Estimando o valor de π usando Python com a técnica de Arquimedes

Reginaldo Demarque
Universidade Federal Fluminense
Departamento de Ciências da Natureza

Por volta de 250 BCE o matemático grego Arquimedes estimou a área do círculo usando o chamado Método de Exaustão de Eudoxo, veja [4]. Com isso ele acabou fornecendo algorítimo para estimar o valor de π.

Iniciando-se com um hexágonos inscritos e circunscritos no círculo, depois dobrando-se sucessivamente seus lados, Arquimedes estimou valor de π com polígonos de 96 lados. Com isso mostrou que

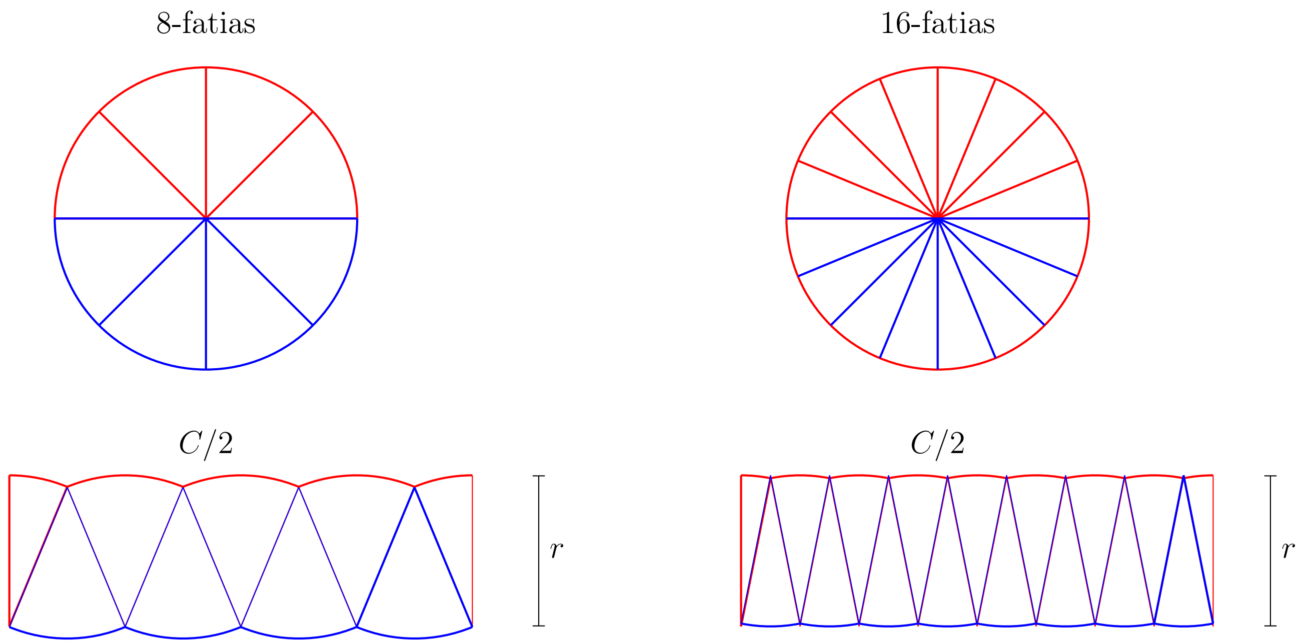
$$\frac{223}{71} < \pi < \frac{22}{7},$$

isto é, chegou a um valor aproximado de π entre 3.1408 e 3.1429.

O objetivo deste texto é explicar como Arquimedes obteve este resultado. Para isso vamos usar a linguagem de python para efetuar os cálculos numéricos.

1 Relação entre a área e a circunferência

Inicialmente, vamos deduzir uma relação entre a área do círculo e a medida da sua circunferência. Para isso, considere um círculo de raio r. A fim de estimar sua área, podemos fatiá-lo como uma pizza, em vários pedaços iguais. Depois podemos reagrupá-los de forma a obter uma figura que se aproxima de um retângulo. Quanto maior o número de fatias, mais próximo de um retângulo fica o reagrupamento. Veja na figura abaixo.



