Unit 2 - Week 1: About computers, Python: Variables and Array

Week 1 Assignment

The due date for submitting this assignment has passed.

Due on 2017-02-07, 23:59 IST.

Submitted assignment (Submitted on 2017-01-28, 15:44)

A computer program multiplies two matrices A[N,N] and B[N,N] and produces C[N,N]. The number of floating point operations required to perform the task is called the computational complexity.

The above statement is valid for first three questions.

1) 1. Total memory required to save the three matrix A[N,N], B[N,N] and C[N,N], for a given N is (Consider Single-precision floating-point format): ***1 point***

- ○ N bytes
- ○ N^2 bytes
- ○ 12N bytes
- ◉ $12N^2$ bytes

2) 2. Computational complexity of the multiplication of the two matrix A[N,N] and B[N,N] for a given N, is of the order of (**Consider Single-precision floating-point format):** ***1 point***

- ○ N
- ○ N^2
- ○ N^3
- ○ N^4

3) 3. Total memory required to save the three matrix *A[N,N], B[N,N]* and *C[N,N],* for a given N, is (**Consider Double-precision floating-point format**): ***1 point***

- ○ N bytes
- ○ N^2 bytes
- ○ $12N^2$ bytes

- $24N^2$ bytes

4) 4. RAM of a typical PC is of the order of *1 point*

- 2-16 GB
- 128-512 GB
- 1-2 TB
- 10 TB

5) 5. Hard disk storage of a typical PC is of the order of *1 point*

- 4-32 GB
- 128 GB-1 TB
- 4-32 TB
- 32-128 TB

6) 6. Number of cores in a typical PC is in the range *1 point*

- 2-16
- 32-128
- 256-512
- More than 1024

7) 7. Flop rating of a typical PC is *1 point*

- 1 megaflops/s
- 1 gigaflops/s
- 100 gigaflops/s
- 1 teraflops/s

8) 8. The binary representation of the decimal number 100 is *1 point*

- 01100100
- 01100101
- 11010100
- 01101100

9) 9. The largest positive integer that can fit in 4 bytes unsigned representation is *1 point*

- 2147483647
- 4294967295
- 2147483648
- 4294967296

10) 10. The binary representation of the decimal number 1.5 stored is *1 point*

- 1.0111
- 1.0110
- 1.0101
- 1.1000

Unit 3 - Week 2:
Python: Control structures, Programming style

Week 2 Assignment

The due date for submitting this assignment has passed.

Due on 2017-02-07, 23:59 IST.

Submitted assignment (Submitted on 2017-01-31, 06:07)

1) 1. In Python program range(4) returns *1 point*

- (selected) [0,1, 2, 3, 4]
- [0,1, 2, 3]
- [4]
- [1, 2, 3, 4]

2) 2. Which among the following statement is true for Python programs? *1 point*

- Variables inside a function is variable outside the function.
- All Python variables are local.
- Python variables have fixed type.
- (selected) Python allows recursive programs.

Previous Page

End

Unit 4 - Week 3: Plotting, Errors, Data input/output

Week 3 Assignment

The due date for submitting this assignment has passed.

Due on 2017-02-14, 23:59 IST.

Submitted assignment

1) The extension of the file name generated from the following program is *1 point*

In [1]: import numpy as np

In [2]: a = 3.2

In [3]: np.save('data',a)

- .h5
- .pdf
- .npy
- .txt

2) What is the output of following program *1 point*

In [1]: import numpy as np

In [2]: x = np.array([1,2,3])

In [3]: y = np.array([4,7,9])

In [4]: z = np.vstack((x,y))

In [5]: print np.shape(z)

- (2, 3)
- (1, 3, 5)
- (1, 3, 9 ,10)
- (3, 9)

3) Which function will you use to plot vectors of a field. *1 point*

- plt.vector()
- plt.imshow()
- plt.quiver()

integration

Week 6: Differentiation, ODE solvers

Week 7: Fourier transforms, PDE solvers

Week 8: Linear Algebra, Summary

Exam Solutions

- plt.figure()

