Lab 07 - Sequences and Series

Overview

In this lab, we will use SageMath to determine the convergence or divergence of a sequence of numbers and of infinite series.

Important SageMath Commands Introduced in this Lab

Command	Description	Example
$\textbf{point}((x,y),\ldots)$	Plots the point (x,y)	$\textbf{point}((1,1), color =$" red "$)$
$\textbf{sum}(expr, var, a, b)$	Evaluates the sum $\sum_{var=a}^{b} expr$	$\textbf{sum}(1/k^2, k, 1, infinity)$
$\textbf{factorial}(n)$	Calculates $n!$	$\textbf{factorial}(5)$

Related Course Material

```
Sections 10.1 and 10.2
```

Example 1

Consider the sequence $\{\cos(n\pi)\arctan(n)\}_{n=1}^{\infty}$. We start by determining the first 10 terms of this sequence. We can do this in SageMath by letting $a_n = \cos(n\pi)\arctan(n)$ and then using a **for** loop and the **range** command.

In [1]:
```
n=var('n')
def a(n):
    return cos(n * pi) * arctan(n)
for i in range(1,11):     ## This will iterate i through the integers 1 through 10
    print(a(i))
```

```
-1/4*pi
arctan(2)
-arctan(3)
arctan(4)
-arctan(5)
arctan(6)
-arctan(7)
arctan(8)
-arctan(9)
arctan(10)
```

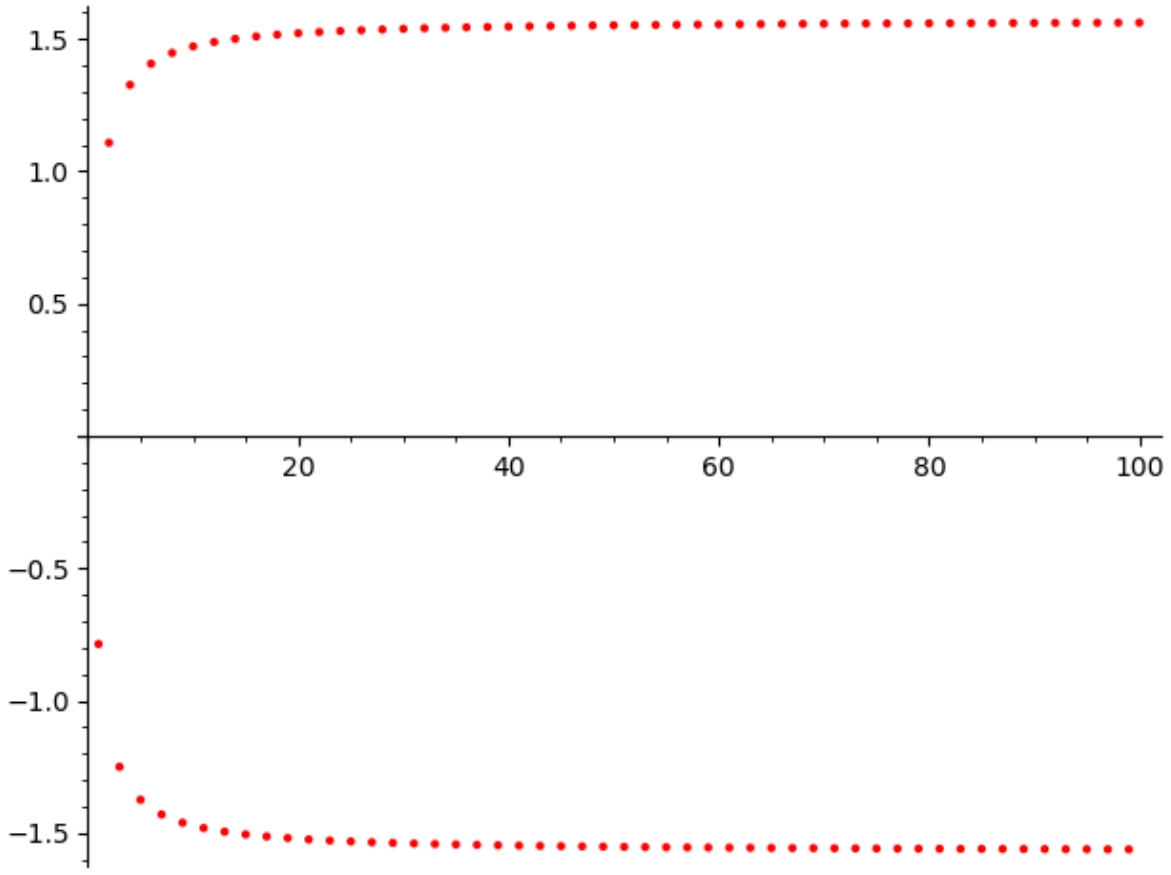
We can get a better idea of what these numbers are by using the **round** command.