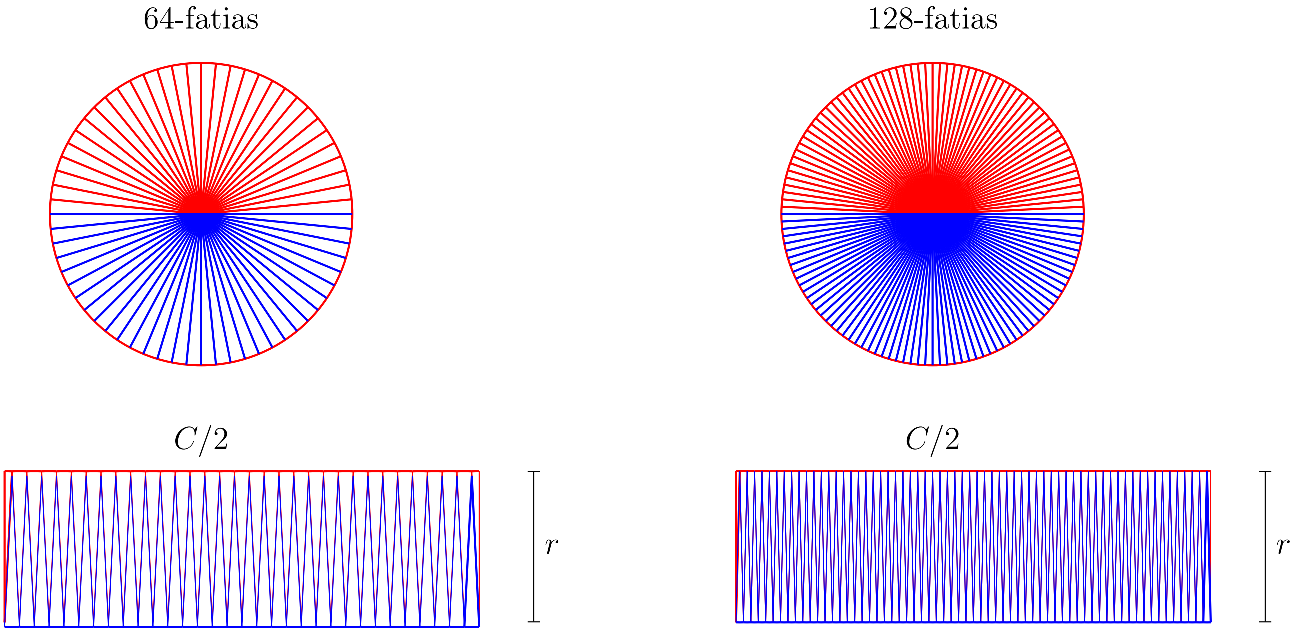


Figura 1: Fatiamento

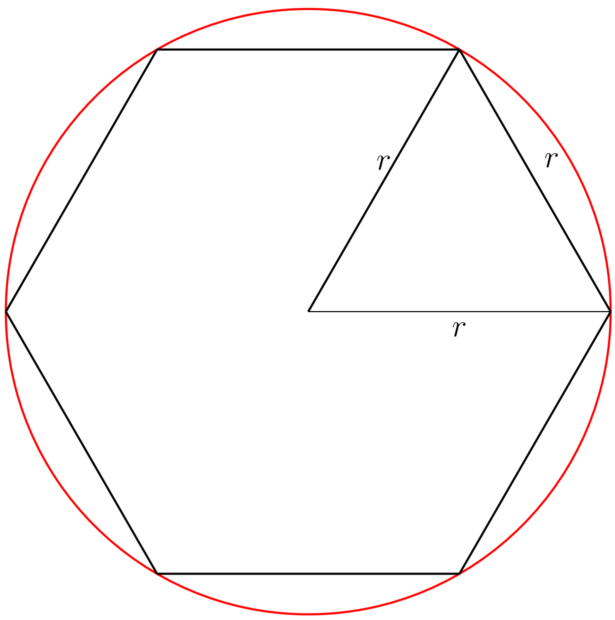
Com isso, podemos deduzir a seguinte relação entre a área e o comprimento da circunferência:

$$A = \frac{C}{2}r.$$

2 Estimando o comprimento da circunferência

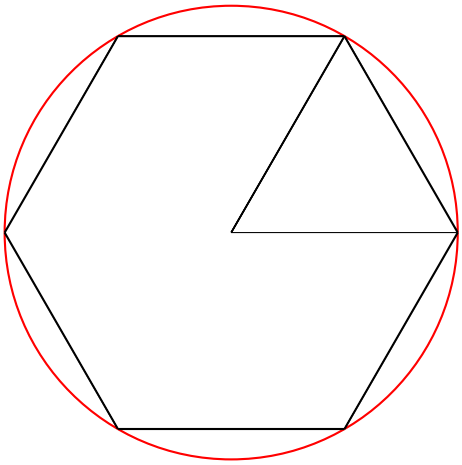
A fim de estimar o comprimento da circunferência, vamos inscrever sobre o círculo polígonos regulares. Iniciando com o hexágono, temos que $C > 6r$.

Polígono de 6 lados

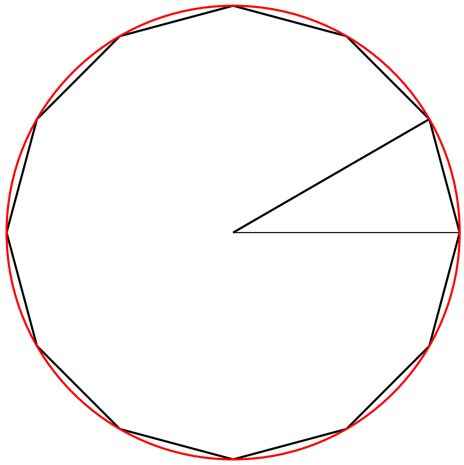


Dobrando-se os lados sucessivamente, temos polígonos com perímetro cada vez mais próximos dos valores de C.

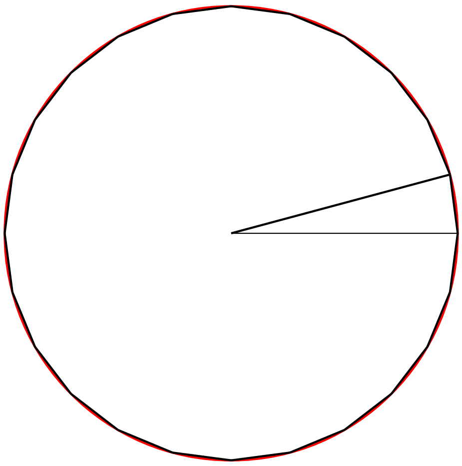
Polígono de 6 lados



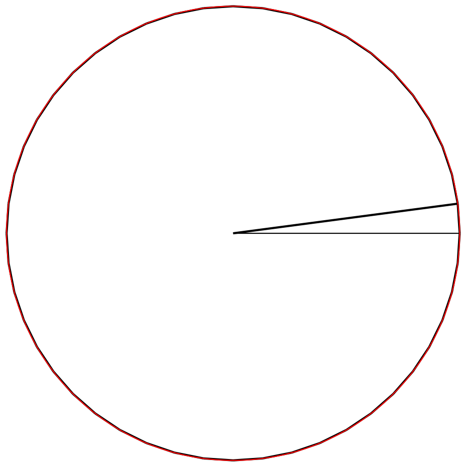
Polígono de 12 lados



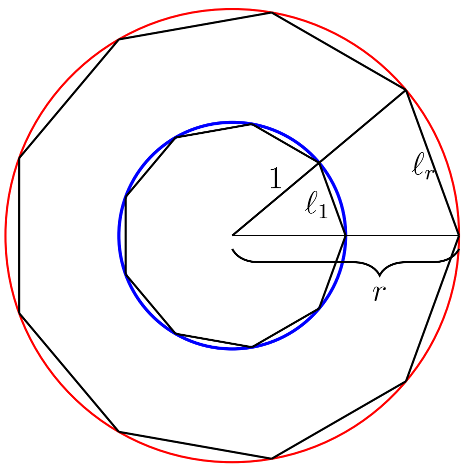
Polígono de 24 lados



Polígono de 48 lados



Podemos restringir o trabalho de fazer essas aproximações no caso do círculo de raio 1.