4) Which function will you use to generate density plot of f(x,y) = sin(x) + cos(y). *1 point*

- plt.plot()
- plt.imshow()
- plt.quiver()
- plt.contour()

5) Which function will you use to generate the contour plot of f(x,y) = sin(x) + cos(y). *1 point*

- plt.plot()
- plt.imshow()
- plt.quiver()
- plt.contour()

6) We expand $\sin(x = \pi/4)$ as $x - \frac{x^3}{3!}$ (two terms of the Taylor's series). The relative error in the expansion is *1 point*

- ~ 0.3%
- ~ 3%
- ~ 30%
- ~ 1%

7) We expand $\exp(x = \pi/4)$ as $1 + x + \frac{x^2}{2} + \frac{x^3}{6}$ (four terms of Taylor's series). The relative error in the expansion is *1 point*

- ~ 0.3%
- ~ 0.9%
- ~ 9%
- ~ 15%

8) For the computation of $\frac{x}{y}$ (real division), error is maximum for the following combination *1 point*

- x ~ 1, y ~ 1
- x ~ 1, y ~ 0
- x ~ 0, y ~ 1
- x ~ 100, y ~ 1

9) For the equation $\frac{dx}{dt} = 10x$, the time scale is *1 point*

- 100
- 10
- 0.1
- none of the above

10) For the damped linear oscillator, $\ddot{x} + r\dot{x} + \omega_0^2 x = 0$, the time scales are *1 point*

- r, ω_0^2
- 1/r, $1/\omega_0$
- 1/r, $1/\omega_0^2$

rx, ω_0^2

Previous Page

End

X

NPTEL

noc17-ph02@nptel.iitm.ac.in ▼

Unit 5 - Week 4: Interpolation

Week 4 Assignment

The due date for submitting this assignment has passed.

Due on 2017-02-21, 23:59 IST.

Submitted assignment (Submitted on 2017-02-14, 10:50)

1) An interpolating function based on Lagrange interpolation of three points (x_0, y_0), (x_1, y_1), and (x_2, y_2) is *1 point*

- $\frac{(x-x_1)}{(x_0-x_1)} y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} y_1 + \frac{(x-x_0)}{(x_2-x_0)} y_2$
- $\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} y_2$
- (selected) $\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} y_0 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} y_2$
- $\frac{(x-x_1)}{(x_0-x_1)} y_0 + \frac{(x-x_0)}{(x_1-x_0)} y_1 + \frac{(x-x_0)}{(x_2-x_0)} y_2$

2) Consider a function f(x)=1/x, and two points (1,1) and (2,1/2). The interpolated value of f(x) at x=1.5 is *1 point*

- 3/4
- (selected) 1/2
- 5/9
- 2/19

Previous Page

End

Unit 6 - Week 5: Numerical integration

Week 5 Assignment

The due date for submitting this assignment has passed.

Due on 2017-03-03, 23:59 IST.

Submitted assignment (Submitted on 2017-02-21, 13:55)

1) Error in the value of integral $\int_a^b f(x)dx$ in Trapezoid rule is (h = b-a) *1 point*

- O(h)
- O(h^2)
- O(h^3) (selected)
- O(h^4)

2) $\int_0^{1/2} x^3 dx$ using 3 point Simpson's rule is *1 point*

- 0.007813
- 0.015625 (selected)
- 0.041667
- 0.211325

Previous Page

End

Unit 7 - Week 6: Differentiation, ODE solvers

Week 6 Assignment

The due date for submitting this assignment has passed.

Due on 2017-03-07, 23:59 IST.

Submitted assignment

1) Consider an ODE $\dot{x} = -i2x$, where x is a real number and i = $\sqrt{-1}$. Which is the correct statement for this equation: *1 point*

- Euler’s forward scheme is unconditionally stable
- Euler’s backward scheme is unconditionally stable.
- The amplitude of the exact solution increases with time.
- None of the above

2) Consider an ODE $\dot{x} = -x$. Given initial condition x(0) = 1, and dt = 0.1. For Euler's forward scheme, what is the value at $x(\Delta t)$, i.e. at first step. *1 point*

- 0.9
- 10
- 1.0
- 0.1

Previous Page

End

Unit 8 - Week 7: Fourier transforms, PDE solvers

Week 7 Assignment

The due date for submitting this assignment has passed.

