQia2TP440MYfRAay8vL3z44YeYMGG
CVvn27dvx2muvaW3Maqs40JpMhR9ARGQt7GGZkEYfaK1SqRAaGqpTHhISgsrKSqMDIGrKOOCRiKwBlwkxjNFjiiZNmqR3
y4yPP/4Y//znP00SFFFTwj3OiMjSuC+aYeo1+2zTpk3Yv38/+vTpAwA4deoUbt26hSlTpmitdL1q1SrTRElERET1xmVCD
GN0UpSRkYHg4GAAwPXr1wEArVu3RuvWrZGRkaGpx2n6RERE1oHLhBjG6KTo8OHD5oiDiIiIzES9TEhd+6LZ+zIh9VqniI
iIiGyHepkQADrrp3GZkL8xKSIiIrIDwwJl2DApGFKJdheZVCJuEtPxTaFB23wQERGR7eAyIbVjUkRERGRHuC9azdh9RkR
ERAQmRUREREQAmBQRERERAWBSRERERASASRERERERACZFRERERAA4JZ+oTqoqgWt6EBHZASZFRLXYl6FAwt5MKAr/3jla

JhEjPiqAq78SETUx7D4jqsG+DAWmbzuvlRABgLKwFNO3nce+DIWFIiMiInNgUkSkh6pKQMLeTL27SavLEvZmQlWlrwYRE
dkiJkVEepzOLtBpIbqfAEBRWIrT2QWNFxQREZkVkyIiPXLv1ZwQ1aceERFZPyZFRHq0cRObtB4REVk/JkVEevTy84BMIk
ZNE+9FqJ6F1svPozHDIiIiM2JSRKSHo4MI8VEBAKCTGKn/jo8K4HpFRERNCJMiohoMC5Rhw6RgSCXaXWRSiRgbJgVznSI
isluqKgHy6/n47sJtyK/nN5mZuBZPipKSkuDn5wexWIyQkBCkpaXVWv/o0aMICQmBWCxGp06dkJycrHX/J598gvDwcLi7
u8Pd3R2DBw/G6dOnzXkJ1IQNC5Th+LwnsX1aH3wwPgjbp/XB8XlPMiEiIru1L0OB/isOYcInpzD7qwuY8Mkp9F9xqEms3
WbRpGjHjh2YM2cOFi5ciPT0dISHhyMqKgqZmZk667Ov/37//zsKEX9Yq0+KQxZfERvgCDhzkemYfAbYHnEChbDnCf8S
SntU7XPAFbnBsYq+GUNNQ+D9u4oORJ0mK+VBLzKY1p5rzKhuHPlkqyxEm+qYwWyPPXH5lUOqMk0DRRvfRPPEw7+/aF+v
Pkk2m3LVtJbcmUsnHh4Ve6ygtCcjNrnY4w+YB3+7d5e5ZbkqFmVl9pD6Nxbqtxiuq9grfNpaFmkF1i+3zGTaTaJY8Sz8QWr
pX9evqzEnZBAtgSsDoS1kRIUhTmBy/8jvF5Y1K1HaTAcqsQkc5OuI1KmTnvB1fhHnjsiQ2cNTFC61DPTXKEY5bqzDqZtrZ
x1jaKiUEXy6TkMMxHlDjsgqjaaNutbRm2NmRWhYW9dVa7dY7HaxM4KEDG7vGMeafGZPfTFJXbT1YoFjzmjYcaqfjGSPK5s
4eC1K0yWfLLMGSZV/rv2QXt8BZVa6NoBgsKJmGGAwnNMWjmTG+vbNYWtcZCgmgbEb6/C6X4zHmHgVvpEpKnP+NhMCh+xo
9XJ4aQYf8kc0Jv4vMX1uRH8NJ0aTJ/q7hKzJ5mxxjXahVbqs/P4QNdnn5BaNHPV6Xb61sk7OP8mIrP4qN8e0FmWbVkr2W
dBqhkyNqkGKqXKqmj0W4EBcrrMrhi7t/cvzzZcWqSPoZfgGaaBYOX7LomYSZkWL5JmhlaCxePVk5R0ax0mlKB/Upg6vb
hiiq9WKYcnhrx2NnvZOvfj6ObQkDJGnLPAbfL5c5EkEq+Jd7tdT2sj+EjcEqIm9MaPC3Hlk6J3s5lBvVxVu6Rn5pcHy
FkKCWmKSBfzQ9kUB2xEmLzN4YgiLkNWJzHvGMZR9/CrGUgdbvnmMf4yJKv/Ni8ZKO9PTgXdmoqvf2zWfK+aG2aZgSgH/+
g/9ZPVqQwcVYdWjUJ8qFCK0PBU4pp8/D1LfOF+9bN1t0/6Ff1ENoi9yFzZ63RHOgQb4IDIsYHdaH0+8OXv0MPWAtHJBB
ldhIhwXgx2X9JRwdeRVXbu+Yy6TBy6tEmRExlOnfnD9TTOjx4wYhMT7yLmHHiFWvXZMxGThXYA0tBdsWWB3ZURxCCY+d
n2cSDAXgWWGmdzWQfIHeAKrmFBSwMGmiqcxnN7f+7aoXXzpy1fODtfr4N3X2HcYBwnjmZzyOfbeHvVVeBHeIaQNh9Pkw
m7/P9mtCjOewh+gh+Zz3tUGVVnFw2AzJRD4UWcMkmqC3zKgyRz91O5pbcpNTH1JIw7Ncx7IbVTE7DXWvoUxlI6xSSGWjwf
+lLWbpC8CVCaVGZE8MCCF8fi0yyGpFbhw4C0tTlmVyHQHPjngUpdfSeGdtfpd8xPIpUEdWzxbHBfmbPl3g2XL1nZbddbLd