In [2]:
```
for i in range(1,11):
    print(round(a(i),5))
```

```
-0.7854
1.10715
-1.24905
1.32582
-1.3734
1.40565
-1.4289
1.44644
-1.46014
1.47113
```

Note that the terms of the sequence do not appear to approach a specific number. We can better tell what is happening by plotting the first 100 or so terms of the sequence. We can plot a point in SageMath by using the **point** command along with the **show** command. To plot multiple points on the same plot, we will store the points in a list and then show the list. SageMath does not allow us to plug the list directly into the **show** command. Instead, we must input the sum of the elements in the list.

In [3]:
```
points = []     ## Creates an empty list to hold our points
for i in range(1,101):
    p = point((i,a(i)), color = "red")
    points.append(p)
show(sum(points))     ## Plugging the sum of the points into the show command
```



From the graph, we see that the odd terms are approaching a specific value, namely $-\frac{\pi}{2}$, and the even terms are approaching a specfic value, namely $\frac{\pi}{2}$. However, since these values are different, the sequence diverges.

We can check our answer in SageMath.

In [4]:
```
limit(a(n), n=infinity)
```

Out[4]:
```
ind
```

The output ind for a limit means that the answer is indefinite but the terms are bounded.

Example 2

Consider the sequence $\left\{\sum_{k=1}^{n} \frac{1}{k^2}\right\}_{n=1}^{\infty}$. Again, let us find the first 10 terms of this sequence.

In order to define a_n, we need to recall how to define a summation in SageMath. One way to do this is to place all of the summands into a list and then use the **sum** command. Let's practice this with $n = 3$. We know that when $n = 3$, we should get

$$\sum_{k=1}^{3} \frac{1}{k^2} = 1 + \frac{1}{4} + \frac{1}{9} = \frac{49}{36}.$$

In [5]:
```
summands = []
for k in range(1,4):     ## Remember that our second input in range needs to be one
    summands.append(1/k^2)
sum(summands)
```

Out[5]:
```
49/36
```

Now that we know how to define the summation in SageMath, we can create the sequence a_n.

In [6]:
```
def a(n):
    summands = []
    for k in range(1, n+1):
        summands.append(1/k^2)
    return sum(summands)
a(3)
```

Out[6]:
```
49/36
```

Now, let's use our definition of a_n to find the first 10 terms of the sequence.

In [7]:
```
for i in range(1,11):
    print(a(i))
```

```
1
5/4
49/36
205/144
5269/3600
5369/3600
266681/176400
1077749/705600
9778141/6350400
1968329/1270080
```

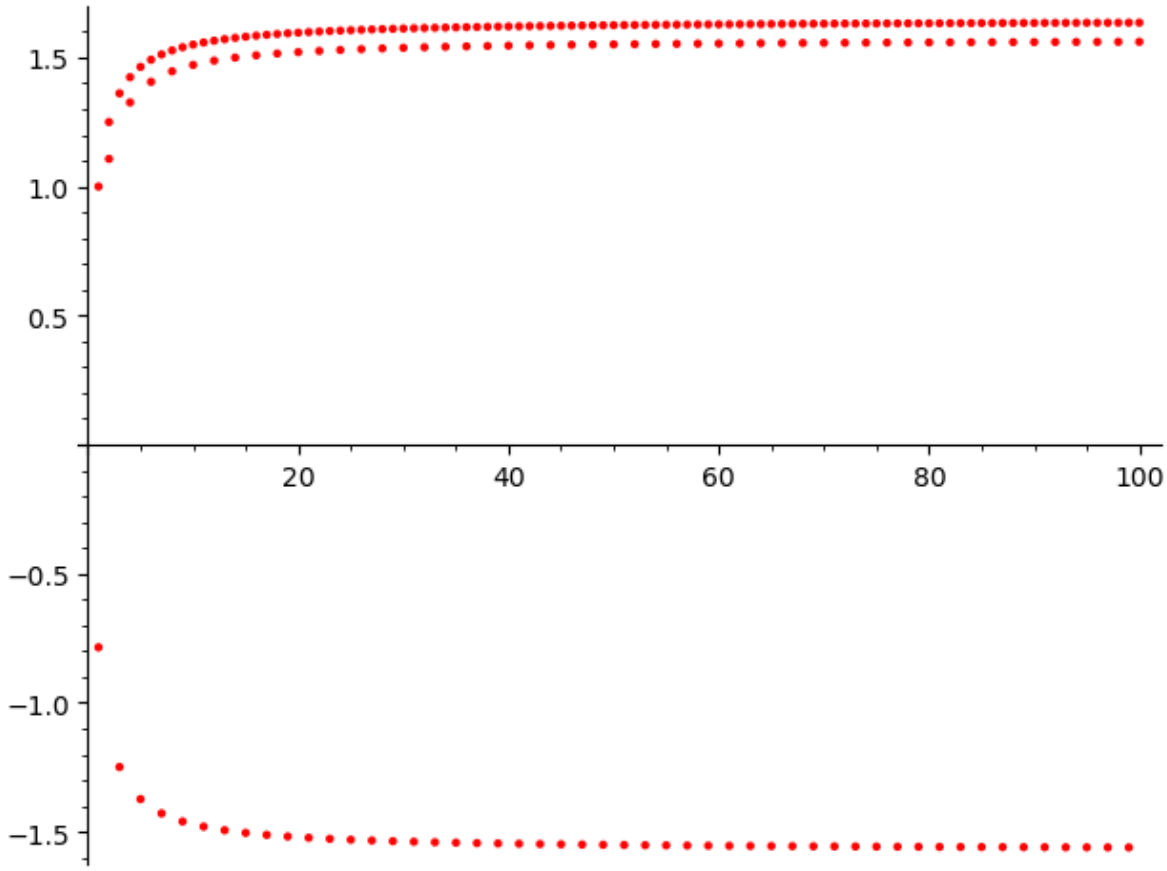
To better see if these number are approaching a specific number, we can round.