Due on 2017-03-14, 23:59 IST.

Submitted assignment

1) f(x,y) = 4 cos(x) sin(y), then the amplitude of the Fourier mode (1,1) is ***1 point***

- 1
- -1
- i
- -i

2) Consider wave equation $\partial_t \phi + c\partial_x \phi = 0$, where c is a constant. We wish to solve the above PDE numerically using h as the grid spacing. According to the CFL condition, the time-step Δt should satisfy ***1 point***

- Δt < h/c
- Δt < c/h
- Δt < h^2/c
- Δt < h/c^2

Previous Page

End

X

NPTEL

noc17-ph02@nptel.iitm.ac.in ▼

Courses » Computational Science and Engineering using Python

Announcements Course Forum Progress Mentor

Unit 9 - Week 8: Linear Algebra, Summary

Course outline

Week 8 Assignment

The due date for submitting this assignment has passed.

Due on 2017-03-21, 23:59 IST.

Submitted assignment

1) Time complexity for solving AX = b, where A is tridiagonal matrix: ***1 point***

- O(N)
- O(N^2)
- O(N^3)
- O(N^4)

2) ***1 point***

The eigenvalues of matrix A = $\left|\begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 4 \end{bmatrix}\right|$ are

- 4, 4, 4
- 3, 3, 3
- 3, 3, 6
- 1, 1, 1

Previous Page End

MCQ Assignment Week 1

Question	Answer
1	d
2	c
3	d
4	a
5	b
6	a
7	c
8	a
9	b
10	d

MCQ Assignment Week 1

Question	Answer
1	b
2	d

MCQ Assignment Week 3

Question	Answer
1	c
2	a
3	c
4	b
5	d
6	a
7	b
8	b
9	c
10	b

MCQ Assignment Week 4

Question	Answer
1	b
2	a

MCQ Assignment Week 5

Question	Answer
1	c
2	b

MCQ Assignment Week 6

Question	Answer
1	b
2	a

MCQ Assignment Week 7

Question	Answer
1	d
2	a

MCQ Assignment Week 8

Question	Answer
1	a
2	c

NPTEL – Computational Science and Engineering Using Python

Programming Assignment and Exam Solutions

Week 2

Question 1

```
Write a Python program that prints the minimum value from an array of N integers. The integers have to be read and stored in a NumPy array.

Note that the default data type for NumPy arrays is float. In order to evaluate the output of your submission correctly, ensure that your NumPy arrays have been explicitly converted to integer data type. For instance, if x is a NumPy array, convert the array as below, before printing the output:

x = numpy.int32(x)

Please refrain from using built-in functions to find the minimum value of an array. The aim of the problem is to familiarize yourself with the process of constructing a logical sequence for programming tasks.

The input will be of N+1 lines. The first line gives the length of the array, N, followed by N lines each containing a single integer entry for the array.

Eg.

INPUT:
4
7
2
-1
10

OUTPUT:
-1
```

Solution

```
import numpy as np

N = int(raw_input())
inputArray = np.zeros(N)
for i in range(N):
    inputArray[i] = raw_input()

inputArray = np.int32(inputArray)

minimumValue = inputArray[0]
for i in range(N):
    if minimumValue > inputArray[i]:
        minimumValue = inputArray[i]

print minimumValue
```

Question 2

```
Write a Python program to multiply two 3x3 matrices. The input matrices A and B must be constructed with NumPy arrays using two integer values a and b as
shown below:

    A[i][j] = a*i*j
    B[i][j] = b*(i+1)*(j+1)

Note that the matrices should hold integer values and not float. Since the default data type for NumPy arrays is float, convert the arrays as below:

A = numpy.int32(A)

In order to evaluate the output of your submission correctly, please ensure that your NumPy arrays have been explicitly converted as above.

Please refrain from using built-in functions to multiply matrices. The aim of the problem is to familiarize yourself with the process of constructing a
logical sequence for programming tasks.

The input will be of two lines. The first line will contain the value of a, and the second line, b.

Eg.

INPUT:
3
4

OUTPUT:
[[  0   0   0]
 [ 96 192 288]
 [192 384 576]]
```