/Br/sdKrG2DOjYzKvzRkIKwoPFnEXmt/yyb/H2+fkaL9Vmc9ffbzZuJBHG/bLBAq9zaN/UBHxGxsNEfEEq9J7Cd3Y+7ZBf
+8cCJFXhwCbyuBWNCCXl3kxDFkeVXPGR+H81BxlYK7UgK9EHmtldaVYwrXi+xmnVNaqBIyDN/tQOGD+bnPCdTfJp2g6lq
4N1m9SUvwxaW9/a0J3yVO+JWrHQ6bJ2OPcJbyj6cYLOZx1nDkm7vf2eLjd/AC+u8L59GwMxeTHDDGV2L5jn9G4wBI0pHR
5zQfoLfkdNVm1fknXpGSENN0tYJfCJ9Gt/8H9Y9yNv5C4DeDjW9V3XUWl8U3VX/nXmGfW6pS5yWLcZJW1AXvvD69p5Ao8
4qpHoxrHH6Eea9ZBqbrMvJmWfrOkMXYgh1nsSumE1Tgkl+6oaM3CHBz3yHHEJG9nrCjXz8g8T0YVgWDi8WxK3BXN9IuO
/x5NIpFbPE3wEx8X5aHwGn/IIjYHTc7zuUnXTLDgcrnAJz1vmc7LB1YdIGtL1CNdSfZdwEw/wEMfnTUXAXpzbMKL9sVRFQj
JhEjPiqAq78SETUx7D4jqsG+DAWmbzuvlRABgLKwFNO3nce+DIWFIiMiInNgUkSkh6pKQMLeTL27SavLEvZmQlWlrwYRE
dkiJkVEepzOLtBpIbqfAEBRWIrT2QWNFxQREZkVkyIiPXLv1ZwQ1aceERFZPyZFRHq0cRObtB4REVk/JkVEevTy84BMIk
ZNE+9FqJ6F1svPozHDIiIiM2JSRKSHo4MI8VEBAKCTGKn/jo8K4HpFRERNCJMiohoMC5Rhw6RgSCXaXWRSiRgbJgVznSI
isluqKgHy6/n47sJtyK/nN5mZuBZPipKSkuDn5wexWIyQkBCkpaXVWv/o0aMICQmBWCxGp06dkJycrHX/J598gvDwcLi7
u8Pd3R2DBw/G6dOnzXkJ1IQNC5Th+LwnsX1aH3wwPgjbp/XB8XlPMiEiIru1L0OB/isOYcInpzD7qwuY8Mkp9F9xqEms3
WbRpGjHjh2YM2cOFi5ciPT0dISHh2P48OHIycnRWz87OxuRkZEIDw9Heno6FixYgFmzZmHnzp2aOkeOHMGECRNw+PBhyO
VydOjQARQTAREREbh9+3ZjXRY1MY4OIoR19sIYR16+zJLjMislt+NfVFbkSAIFmvz6t27N4KDg7FhwwZZZmHnzp2aOkeOHMGECRNw+PBhyO

b6BiU+eDt79qzB5xOJRDplgiDoLSfDGPMcxcTEYODAgejevTtefvllJCcnY9OmTcjPz7fwVdiHB1/nfO03vuHDh2Ps2LH
o1q0bBg8ejB9++AEA8Nlnn1k4MlIzxfuELUVkFjNnzsT48eNrrdOxY0eDziWVSvHf//5Xq+yPP/5ARUWFzi8DMlxDnqM+
ffoAAK5du8ZfyWbk5eUFR0dHnVah3NxcvvYtzNXVFd26dcPVq1ctHYrdU88CVCqVkMlkmvL6vE+YFJFZeHl5wcvLyyTnC
gsLw7Jly6BQKDQv+P3798PFxQUhISEmeQx71JDnKD09HQC0PoDI9JydnRESEoLU1FSMHj1aU56amoqRI0daMDIqKytDVl
YWwsPDLR2K3fPz84NUKkVqaip69uwJoHo83tGjR7FixQqjzsWkiCwuJycHBQUFyMnJgUqlwoULFwAADz/8MB566CFEREQ
gICAAkydPxrvvvouCggLMnTsX06ZNQ8uWLS0bvB2Qy+U4deoUBg0aBIlEgjNnziAmJgbPPPMMOnToYOnwmrzY2FhMnjwZ
oaGhCAsLw8cff4ycnBxER0dbOjS7MnfuXERFRaFDhw7Izc3F22+/jaKiIjz//POWDs0u/Pnnn7h27Zrm7+zsbFy4cAEeH
h7o0KED5syZg+XLl6NLly7oULli9fjhYtWmDixInGPZBJ5scRNcDzzz8vANC5HT58WFPn5s2bwogRI4TmzZsLHh4ews
yZM4XS0lLLBW1Hzp07J/Tu3VuQSCSCWCwWHn30BUSE+Pl4oLi62dGh2Y/369YKvr6/g7OwsBAcHC0ePHrV0SHZn3Lhxgkw
mE5ycnAQfHx9hzJgxwpUrVywdlt04fPiw3u+J559/XhCE6mn58fHxglQqFVxcXIQBAwYily9fNvpxRIIgCCZJ44iIiIhs
GGefERERERYFJERERERREAJkVERERREAJgUERERERQFgUkRERERQEgEkRERERERQAmRURERERQAmBQRERERAWBSRERERASASRERE
RERACZFRGQnfv/9d0ilUixfvlxT9t///hfOzs7Yv3+/BSMjImvBvc+IyG6kpKRg1KhROHyJLp27YqePXtixIgRWLNmja
VDIyIrwKSIiOzKjBkzcODAATz++++++ePEizpw5A7ELdu3cLZs2fRvXt3dERFaCY4q