In [8]:
```
round(a(i))
```

Out[8]:
```
2
```

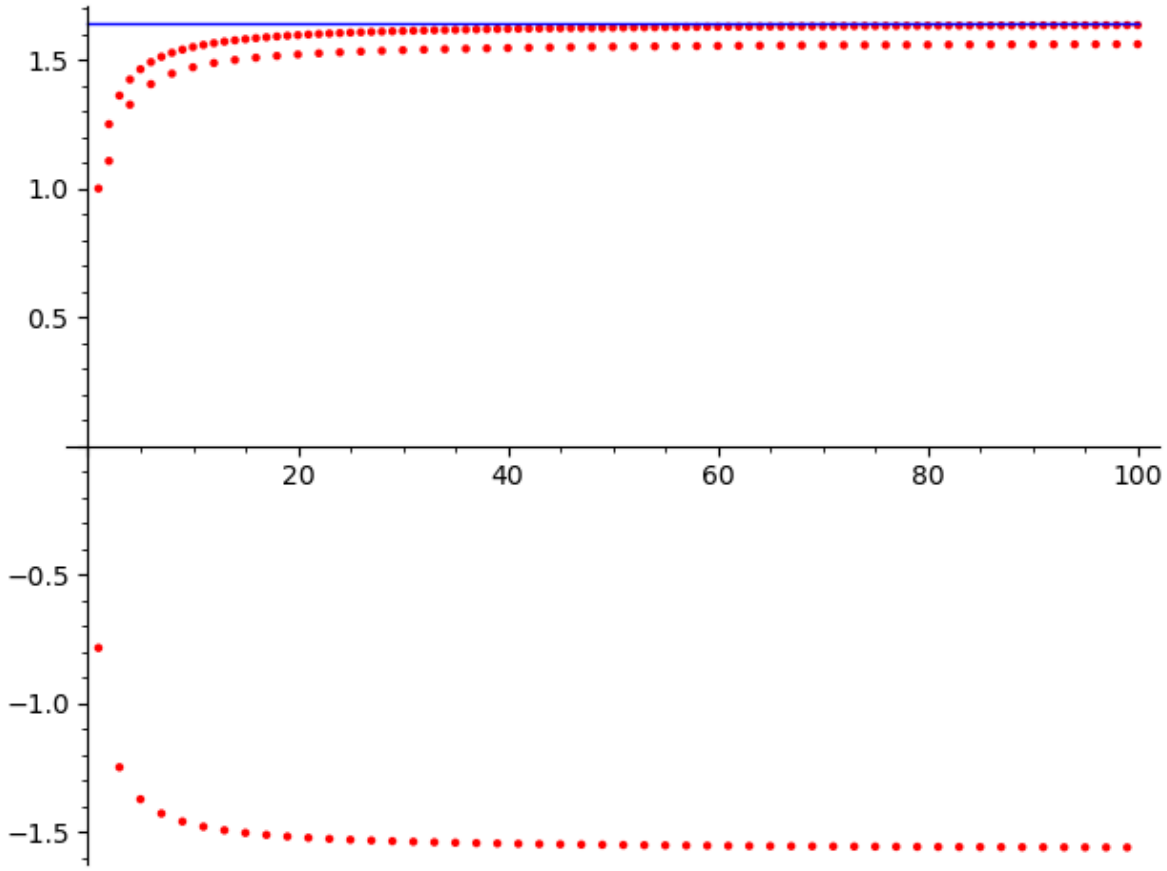
It looks like the terms are getting closer together. Let's plot the first 100 terms and see.