Solution

```
import numpy as np

a = int(raw_input())
b = int(raw_input())

A = np.zeros((3, 3))
B = np.zeros((3, 3))
C = np.zeros((3, 3))

for i in range(3):
    for j in range(3):
        A[i,j] = a*i*j
        B[i,j] = b*(i+1)*(j+1)

for i in range(3):
    for j in range(3):
        for k in range(3):
            C[i,j] += A[i,k]*B[k,j]

C = np.int32(C)

print(C)
```

Week 4

Question 1

Consider the following table of values for some unknown function $f(x)$

x	f(x)
-2.0	-58.000
-1.6	-27.152
-1.2	-8.016
-0.8	2.096
-0.4	5.872
0.0	6.000
0.4	5.168
0.8	6.064
1.2	11.376
1.6	23.792
2.0	46.000

Write a Python program to interpolate the function $f(x)$ at a given input value, x_0, within the above interval of $(-2, 2)$ using the Lagrange interpolation formula.

The input will consist of a single line containing the value of x_0 such that $-2 \leq x_0 \leq 2$

Print the output value for $f(x_0)$ rounded up to 3 decimal places. You can use the NumPy function round() for this.

Eg.

INPUT:
-1.5

OUTPUT:
-21.375

Solution

```
import numpy as np

x = np.array([-2. , -1.6, -1.2, -0.8, -0.4,  0. ,
               0.4,  0.8,  1.2,  1.6,  2. ])
y = np.array([-58.   , -27.152,  -8.016,   2.096,
               5.872,   6.   ,   5.168,   6.064,
              11.376,  23.792,  46.   ])

x0 = input()
y0 = 0.0

for i in range(6):
    term = 1.0
    for j in range(6):
        if j != i:
            term = term*(x0 - x[j])/(x[i] - x[j])
    y0 += term*y[i]

print np.round(y0, 3)
```

Question 2

Consider a function $z = f(x, y)$ defined within the limits $(1 \leq x \leq 2)$ and $(1 \leq y \leq 2)$. The table below shows the values of z for certain input values of x and y:

	y_1 = 1.00	y_2 = 1.25	y_3 = 1.50	y_4 = 1.75	y_5 = 2.00
x_1 = 1.00	-1.592568	-2.855972	-3.941807	-4.782558	-5.325954
x_2 = 1.25	-2.113761	-3.406294	-4.487041	-5.288806	-5.761738
x_3 = 1.50	-2.503530	-3.744830	-4.753294	-5.466221	-5.839285
x_4 = 1.75	-2.737642	-3.850529	-4.724009	-5.303773	-5.553774
x_5 = 2.00	-2.801541	-3.716822	-4.401009	-4.811562	-4.922956

For instance $f(1.5, 1.75) = -5.466221$, $f(1.25, 2.0) = -5.761738$ and so on.

Use the 2D Lagrange interpolation formula to interpolate the values from the above table to a given input coordinate (x_0, y_0), where $1 \leq x_0, y_0 \leq 2$

The input will consist of 2 lines, with the first and second lines containing the values of x_0 and y_0 respectively.

Print the output value for $z_0 = f(x_0, y_0)$ rounded up to 3 decimal places. You can use the NumPy function round() for this.

Eg.