IyK788ssvuHPnDqQqqqnDz5k1Lh0NEVoQtRURkN8rLy9GrVy8EBQWha9euLWVqFS5fvgxvb29ZsZjf/7v//D
N998g4sXL1Lh5k1LVqFS5fvgxvb29ZsZjf/7v//D4f90qeL7s18bCAAAAAElFTkSuQmCC\n",
      "text/plain": [
       "<Figure size 640x480 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "fig, ax = plt.subplots()\n",
    "\n",
    "# Settings\n",
    "ax.set_xlabel('x')\n",
    "ax.set_ylabel('probability density')\n",
    "ax.set_title('Random Numbers from Gaussian Distribution')\n",
    "\n",
    "# Plot the random numbers (X-axis) and their probability density (Y-axis)\n",
    "ax.scatter(x, y)\n",
    "\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Task 4. In this task, please change the values mu and sigma to define another Gaussian distribution.\n",
    "# Then, generate 1000 random numbers, calculate their probaiblity density values, and plot as we did in the example above.\n",
    "\n",
    "# Your code here:\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Checkpoint\n",
    "\n",
    "You got your copy of this notebook by forking and then cloning my Github repository. Now that you have made some changes you should commit your work with a suitable commit message and push your changes back to your Github repository. Show your tutor your updated Github repository to get your checkpoint mark."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Task 5. In this task, please change the values mu and sigma to define another Gaussian distribution.\n",
    "# Then, generate 1000 random numbers, calculate their probaiblity density values, and plot as we did in the example above.\n",
    "\n",
    "# Your code here:\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Checkpoint\n",
    "\n",
    "You got your copy of this notebook by forking and then cloning my Github repository. Now that you have made some changes you should commit your work with a suitable commit message and push your changes back to your Github repository. Show your tutor your updated Github repository to get your checkpoint mark."
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "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": "3.8.5"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 4
}

```
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "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": "3.7.10"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```