In [9]:
```
for i in range(1,101):
    p = point((i,a(i)), color = "red")
    points.append(p)
show(sum(points))
```



According to the graph, these terms are converging to a specific number. In fact, they are converging to $\frac{\pi^2}{6}$.

In [10]:
```
line = plot(pi^2/6, xmin = 0, xmax = 100)
show(sum(points) + line)     ## You may have to change the name of the list points
```



SageMath has the ability to tell us that this is indeed what the sequence converges to. Unfortunately, it will not work with our current definition of a_n.

In [11]:
```
limit(a(n),n=infinity)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File /usr/local/sage/src/sage/symbolic/expression.pyx:1513, in sage.symbolic.expre
ssion.Expression._integer_()
   1512 try:
-> 1513     n = self.pyobject()
   1514 except TypeError:

File /usr/local/sage/src/sage/symbolic/expression.pyx:774, in sage.symbolic.expres
sion.Expression.pyobject()
    773 if not is_a_numeric(self._gobj):
--> 774     raise TypeError("self must be a numeric expression")
    775 return py_object_from_numeric(self._gobj)

TypeError: self must be a numeric expression

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In [11], line 1
----> 1 limit(a(n),n=infinity)

Cell In [6], line 3, in a(n)
      1 def a(n):
      2     summands = []
----> 3     for k in range(Integer(1), n+Integer(1)):
      4         summands.append(Integer(1)/k**Integer(2))
      5     return sum(summands)

File /usr/local/sage/src/sage/symbolic/expression.pyx:6504, in sage.symbolic.expre
ssion.Expression.__index__()
   6502         [0, 1, 2, 3, 4]
   6503     """
-> 6504     return int(self._integer_())
   6505
   6506 def iterator(self):

File /usr/local/sage/src/sage/symbolic/expression.pyx:1515, in sage.symbolic.expre
ssion.Expression._integer_()
   1513     n = self.pyobject()
   1514 except TypeError:
-> 1515     raise TypeError("unable to convert %r to an integer" % self)
   1516 if isinstance(n, sage.rings.integer.Integer):
   1517     return n

TypeError: unable to convert n + 1 to an integer
```

To fix this, we can use the **sum(expr, var, a, b)** command to define our summation instead of a **for** loop. Let's first test this command with $n = 3$ and make sure that we get $\frac{49}{36}$ as expected.

In [12]:
```
k = var('k')
sum(1/k^2, k, 1, 3)
```

Out[12]: 49/36

Therefore, we will use the **sum** command and let $b = \infty$.

In [13]:
```
show(sum(1/k^2, k, 1, infinity))
```

$$\frac{1}{6}\,\pi^2$$

Example 3

A typical format for a recursively defined sequence is $a_{n+1} = f(a_n)$ for $n = 1, 2, 3, \ldots$ with a_1 given explicitly. Under the assumptions that $\{a_n\}$ converges to L and f is a continuous function, we have that

$$L = \lim_{n\to\infty} a_{n+1} = \lim_{n\to\infty} f(a_n) = f\left(\lim_{n\to\infty} a_n\right) = f(L).$$

Therefore, L must be a solution to $f(L) = L$. This equation is often difficult to solve by hand, but we can use SageMath to find a solution.

Consider the recursive sequence $\{a_n\}$ defined by $a_1 = \sqrt{2}$ and $a_{n+1} = \sqrt{2 + a_n}$. Let's determine the first 10 terms of this sequence.

In [14]:
```
a = [sqrt(2)]     ## Defining the first term of our sequence
for i in range(1,10):     ## Using a for loop to find a_2 through a_10
    newTerm = sqrt(2 + a[i-1])     ## We do a[i-1] since the index of a list starts
    print(newTerm)                 ## Prints the term a_(i+1)
    a.append(newTerm)              ## Adds the term to the list so that it can be u
```

```
sqrt(sqrt(2) + 2)
sqrt(sqrt(sqrt(2) + 2) + 2)
sqrt(sqrt(sqrt(sqrt(2) + 2) + 2) + 2)
sqrt(sqrt(sqrt(sqrt(sqrt(2) + 2) + 2) + 2) + 2)
sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(2) + 2) + 2) + 2) + 2) + 2)
sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(2) + 2) + 2) + 2) + 2) + 2) + 2)
sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(2) + 2) + 2) + 2) + 2) + 2) + 2) + 2)
sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(2) + 2) + 2) + 2) + 2) + 2) + 2) + 2)
+ 2)
sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(sqrt(2) + 2) + 2) + 2) + 2) + 2) + 2)
+ 2) + 2) + 2)
```