INPUT:
1.8
1.9

OUTPUT:
-5.405

Solution

```
import numpy as np

Z = np.array([[-1.592568, -2.855972, -3.941807, -4.782558, -5.325954],
              [-2.113761, -3.406294, -4.487041, -5.288806, -5.761738],
              [-2.50353 , -3.74483 , -4.753294, -5.466221, -5.839285],
              [-2.737642, -3.850529, -4.724009, -5.303773, -5.553774],
              [-2.801541, -3.716822, -4.401009, -4.811562, -4.922956]])

x = np.linspace(1, 2, 5)
y = np.linspace(1, 2, 5)

x0 = input()
y0 = input()

z0 = 0.0
for i in range(5):
    for j in range(5):
        xTerm = 1.0
        for k in range(5):
            if k != i:
                xTerm = xTerm*(x0 - x[k])/(x[i] - x[k])
        yTerm = 1.0
```

```
        for k in range(5):
            if k != j:
                yTerm = yTerm*(y0 - y[k])/(y[j] - y[k])

        z0 += Z[i,j]*xTerm*yTerm

print np.round(z0, 3)
```

Week 5

Question 1

Write a Python program to numerically evaluate the following integral using the 5 point Gaussian Quadrature rule:

$\int_0^{\frac{\pi}{8}} \sqrt{\tan ax}dx$

To apply the five-point Gaussian quadrature, use the following table of weights and nodes within the interval $[-1, 1]$. To evaluate the integral within the interval $[0, \frac{\pi}{8}]$, rewrite the function appropriately to change the limits to $[-1, 1]$. Then evaluate at the x_j's given below and multiply with the corresponding weights w_j's.

x_j	w_j
-0.9061798	0.2369269
-0.5384693	0.4786287
0.0	0.5688889
0.5384693	0.4786287
0.9061798	0.2369269

The input will consist of a single line containing the value of a, where $0 < a < 4$

Print the output of your program rounded to 3 decimal places. You may use the function numpy.round() for this.

Eg.

INPUT:
1

OUTPUT:
0.166

Solution

```
import numpy as np

a = 0.0
b = np.pi/8.0
c = np.float64(input())

x = np.array([-0.9061798, -0.5384693, 0.0, 0.5384693, 0.9061798])
w = np.array([0.2369269, 0.4786287, 0.5688889, 0.4786287, 0.2369269])

alph = (b - a)/2.0
beta = (b + a)/2.0
```

```
x = alph*x + beta
f = np.sqrt(np.tan(c*x))

intSum = w*f

intSum = intSum*(b - a)/2.0
print(np.round(sum(intSum), 3))
```

Question 2

Write a Python program to numerically evaluate the following integral using the five-point Gauss-Hermite quadrature:

$\int_{-\infty}^{\infty} e^{-x^2} \cos axdx$

To apply the 5-point Gauss-Hermite quadrature, use the following table of weights and nodes.

x_j	w_j
-2.02018	0.0199532
-0.958572	0.393619
0.0	0.945309
0.958572	0.393619
2.02018	0.0199532

The input will consist of a single line containing the value of a.

Print the output of your program rounded to 3 decimal places. You may use the function numpy.round() for this.

Eg.

INPUT:
1

OUTPUT:
1.38

Solution

```
import numpy as np

a = np.float64(input())

x = np.array([-2.02018, -0.958572 , 0.0, 0.958572, 2.02018])
w = np.array([0.0199532, 0.393619, 0.945309, 0.393619, 0.0199532])

f = np.cos(a*x)
intSum = w*f

print(np.round(sum(intSum), 3))
```

Week 6

Question 1

Write a Python program to evaluate the first derivative, $\frac{dy}{dx}$, of the following function at a given point x_0 using the three-point central difference scheme.

$y = \frac{x^3}{8} - \frac{x^2}{5} + x + 1$

The node spacing required to evaluate the derivative, h will be given as input.

The input will consist of two lines with the first and second lines providing the values of x_0 and h respectively.

Print the output of your program rounded to 3 decimal places. You may use the function numpy.round() for this.

Eg.

