求无理数的近似值

刘宸煊工业出品

2021
04/01

目录

CONTENT

2021
04/01

PART O1

一般方法

刘宸煊工业出品

2021
04/01

01

一般方法

古时候，人们使用几何方法（即割圆法）来计算圆周率
现在人们可以运用泰勒级数、连分式、蒙特卡罗法、傅立叶级数法来计算无理数近似值

PART O2

蒙特卡洛法

刘宸煊工业出品

2021
04/01

蒙特卡罗法

蒙特卡罗(Monte Carlo) 求单位圆的面积是求取的近似值的关键.其可采用下面的近似计算方法.在单位正方形内随机地投入很多的点且每个点落在单位正方形的机会相等.将落在 单位圆内的点数m 与单位正方形内的点数n 作比，可以获得单位圆面积的近似值.

E:\个人文件\效果\pi.py - Sublime Text (UNREGISTERED)

文件(F) 编辑(E) 选择(S) 查找(I) 视图(V) 跳转(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

临时试验.py | 危险的病毒.bat | index.html | pi.py | index.py | skrollr.min.js | *REPL* [python] | easygui.py | __init__.py | 表情包文字.py | 做图.py | untitled

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
# 投点次数
n = 1
# 圆的半径、圆心
r = 1.0
a,b = (0.,0.)
# 正方形区域
x_min, x_max = a-r, a+r
y_min, y_max = b-r, b+r
# 在正方形区域内随机投点
x = np.random.uniform(x_min, x_max, n) #均匀分布
y = np.random.uniform(y_min, y_max, n)
#计算点到圆心的距离
d = np.sqrt((x-a)**2 + (y-b)**2)
#统计落在圆内点的数目
res = sum(np.where(d < r, 1, 0))
#计算pi的近似值（Monte Carlo:用统计值去近似真实值）
pi = 4 * res / n
print('pi: ',pi)
#画个图
fig = plt.figure()
```

```
pi:  4.0
[Finished in 2.5s]
```

Line 5, Column 6; Build finished

Tab Size: 4 Python

E:\个人文件\效果\pi.py - Sublime Text (UNREGISTERED)

文件(F) 编辑(E) 选择(S) 查找(I) 视图(V) 跳转(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

临时试验.py | 危险的病毒.bat | index.html | pi.py | index.py | skrollr.min.js | *REPL* [python] | easygui.py | __init__.py | 表情包文字.py | 做图.py | untitled

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
# 投点次数
n = 1000
# 圆的半径、圆心
r = 1.0
a,b = (0.,0.)
# 正方形区域
x_min, x_max = a-r, a+r
y_min, y_max = b-r, b+r
# 在正方形区域内随机投点
x = np.random.uniform(x_min, x_max, n) #均匀分布
y = np.random.uniform(y_min, y_max, n)
#计算点到圆心的距离
d = np.sqrt((x-a)**2 + (y-b)**2)
#统计落在圆内点的数目
res = sum(np.where(d < r, 1, 0))
#计算pi的近似值（Monte Carlo:用统计值去近似真实值）
pi = 4 * res / n
print('pi: ',pi)
#画个图
fig = plt.figure()
```

```
pi:  3.136
[Finished in 2.8s]
```

Line 5, Column 9; Build finished

Tab Size: 4 Python

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
# 投点次数
n = 10000
# 圆的半径、圆心
r = 1.0
a,b = (0.,0.)
# 正方形区域
x_min, x_max = a-r, a+r
y_min, y_max = b-r, b+r
# 在正方形区域内随机投点
x = np.random.uniform(x_min, x_max, n) #均匀分布
y = np.random.uniform(y_min, y_max, n)
#计算点到圆心的距离
d = np.sqrt((x-a)**2 + (y-b)**2)
#统计落在圆内点的数目
res = sum(np.where(d < r, 1, 0))
#计算pi的近似值（Monte Carlo:用统计值去近似真实值）
pi = 4 * res / n
print('pi: ',pi)
#画个图
fig = plt.figure()
```