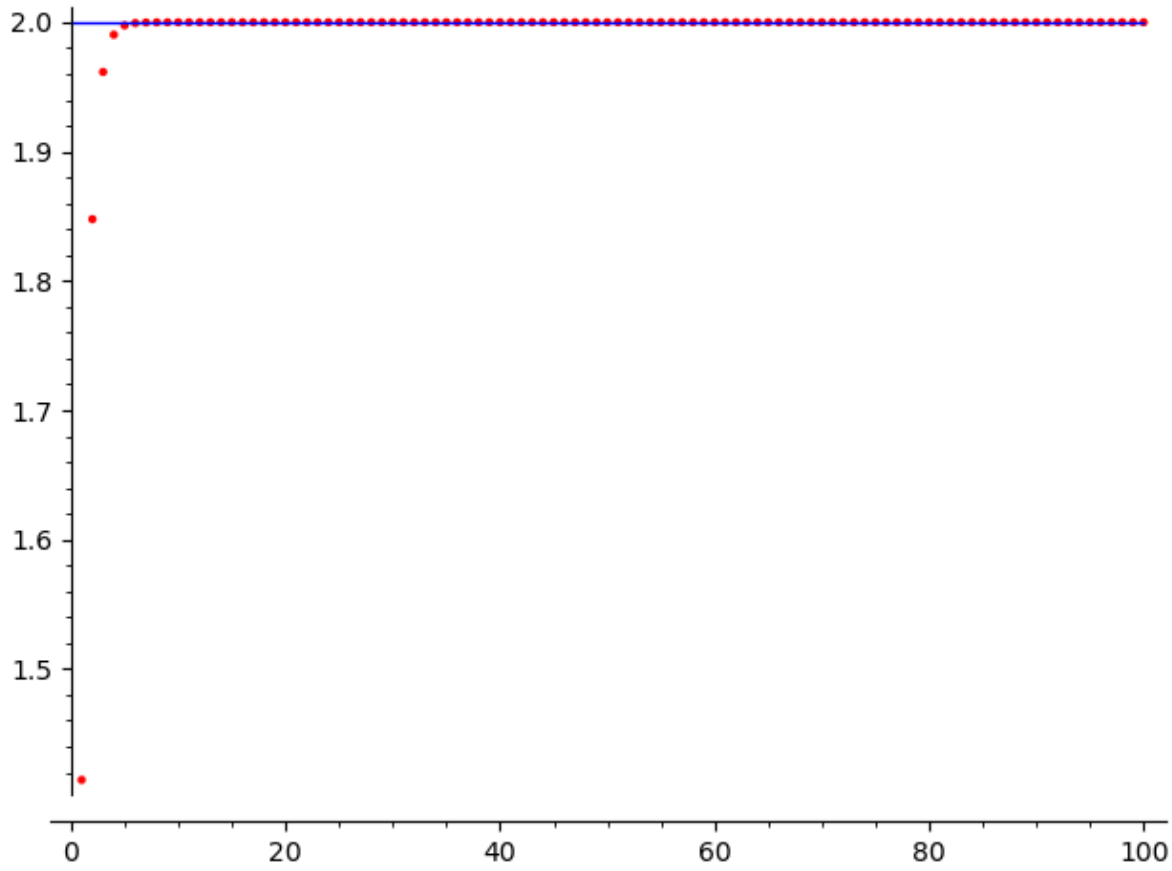
We can round our outputs by either using the **round** command or by starting with $\sqrt{2.0}$ instead of $\sqrt{2}$.

In [15]:
```
a = [sqrt(2.0)]
for i in range(1,10):
    newTerm = sqrt(2 + a[i-1])
    print(newTerm)
    a.append(newTerm)
```

```
1.84775906502257
1.96157056080646
1.99036945334439
1.99759091241034
1.99939763739241
1.99984940367829
1.99996235056520
1.99999058761915
1.99999764690340
```

It appears that the terms are converging to 2. First, let's check this by plotting the first 100 terms and the line $y = 2$.

In [16]:
```
a = [sqrt(2.0)]
for i in range(1,100):
    newTerm = sqrt(2 + a[i-1])
    a.append(newTerm)

points = []
for i in range(1,101):
    p = point((i,a[i-1]), color = "red")      ## Again we have to use a[i-1] because
    points.append(p)

line = plot(2, xmin=0, xmax = 100)

show(sum(points) + line)
```



A second way to verify that the sequence converges to 2 is to solve the equation $f(L) = L$; that is, we need to solve the equation $\sqrt{2+L} = L$. We can solve this in SageMath by using the **solve** command.