INPUT:
2
0.05

OUTPUT:
1.7

Solution

```
import numpy as np

x0 = input()
h = input()

x = np.array([x0 - h, x0, x0 + h])
y = (x**3.0)/8.0 - (x**2.0)/5.0 + x + 1.0

dydx = (y[2] - y[0])/(2.0*h)
print np.round(dydx, 3)
```

Question 2

Write a Python program to solve the Ordinary Differential Equation (ODE) for simple harmonic motion using the second order Runge-Kutta (RK2) method. The equation of motion for simple harmonic oscillator is given by:

$$\frac{d^2x}{dt^2} + x = 0$$

The time-step, dt, to be used in RK2 along with the final time up to which the ODE must be solved, t_{max}, will be provided as input. The program must compute the value of x at t_{max} and print it as output (rounded to 3 decimal places).

The initial conditions at $t = 0$ are:

$x(0) = 0$
$\left.\frac{dx}{dt}\right|_{t=0} = 0.5$

The input will consist of two lines with the first and second lines providing the values of t_{max} and dt respectively.

Print the value of $x(t_{max})$ rounded to 3 decimal places. You may use the function numpy.round() for this.

Eg.

INPUT:
10
0.01

OUTPUT:
-0.272

Solution

```
import numpy as np

tMax = input()
dt = input()

t = 0.0
x = 0.0
v = 0.5

# RK2 time-integration loop.
# Check whether t + (dt/2.0) < tMax, not whether t < tMax
# This is because since t is float, t could be 4.99999 when it should be 5.0
while t + (dt/2.0) < tMax:

    # Evaluate equations at mid-point
    vHalf = v - dt*x/2.0
    xHalf = x + dt*v/2.0

    v = v - dt*xHalf
    x = x + dt*vHalf

    t += dt

print np.round(x, 3)
```

Week 7

Question 1

```
Write a Python program to calculate the Discrete Fourier Transform of the following function:
```

$f(x) = \sin 5x$

Evaluate the above expression over a domain of range **0 to 10**, divided by **256** uniformly spaced points **(including the end points)**. Then use Python's rfft() function from the NumPy module to obtain the Fourier transform of the given function.

Also use the NumPy module's rfftfreq() function to generate a list of corresponding wave-numbers associated with the Fourier amplitudes obtained from rfft(). Don't forget to specify the minimum grid spacing when using rfftfreq(). Otherwise it will use a default spacing leading to incorrect wave-numbers.

Search within the list of wave-numbers for a given input wave-number, k, and locate the corresponding Fourier amplitude for k. Print this value as the output of your program.

The input will consist of a single line containing the value of k.

Print the output of your program rounded to 4 decimal places. You may use the function numpy.round() for this.

Eg.

INPUT:
1.0

OUTPUT:
0.7797

Solution

```
import numpy as np

N = 256
L = 10.0

dx = L/N
x = np.linspace(0.0, L, N)
y = np.sin(5.0*x)

fourier = np.fft.rfft(y)
freq = np.fft.rfftfreq(len(x), dx)

k = input()
fAmpl = fourier[np.where(freq == k)]
realAmpl = np.abs(fAmpl)[0]
print(np.round(realAmpl, 4))
```

Question 2

Write a Python program to solve the 1D diffusion equation numerically. The diffusion equation is written as:

$$\frac{\partial \Psi}{\partial t} = \alpha \frac{\partial^2 \Psi}{\partial x^2}$$

Solve the above PDE over a domain of range -5 to 5 for the case where $\alpha = 1$. Use the following boundary conditions at either ends of the domain:

$$\Psi(-5, t) = 0$$
$$\Psi(5, t) = 0$$

You would have to use the three-point central scheme for calculating the second derivative on the RHS of the above PDE. Use the second order Runge-Kutta method (RK2) for performing time-iteration. Discretize the domain with an appropriate grid spacing, h, and correspondingly, calculate an appropriate value of Δt in order to satisfy the stability condition:

$$\Delta t < \frac{h^2}{2\alpha}$$

Use the following function as initial condition for starting your time-iteration:

$$\Psi(x, 0) = e^{-16x^2}$$

Plot the solution from your simulation at times $t = 0.0, 1.0, 2.0, 3.0, 4.0$ and 5.0, all in **a single frame**. You can do this by calling the matplotlib's plot function repeatedly for each of the above times and then at the end of the simulation, draw the plot into an output file using the prutorlib's plot function.

Note that since this problem is being evaluated according to a single plot, there will be no test cases or inputs for your program to take.

Solution

```
import numpy as np
import prutorlib
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot

N = 500
L = 5.0
x = np.linspace(-L, L, N)

y = np.exp(-16.0*x*x)
yMid = np.zeros(N)

t = 0.0
dx = 2.0*L/N
dt = 0.0002

writeInterval = 0.0

while t < 5.1:
    yMid[1:-1] = y[1:-1] + dt*(y[:-2] - 2.0*y[1:-1] + y[2:])/(2.0*dx*dx)
    yMid[0] = 0.0
    yMid[-1] = 0.0
```

```
    y[1:-1] = y[1:-1] + dt*(yMid[:-2] - 2.0*yMid[1:-1] + yMid[2:])/(dx*dx)
    y[0] = 0.0
    y[-1] = 0.0

    if t + dt/2.0 > writeInterval:
        matplotlib.pyplot.plot(x, y)
        writeInterval += 1.0

    # Increment time
    t += dt

prutorlib.plot(matplotlib.pyplot, "output.png")
```

Week 8

Question 1

Write a Python program to solve the following system of linear equations using the linalg.solve() function from the scipy module.

$3x + 4y - z = a$
$x + 2y + z = b$
$5x - y + 7z = c$

The values for a, b, and c will be provided as input. After computing the solution to the system, (x, y, z), print the value of $x + y + z$ as the output of your program.

The input will consist of 3 lines containing the values of a, b and c in that order.

Print the output of your program rounded to 2 decimal places. You may use the function numpy.round() for this.

Eg.

INPUT:
-1
3
8

OUTPUT:
2.0

Solution

```
import numpy as np
from scipy import linalg

a = input()
b = input()
c = input()

A = np.array([[3, 4, -1], [1, 2, 1], [5, -1, 7]])
```

```
b = np.array([a, b, c])
x = linalg.solve(A, b)

print np.round(sum(x), 2)
```

Question 2

Write a Python program to calculate the eigenvalues of the following symmetric matrix using the linalg.eig() function from the scipy module.

$$A = \begin{bmatrix} 1 & a & b \\ a & 2 & c \\ b & c & 3 \end{bmatrix}$$

The values for a, b, and c will be provided as input. After computing the eigenvalues, x, y, and z, where $x \leq y \leq z$, print the value of $z + y - x$ as the output of your program.

Note that scipy.linalg.eig() calculates complex eigenvalues. However, since the input matrix is symmetric, the computed values will be real, and you will have to discard the complex part before printing the answer.

The input will consist of 3 lines containing the values of a, b and c in that order.

Print the output of your program rounded to 2 decimal places. You may use the function numpy.round() for this.

Eg.

INPUT:
1
2
3

OUTPUT:
6.97

Solution

```
import numpy as np
from scipy import linalg

a = input()
b = input()
c = input()

A = np.array([[1, a, b], [a, 2, c], [b, c, 3]])
eVals, eVecs = linalg.eig(A)
eVals = np.real(eVals)
eVals.sort()

print np.round(eVals[2] + eVals[1] - eVals[0], 2)
```

Exam 1

Question 1

Write a Python program to check if a given input integer is a prime number. If the number is prime, print **0**, else print **1**. You should try to make your program efficient with as few operations as possible.

The input will be an integer, $N < 10^6$.

Print the output as 0 or 1 without any decimals or leading zeros. **[9 Marks]**

Eg.

INPUT:
104729

OUTPUT:
0

Solution

```
import numpy as np

inputNumber = input()

def primalityTest(N):
    if N < 4:
        return 0

    if N % 2 == 0:
        return 1

    testNumber = np.int32(np.sqrt(N))
    for i in range(3, testNumber+1, 2):
        if N % i == 0:
            return 1

    return 0

print primalityTest(inputNumber)
```

Question 2

Consider the following recursive formula which computes x_{n+1} at the (n + 1)th iteration, using the value from the nth iteration, x_n as input.

$$x_{n+1} = rx_n(1 - x_n)$$

Using $r = 2.2$, start with an initial value of $x_0 = 0.31$, and compute the value of x_n through n iterations for a given input value of n.