De fato, se C_r é a circunferência do círculo de raio r e ℓ_r é o lado do polígono de m lados inscrito neste círculo, por semelhança de triângulo, vemos que

$$\frac{\ell_r}{r} = \frac{\ell_1}{1} \Rightarrow \ell_r = \ell_1 r,$$

donde

$$C_r \approx m\ell_r = m\ell_1 r.$$

Com isso, temos que

$$A_r = \frac{C_r}{2} r \approx \frac{m\ell_1}{2} r^2.$$

Na seção seguinte veremos que a medida que m aumenta, o número $m\ell_1/2$ se aproxima de um valor, denotado pela letra grega π. Assim, podemos concluir que

$$A = \pi r^2.$$

O mesmo pode ser feito com polígonos circunscritos. Usando esse procedimento, Arquimedes estimou o comprimento da circunferência com polígonos de 96 lados. Com isso mostrou que

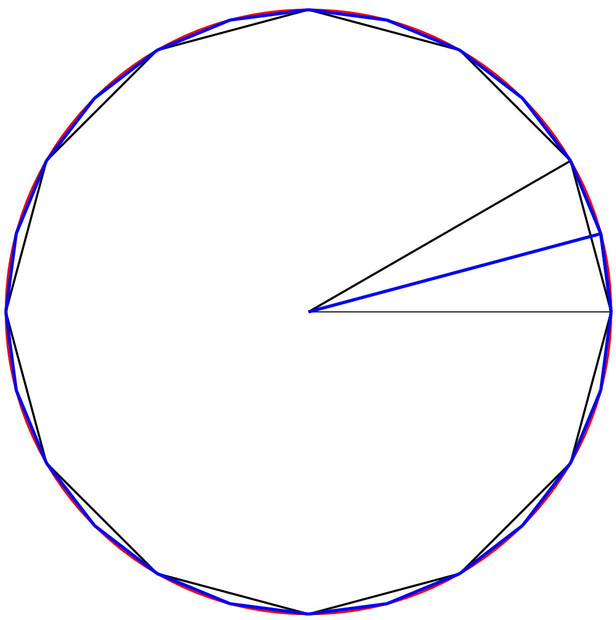
$$\frac{223}{71} < \pi < \frac{22}{7},$$

isto é, chegou a um valor aproximado de π entre 3.1408 e 3.1429.

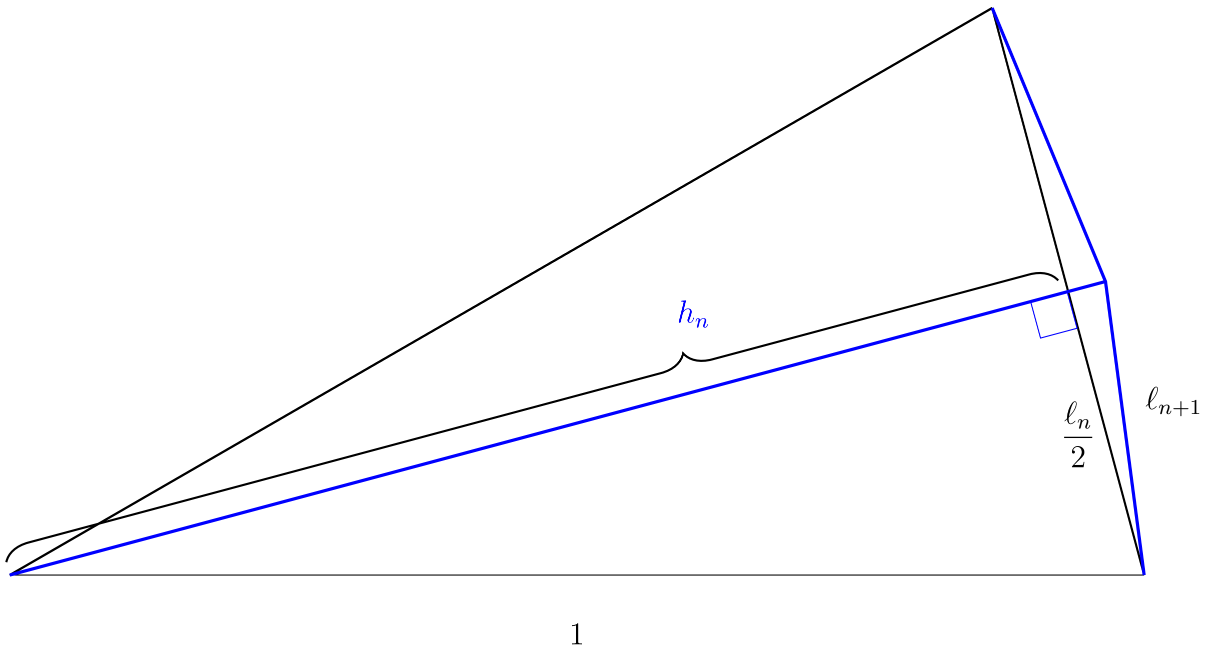
3 Estimativa da Circunferência do círculo de raio 1

Nesta seção vamos trabalhar com a circunferência do círculo de raio 1. Começamos com o hexágono inscrito e dobramos o número de lados sucessivamente a fim de obter uma aproximação cada vez melhor para o comprimento da circunferência.

Se na etapa n temos um polígono de lado ℓ_n, dividindo-se cada lado ao meio, obtemos um novo polígono inscrito de lado ℓ_{n+1} com o dobro de lados.



Vamos obter uma fórmula de recorrência que nos dá ℓ_{n+1} em função de ℓ_n. Para isso, vamos olhar o triângulo destacado acima mais de perto.



Na figura acima, podemos identificar dois triângulos retângulos. Assim, pelo Teorema de Pitágoras, no maior obtemos h_n e no menor o lado ℓ_{n+1}.

$$h_n = \sqrt{1 - \frac{\ell_n^2}{4}}, \quad \ell_{n+1} = \sqrt{\frac{\ell_n^2}{4} + (1 - h_n)^2}$$

Com isso, se M_n é o número de lados do polígono de lado ℓ_n, temos que

$$C \approx M_n \ell_n.$$

4 Executando as aproximações com Python

Agora que temos as fórmulas de recorrência, vamos usar o python para executar os cálculos. A seguir temos o script que construímos:

```
import sympy as sp

#dados iniciais
m=6 #começando com um hexágono regular
l=1 #lado do hexágono

#aproximação da circunferência usando um polígono de m lados inscritos no círculo
for n in range(4):
 h=sp.sqrt(1-l**2/4) #altura de cada triângulo dado o lado
 l=sp.sqrt(l**2/4+(1-h)**2) #lado do polígono da próxima iteração
 m=2*m  #número de lados do polígono seguinte
 p=m*l  #perímetro do polígono de m lados
 n+=1
```

n	lados	h_n	l_n	perímetro	π
1	12	0.866025403784439	0.517638090205041	6.21165708246050	3.10582854123025
2	24	0.965925826289068	0.261052384440103	6.26525722656248	3.13262861328124
3	48	0.991444861373810	0.130806258460286	6.27870040609373	3.13935020304687
4	96	0.997858923238603	0.0654381656435523	6.28206390178102	3.14103195089051

Começando com o hexágono, após 4 iterações, assim como Arquimedes, obtemos um polígono de 96 lados cujo perímetro é aproximadamente 6.28206390178102.

Com isso, para o círculo de raio 1, temos a seguinte aproximação para a área:

$$A \approx 3.14103195089051.$$

O que nos dá uma aproximação para o valor de π. Usando-se o valor de $\pi = 3.14159265358979$, armazenado no python, temos que o erro nesta aproximação é:

$$\varepsilon = 0.000560702699283322$$

Modificando um pouco nosso script, podemos escrever um novo que nos dá o número de lados do polígono inscrito de modo que tenhamos um erro abaixo de uma valor dado.

```
import sympy as sp

#dados iniciais
E=10**(-(6)) #erro estimado
m=6 #começando com um hexágono regular
l=1 #lado do hexágono
pi=3 # valor de pi inicial
erro=sp.Abs(sp.pi.evalf()-3) #valor do erro inicial
n=0

#aproximação da área usando um polígono de n lados inscritos no círculo

while erro>E:
 h=sp.sqrt(1-l**2/4) #altura de cada triângulo dado o lado
 l=sp.sqrt(l**2/4+(1-h)**2) #lado do polígono da próxima iteração
 m=2*m #número de lados do polígono
 p=m*l #perímetro do polígono de m lados
 pi=p/2 #aproximação de pi
 erro=sp.Abs(sp.pi.evalf()-pi)
 n+=1
```

n	lados	h_n	l_n	perímetro	π
1	12	0.866025403784439	0.517638090205041	6.21165708246050	3.10582854123025
2	24	0.965925826289068	0.261052384440103	6.26525722656248	3.13262861328124
3	48	0.991444861373810	0.130806258460286	6.27870040609373	3.13935020304687
4	96	0.997858923238603	0.0654381656435523	6.28206390178102	3.14103195089051
5	192	0.999464587476366	0.0327234632529736	6.28290494457092	3.14145247228546
6	384	0.999866137909562	0.0163622792078743	6.28311521582372	3.14155760791186
7	768	0.999966533917401	0.00818120805246958	6.28316778429664	3.14158389214832
8	1536	0.999991633444351	0.00409061258232819	6.28318092645610	3.14159046322805
9	3072	0.999997908358900	0.00204530736067661	6.28318421199854	3.14159210599927

Para um erro menor que 10^{-6}, precisamos de apenas 9 iterações, o que nos dá um polígono de 3072 lados circunscrito no círculo de raio 1, obtendo um valor aproximado para π de 3.14159210599927.

Na época de Arquimedes, como não haviam os recursos computacionais e nem a álgebra moderna, todos esses cálculos tinham que ser feitos à mão e usando apenas a geometria, o que era um trabalho bastante árduo e portanto impressionante!

O matemático alemão Ludolph van Ceulen, no século XV, calculou um valor de π com 35 casas decimais, começando com um quadrado e dobrando-se o número de lados sucessivamente até obter um polígono de 2^{62} lados! Entretanto, este resultado só foi publicado em 1610, após sua morte. O resultado foi o número

3.14159265358979323846264338327950288, veja [1, 2, 3]. Ele empreendeu mais de trinta anos da sua vida nesta tarefa e sua mulher mandou gravar em seu túmulo o valor de π com essas 35 casas decimais, veja figura 2.

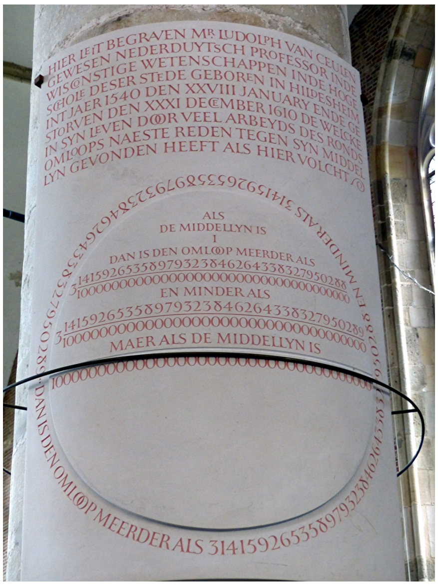


Figura 2: Lápide no túmulo do matemático Ludolph van Ceulen.

Podemos mudar nosso script para aproximar o valor de π assim como fez Ceulen.

```python
from datetime import datetime
start_time = datetime.now() # para estimar o tempo de cálculo

import sympy as sp
#dados iniciais
m=4 #começando com um quadrado inscrito
l=(sp.sqrt(2)).evalf(36) #lado do quadrado inscrito


#aproximação usando um polígono de m lados inscritos no círculo

for n in range(60):
 h=sp.sqrt(1-l**2/4) #altura de cada triângulo dado o lado
 l=sp.sqrt(l**2/4+(1-h)**2) #lado do polígono da próxima iteração
 m=2*m #número de lados do polígono
 p=m*l #perímetro do polígono de m lados
 n+=1

end_time = datetime.now()
```

Começando com um quadrado de lado $\sqrt{2} \approx 1.41421356237309504880168872420969808$ inscrito no círculo de raio 1, após 60 iterações, obtemos um polígono de 2^{62} lados cujo perímetro é aproximadamente 6.28318530717958647692528676655900577. Assim, obtemos a mesma aproximação para π obtida por Ceulen.

$$3.14159265358979323846264338327950288.$$

Contudo, diferentemente dele, que gastou mais de 30 anos, o computador demorou 0:00:00.048212 segundos nesta tarefa!