In [17]:
```
L = var('L')
solve(sqrt(2 + L) == L, L)
```

Out[17]:
```
[L == sqrt(L + 2)]
```

Unfortunately, SageMath offers no help with the equation in its current state. However, we can simplify the equation a bit by squaring both sides and obtaining $2 + L = L^2$. Now, we can ask SageMath to solve this equation.

In [18]:
```
solve (2 + L == L^2, L)
```

Out[18]:
```
[L == 2, L == -1]
```

SageMath returns the two answers $L = 2$ and $L = -1$. We know that $a_{n+1} = \sqrt{2 + a_n}$ will always be positive. Thus, the solution $L = -1$ is not possible. Therefore, we find that the sequence converges to $L = 2$.

Example 4

For each of the following sequences generate the first 10 terms, plot the first 100 terms, and determine whether the sequence converges or diverges. If it converges, determine the exact value it converges to.

1. $\left\{\sqrt{n^2+n} - n\right\}_{n=1}^{\infty}$
2. $\left\{n \sin\left(\frac{\pi}{n}\right)\right\}_{n=1}^{\infty}$
3. $\sum_{n=1}^{\infty} \frac{1}{n^4}$
4. $\sum_{n=0}^{\infty} \frac{10^n}{n!}$ You need to use **factorial(n)** for n!

In [24]:
```
n=var('n')
def a(n):
    return sqrt(n^(2)+n)-n
for i in range(1,101):
    print(a(i))

limit(a(n), n=infinity)
```

sqrt(2) - 1
sqrt(6) - 2
2*sqrt(3) - 3
2*sqrt(5) - 4
sqrt(30) - 5
sqrt(42) - 6
2*sqrt(14) - 7
6*sqrt(2) - 8
3*sqrt(10) - 9
sqrt(110) - 10
2*sqrt(33) - 11
2*sqrt(39) - 12
sqrt(182) - 13
sqrt(210) - 14
4*sqrt(15) - 15
4*sqrt(17) - 16
3*sqrt(34) - 17
3*sqrt(38) - 18
2*sqrt(95) - 19
2*sqrt(105) - 20
sqrt(462) - 21
sqrt(506) - 22
2*sqrt(138) - 23
10*sqrt(6) - 24
5*sqrt(26) - 25
3*sqrt(78) - 26
6*sqrt(21) - 27
2*sqrt(203) - 28
sqrt(870) - 29
sqrt(930) - 30
4*sqrt(62) - 31
4*sqrt(66) - 32
sqrt(1122) - 33
sqrt(1190) - 34
6*sqrt(35) - 35
6*sqrt(37) - 36
sqrt(1406) - 37
sqrt(1482) - 38
2*sqrt(390) - 39
2*sqrt(410) - 40
sqrt(1722) - 41
sqrt(1806) - 42
2*sqrt(473) - 43
6*sqrt(55) - 44
3*sqrt(230) - 45
sqrt(2162) - 46
4*sqrt(141) - 47
28*sqrt(3) - 48
35*sqrt(2) - 49
5*sqrt(102) - 50
2*sqrt(663) - 51
2*sqrt(689) - 52
3*sqrt(318) - 53
3*sqrt(330) - 54
2*sqrt(770) - 55
2*sqrt(798) - 56

```
sqrt(3306) - 57
sqrt(3422) - 58
2*sqrt(885) - 59
2*sqrt(915) - 60
sqrt(3782) - 61
3*sqrt(434) - 62
24*sqrt(7) - 63
8*sqrt(65) - 64
sqrt(4290) - 65
sqrt(4422) - 66
2*sqrt(1139) - 67
2*sqrt(1173) - 68
sqrt(4830) - 69
sqrt(4970) - 70
6*sqrt(142) - 71
6*sqrt(146) - 72
sqrt(5402) - 73
5*sqrt(222) - 74
10*sqrt(57) - 75
2*sqrt(1463) - 76
sqrt(6006) - 77
sqrt(6162) - 78
4*sqrt(395) - 79
36*sqrt(5) - 80
9*sqrt(82) - 81
sqrt(6806) - 82
2*sqrt(1743) - 83
2*sqrt(1785) - 84
sqrt(7310) - 85
sqrt(7482) - 86
2*sqrt(1914) - 87
2*sqrt(1958) - 88
3*sqrt(890) - 89
3*sqrt(910) - 90
2*sqrt(2093) - 91
2*sqrt(2139) - 92
sqrt(8742) - 93
sqrt(8930) - 94
4*sqrt(570) - 95
4*sqrt(582) - 96
7*sqrt(194) - 97
21*sqrt(22) - 98
30*sqrt(11) - 99
10*sqrt(101) - 100
```