The input will consist of a single line containing the value of n.

Print the value of x_n rounded to 3 decimal places. You may use the function numpy.round() for this. **[8 Marks]**

Eg.

INPUT:
10

OUTPUT:
0.545

Solution

```
import numpy as np

r = 2.2
x = 0.31

n = input()

for i in range(1, n+1):
    x = r*x*(1 - x)

print np.round(x, 3)
```

Question 3

Write a Python program to solve the following system of linear equations using the linalg.solve() function from the scipy module.

$$ax + y + 2z = 17$$
$$3x + 2by + z = 22$$
$$2x + 3y + 5cz = 43$$

The values for a, b, and c will be provided as input. After computing the solution to the system, (x, y, z), print the value of $x + y + z$ as the output of your program.

The input will consist of 3 lines containing the values of a, b and c in that order.

Print the output of your program rounded to 2 decimal places. You may use the function numpy.round() for this. **[8 Marks]**

Eg.

INPUT:
1
1
1

OUTPUT:
12.0

Solution

```
import numpy as np
from scipy import linalg

a = input()
b = input()
c = input()

A = np.array([[a, 1, 2], [3, 2*b, 1], [2, 3, 5*c]])
b = np.array([17, 22, 43])
x = linalg.solve(A, b)

print np.round(sum(x), 2)
```

Exam 2

Question 1

Write a Python program to solve the following system of linear equations using the linalg.solve() function from the scipy module.

$$ax + 2y + 2z = 11$$
$$3x + 2by + 3z = 16$$
$$3x + 3y + 5cz = 24$$

The values for a, b, and c will be provided as input. After computing the solution to the system, (x, y, z), print the value of $x + y + z$ as the output of your program.

The input will consist of 3 lines containing the values of a, b and c in that order.

Print the output of your program rounded to 2 decimal places. You may use the function numpy.round() for this. **[8 Marks]**

Eg.

```
INPUT:
1
1
1

OUTPUT:
6.0
```

Solution

```
import numpy as np
from scipy import linalg

a = input()
b = input()
c = input()

A = np.array([[a, 2, 2], [3, 2*b, 3], [3, 3, 5*c]])
b = np.array([11, 16, 24])
x = linalg.solve(A, b)

print np.round(sum(x), 2)
```

Question 2

```
Write a Python program to print the second largest entry from an array of N integers.

The input will be of N+1 lines. The first line gives the length of the array, N, followed by N lines each containing a single integer entry for the array.

Print the output as an integer without any decimals or leading zeros. [9 Marks]

Eg.

INPUT:
4
7
2
-1
10

OUTPUT:
7
```

Solution

```
import numpy as np

N = input()
intArray = np.zeros(N)

maxNum = 0
secNum = 0
for i in range(N):
    inpNum = input()
    if inpNum > maxNum:
        secNum = maxNum
        maxNum = inpNum
    elif inpNum < maxNum and inpNum > secNum:
        secNum = inpNum

print secNum
```

Question 3

Write a Python program to evaluate the second derivative, $\frac{d^2y}{dx^2}$, of the following function at a given point x_0 using the three-point central difference scheme.

$y = \frac{x^3}{2} - \frac{x^2}{3} + x + 1$

The node spacing required to evaluate the derivative, h will be given as input.

The input will consist of two lines with the first and second lines providing the values of x_0 and h respectively.

Print the output of your program rounded to 3 decimal places. You may use the function numpy.round() for this. **[8 Marks]**

Eg.

```
INPUT:
0.5
0.01

OUTPUT:
0.833
```

Solution

```
import numpy as np

x0 = input()
h = input()

x = np.array([x0 - h, x0, x0 + h])
y = (x**3.0)/2.0 - (x**2.0)/3.0 + x + 1.0

dydx = (y[2] - 2.0*y[1] + y[0])/(h*h)
print np.round(dydx, 3)
```