```
pi:  3.1096
[Finished in 3.6s]
```

E:\个人文件\效果\pi.py - Sublime Text (UNREGISTERED)

文件(F) 编辑(E) 选择(S) 查找(I) 视图(V) 跳转(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

临时试验.py | 危险的病毒.bat | index.html | pi.py | index.py | skrollr.min.js | *REPL* [python] | easygui.py | __init__.py | 表情包文字.py | 做图.py | untitled

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
# 投点次数
n = 100000
# 圆的半径、圆心
r = 1.0
a,b = (0.,0.)
# 正方形区域
x_min, x_max = a-r, a+r
y_min, y_max = b-r, b+r
# 在正方形区域内随机投点
x = np.random.uniform(x_min, x_max, n) #均匀分布
y = np.random.uniform(y_min, y_max, n)
#计算点到圆心的距离
d = np.sqrt((x-a)**2 + (y-b)**2)
#统计落在圆内点的数目
res = sum(np.where(d < r, 1, 0))
#计算pi的近似值（Monte Carlo:用统计值去近似真实值）
pi = 4 * res / n
print('pi: ',pi)
#画个图
fig = plt.figure()
```

```
pi:  3.1318
[Finished in 2.9s]
```

Line 5, Column 11; Build finished

Tab Size: 4 Python

E:\个人文件\效果\pi.py - Sublime Text (UNREGISTERED)

文件(F) 编辑(E) 选择(S) 查找(I) 视图(V) 跳转(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

临时试验.py | 危险的病毒.bat | index.html | pi.py | index.py | skrollr.min.js | *REPL* [python] | easygui.py | __init__.py | 表情包文字.py | 做图.py | untitled

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
# 投点次数
n = 1000000
# 圆的半径、圆心
r = 1.0
a,b = (0.,0.)
# 正方形区域
x_min, x_max = a-r, a+r
y_min, y_max = b-r, b+r
# 在正方形区域内随机投点
x = np.random.uniform(x_min, x_max, n) #均匀分布
y = np.random.uniform(y_min, y_max, n)
#计算点到圆心的距离
d = np.sqrt((x-a)**2 + (y-b)**2)
#统计落在圆内点的数目
res = sum(np.where(d < r, 1, 0))
#计算pi的近似值（Monte Carlo:用统计值去近似真实值）
pi = 4 * res / n
print('pi: ',pi)
#画个图
fig = plt.figure()
```

```
pi:  3.141828
[Finished in 3.2s]
```

Line 5, Column 12; Build finished

Tab Size: 4 Python

E:\个人文件\效果\pi.py - Sublime Text (UNREGISTERED)

文件(F) 编辑(E) 选择(S) 查找(I) 视图(V) 跳转(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

临时试验.py | 危险的病毒.bat | index.html | pi.py | index.py | skrollr.min.js | *REPL* [python] | easygui.py | __init__.py | 表情包文字.py | 做图.py | untitled

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
# 投点次数
n = 100000000
# 圆的半径、圆心
r = 1.0
a,b = (0.,0.)
# 正方形区域
x_min, x_max = a-r, a+r
y_min, y_max = b-r, b+r
# 在正方形区域内随机投点
x = np.random.uniform(x_min, x_max, n) #均匀分布
y = np.random.uniform(y_min, y_max, n)
#计算点到圆心的距离
d = np.sqrt((x-a)**2 + (y-b)**2)
#统计落在圆内点的数目
res = sum(np.where(d < r, 1, 0))
#计算pi的近似值（Monte Carlo:用统计值去近似真实值）
pi = 4 * res / n
print('pi: ',pi)
#画个图
fig = plt.figure()
```

```
pi:  3.14166316
```

Line 8, Column 14 Tab Size: 4 Python

感谢您的收看

刘宸煊工业出品

2021
04/01