Out[24]: 1/2

In [1]:
```
n=var('n')
def a(n):
    return n*(sin(pi/2))
for i in range(1,101):
    print(a(i))

limit(a(n), n=infinity)
```

```
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Out[1]:
```
+Infinity
```

In [8]:
```
def a(n):
    summands = []
    for k in range(1, 11):
        summands.append(1/a^4)
sum(summands)
```

Out[8]:
```
0
```

In [9]:
```
def a(n):
    summands = []
    for k in range(1, 11):
        summands.append(1/a^10)
sum(summands)
```

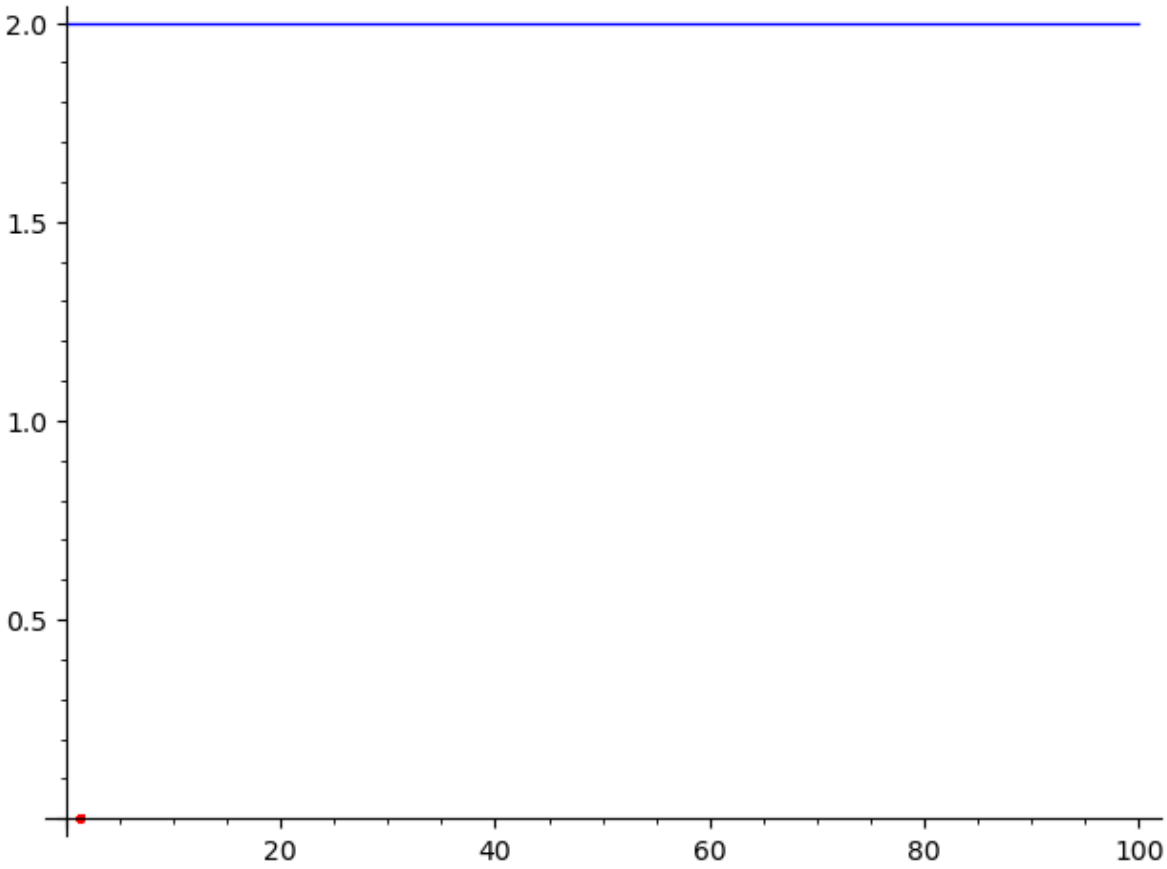
Out[9]:
```
0
```

Example 5

Consider the recursive sequence $a_{n+1} = \frac{1}{2}\left(a_n + \frac{2}{a_n}\right)$, where $a_i = 1$. Generate the first 10 terms, plot the first 100 terms, and verify that the sequence converges to $\sqrt{2}$.

In [15]:
```
a = [sqrt(2)]     ## Defining the first term of our sequence
for i in range(1,10):     ## Using a for loop to find a_2 through a_10
    newTerm = 0.5 * (a[-1] + 2 / a[-1])
    print(newTerm)
    a.append(newTerm)

a = [sqrt(2.0)]
for i in range(1,100):
    newTerm = 0.5 * (a[-1] + 2 / a[-1])
    a.append(newTerm)

points = []
for i in range(1,101):
    p = point(0.5 * (a[-1] + 2 / a[-1]), color = "red")     ## Again we have to use
    points.append(p)

line = plot(2, xmin=0, xmax = 100)

show(sum(points) + line)