n	lados	π	Erro
1	8	3.061467458920718173827679872243190934091	0.08012519466907506463496351103631195010641
2	16	3.121445152258052285572557895632355854843	0.02014750133174095289008548764714702935412
3	32	3.136548490545939263814258044436539067556	0.005044163043853974648385338842963816640800
4	64	3.140331156954752912317118524331690132144	0.001261496635040326145524858947812752053484
5	128	3.141277250932772868062019770788214408380	0.0003154026570203704006236124912884758174896
6	256	3.141513801144301076328515059456822307935	0.00007885244549216213412832382268057626182594
7	512	3.141572940367091384135800110270761429534	0.00001971322270185432684327300874145466349443
8	1024	3.141587725277159700628854262701918739399	0.000004928312633537833789120577584144797880642
9	2048	3.141591421511199973997971763740833955748	0.000001232078593264464671619538668928449619849
10	4096	3.141592345570117742340375994157369930305	0.0000003080196754961222673891221329538919595397
11	8192	3.141592576584872665681606092237875309732	0.00000007700492057278103729104162757446511884998
12	16384	3.141592634338562989095478263627791293954	0.00000001925123024936716511965171159024313500172
13	32768	3.141592648776985669485107969277177075698	0.000000004812807568977535414002325808499394400191
14	65536	3.141592652386591345803525521057963884339	0.000000001203201892659117862221538999858518173664
15	131072	3.141592653288992765271943042173740003461	0.0000000003008004731907003411057628807366122764136
16	262144	3.141592653514593120163348243281080432652	0.00000000007520011829929513999842245154551357385307
17	524288	3.141592653570993208887718344859772114691	0.00000000001880002957492503841973076950567003339971
18	1048576	3.141592653585093231068905795335812349244	0.000000000004700007393737587943690534952755957187043
19	2097152	3.141592653588618236614208590772407884981	0.000000000001175001848434792507094999216202598747613
20	4194304	3.141592653589499488000534660432655861546	2.937504621087228468470226513583534071911e-13
21	8388608	3.141592653589719800847116201022786548981	7.343761552718225671633521587452349901280e-14
22	16777216	3.141592653589774879058761587618761014171	1.835940388179566074187002601107291222838e-14
23	33554432	3.141592653589788648611672934358282242552	4.589850970448921220641645347417473967514e-15
24	67108864	3.141592653589792090999900771048820525402	1.147462742612230682358795015362411175014e-15
25	134217728	3.141592653589792951596957730221808719599	2.868656856530576941645977480966517361356e-16
26	268435456	3.141592653589793166746221970015077869617	7.171642141326442501458066003840667964811e-17
27	536870912	3.141592653589793220533538029963396538463	1.792910535331610634573463223221780616765e-17
28	1073741824	3.141592653589793233980367044950476292008	4.482276338329026592189287067638274241820e-18
29	2147483648	3.141592653589793237342074298697246235790	1.120569084582256648407087323108499131741e-18
30	4294967296	3.141592653589793238182501112133938722073	2.801422711455641621243174450839116253734e-19
31	8589934592	3.141592653589793238392607815493111843665	7.003556778639104053252577033546626792541e-20
32	17179869184	3.141592653589793238445134491332905124064	1.750889194659776013329215470214305160158e-20
33	34359738368	3.141592653589793238458266160292853444164	4.377222986649440033403394734674005210511e-21
34	68719476736	3.141592653589793238461549077532840524189	1.094305746662360008431204742806743612743e-21
35	137438953472	3.141592653589793238462369806842837294195	2.735764366655900021881572448399282133007e-22
36	274877906944	3.141592653589793238462574989170336486697	6.839410916639750062739537034822436344023e-23
37	549755813888	3.141592653589793238462626284752211284822	1.709852729159937523720490172529840097511e-23
38	1099511627776	3.141592653589793238462639108647679984353	4.274631822899843878177847549818008913820e-24
39	2199023255552	3.141592653589793238462642314621547159236	1.068657955724961038421084005947910898498e-24
40	4398046511104	3.141592653589793238462643116115013952957	2.671644889312403399613301397292878396749e-25
41	8796093022208	3.141592653589793238462643316488380651387	6.679112223281016534639167317463207496908e-26
42	17592186044416	3.141592653589793238462643366581722325995	1.669778055820259873378301703816524377823e-26
43	35184372088832	3.141592653589793238462643379105057744646	4.174445139550707080630853004048535980522e-27
44	70368744177664	3.141592653589793238462643382235891599309	1.043611284887688249594732999913579002323e-27
45	140737488355328	3.141592653589793238462643383018600062975	2.609028212219335418357029988798397577736e-28
46	281474976710656	3.141592653589793238462643383214277178892	6.522570530552930320700474532573996821438e-29
47	562949953421312	3.141592653589793238462643383263196457871	1.630642632641676411281043303577001363183e-29
48	1125899906842624	3.141592653589793238462643383275426277616	4.076606581604191028202608258942503407958e-30
49	2251799813685248	3.141592653589793238462643383278483732552	1.019151645424006631090149867625640237482e-30
50	4503599627370496	3.141592653589793238462643383279248096286	2.547879113330427837330396640163956738779e-31
51	9007199254740992	3.141592653589793238462643383279439187219	6.369697781030182189376211311408453297686e-32
52	18014398509481984	3.141592653589793238462643383279486959953	1.592424445257545547344052827852113324421e-32
53	36028797018963968	3.141592653589793238462643383279498903136	3.981061136102737907857934959644668803676e-33
54	72057594037927936	3.141592653589793238462643383279501888932	9.952653069845585164622866299255526935409e-34
55	144115188075855872	3.141592653589793238462643383279502635381	2.488163382255766488644731024885809196962e-34
56	288230376151711744	3.141592653589793238462643383279502821993	6.220413047414224121172405565091621516803e-35
57	576460752303423488	3.141592653589793238462643383279502868646	1.555105557740959980073390392711454641399e-35
58	1152921504606846976	3.141592653589793238462643383279502880309	3.887763894352399950183475981778636603498e-36
59	2305843009213693952	3.141592653589793238462643383279502883225	9.719409735880999875458689954446591508745e-37
60	4611686018427387904	3.141592653589793238462643383279502883954	2.429967228340447457879122560539110987116e-37

Referências

[1] Cipra, B. Digits of pi. `https://www.ams.org/publicoutreach/math-history/hap-6-pi.pdf`, accessed 2022-07-28.

[2] Muller, D., and Kontorovich, A. The discovery that transformed pi. Veritasium, `https://youtu.be/gMlf1ELvRzc`,accessed 2022-07-28.

[3] O'Connor, J. J., and Robertson, E. F. Mactutor history of mathematics archive. `https://mathshistory.st-andrews.ac.uk/Biographies/Van_Ceulen/`, accessed 2022-07-28.

[4] Wikipédia. Pi. https://en.wikipedia.org/wiki/Pi, accessed 2022-07-28.