L = var('L')
solve(sqrt(2 + L) == L, L)


solve (2 + L == L^2, L)
```

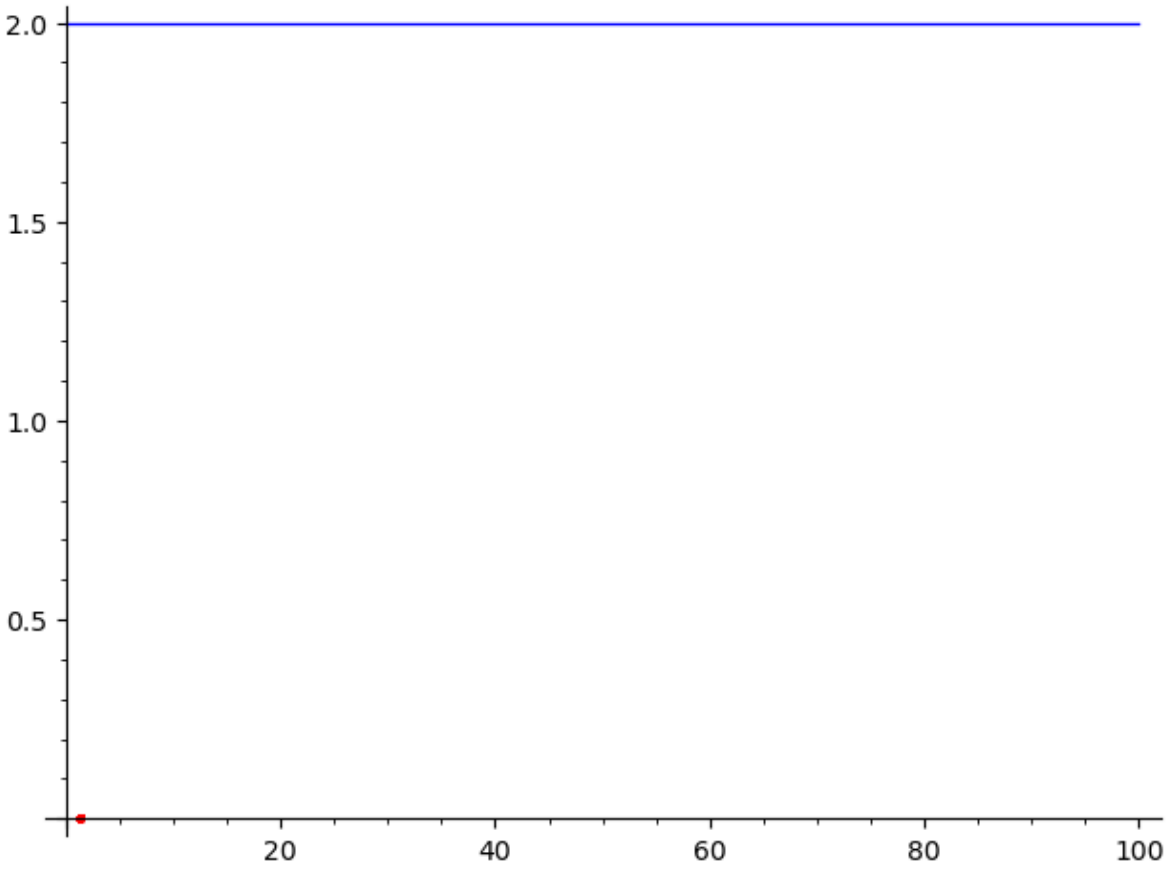
```
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
```



Out[15]: [L == 2, L == -1]

In [16]:
```
a = [sqrt(2)]     ## Defining the first term of our sequence
for i in range(1,10):     ## Using a for loop to find a_2 through a_10
    newTerm = 0.5 * (a[-1] + 2 / a[-1])
    print(newTerm)
    a.append(newTerm)

a = [sqrt(2.0)]
for i in range(1,100):
    newTerm = 0.5 * (a[-1] + 2 / a[-1])
    a.append(newTerm)

points = []
for i in range(1,101):
    p = point(0.5 * (a[-1] + 2 / a[-1]), color = "red")     ## Again we have to use
    points.append(p)

line = plot(2, xmin=0, xmax = 100)

show(sum(points) + line)

L = var('L')
solve(sqrt(2 + L) == L, L)


solve (2 + L == L^2, L)
```

```
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
1.00000000000000*sqrt(2)
```



Out[16]: `[L == 2, L == -1]`

In [ ]: