

```
<N=100
eps=1.0
for i in range (N) :
    eps = eps /2
    one_Plus_eps =1.0 +eps
    print ('eps =',eps,', one + eps =', one_Plus_eps)>
```

eps= 2.220446049250313e-16 , one + eps =
1.0000000000000002

The image shows a Python IDE with a file named 'без имени.py' open. The code in the IDE is as follows:

```

1 N=100
2 eps=1.0
3 for i in range (N) :
4     eps = eps /2
5     one_plus_eps =1.0 +eps
6     print ('eps =',eps,', one + eps =', one_plus_eps)
7
8
9

```

Below the IDE, a Windows command prompt window titled 'C:\Windows\system32\cmd.exe' displays the output of the script. The output shows the value of 'eps' decreasing exponentially from $5.820766091346741 \times 10^{-11}$ to $3.469446951953614 \times 10^{-18}$, while 'one + eps' remains at 1.0 for most iterations, except for the final ones where it shows a slight increase due to floating-point precision.

```

eps = 5.820766091346741e-11 , one + eps = 1.0000000000582077
eps = 2.9103830456733704e-11 , one + eps = 1.0000000000291038
eps = 1.4551915228366852e-11 , one + eps = 1.000000000014552
eps = 7.275957614183426e-12 , one + eps = 1.000000000007276
eps = 3.637978807091713e-12 , one + eps = 1.000000000003638
eps = 1.8189894035458565e-12 , one + eps = 1.000000000001819
eps = 9.094947017729282e-13 , one + eps = 1.0000000000009095
eps = 4.547473508864641e-13 , one + eps = 1.0000000000004547
eps = 2.2737367544323206e-13 , one + eps = 1.0000000000002274
eps = 1.1368683772161603e-13 , one + eps = 1.0000000000001137
eps = 5.684341886080802e-14 , one + eps = 1.0000000000000568
eps = 2.842170943040401e-14 , one + eps = 1.0000000000000284
eps = 1.4210854715202004e-14 , one + eps = 1.0000000000000142
eps = 7.105427357601002e-15 , one + eps = 1.000000000000007
eps = 3.552713678800501e-15 , one + eps = 1.0000000000000036
eps = 1.7763568394002505e-15 , one + eps = 1.0000000000000018
eps = 8.881784197001252e-16 , one + eps = 1.0000000000000009
eps = 4.440892098500626e-16 , one + eps = 1.0000000000000004
eps = 2.220446049250313e-16 , one + eps = 1.0000000000000002
eps = 1.1102230246251565e-16 , one + eps = 1.0
eps = 5.551115123125783e-17 , one + eps = 1.0
eps = 2.7755575615628914e-17 , one + eps = 1.0
eps = 1.3877787807814457e-17 , one + eps = 1.0
eps = 6.938893903907228e-18 , one + eps = 1.0
eps = 3.469446951953614e-18 , one + eps = 1.0

```

№5

```

<N=100
eps=complex(1, 1)
for i in range (N) :
    eps = eps /2
    one_plus_eps =complex( 1, 1)+eps
    print ('eps =',eps,', one + eps =', one_plus_eps)
>

```

$\text{eps} = (2.220446049250313\text{e-}16 + 2.220446049250313\text{e-}16j)$, $\text{one} + \text{eps} = (1.0000000000000002 + 1.0000000000000002j)$

```

er.py  без имени.py  rt.py
1  N=100
2  eps=complex(1, 1)
3  for i in range(N):
4      eps = eps / 2
5      one_plus_eps = complex(1, 1) + eps
6      print('eps =', eps, ', one + eps =', one_plus_eps)
7
8
9

```

```

C:\Windows\system32\cmd.exe
000284+1.0000000000000000284j)
eps = <1.4210854715202004e-14+1.4210854715202004e-14j> , one + eps = <1.000000000
00000142+1.000000000000000142j>
eps = <7.105427357601002e-15+7.105427357601002e-15j> , one + eps = <1.00000000000
00007+1.0000000000000007j>
eps = <3.552713678800501e-15+3.552713678800501e-15j> , one + eps = <1.00000000000
00036+1.00000000000000036j>
eps = <1.7763568394002505e-15+1.7763568394002505e-15j> , one + eps = <1.000000000
000018+1.00000000000000018j>
eps = <8.881784197001252e-16+8.881784197001252e-16j> , one + eps = <1.00000000000
00009+1.0000000000000009j>
eps = <4.440892098500626e-16+4.440892098500626e-16j> , one + eps = <1.00000000000
00004+1.0000000000000004j>
eps = <2.220446049250313e-16+2.220446049250313e-16j> , one + eps = <1.00000000000
00002+1.0000000000000002j>
eps = <1.1102230246251565e-16+1.1102230246251565e-16j> , one + eps = <1+1j>
eps = <5.551115123125783e-17+5.551115123125783e-17j> , one + eps = <1+1j>
eps = <2.775575615628914e-17+2.775575615628914e-17j> , one + eps = <1+1j>
eps = <1.3877787807814457e-17+1.3877787807814457e-17j> , one + eps = <1+1j>
eps = <6.938893903907228e-18+6.938893903907228e-18j> , one + eps = <1+1j>
eps = <3.469446951953614e-18+3.469446951953614e-18j> , one + eps = <1+1j>
eps = <1.734723475976807e-18+1.734723475976807e-18j> , one + eps = <1+1j>
eps = <8.673617279884035e-19+8.673617279884035e-19j> , one + eps = <1+1j>
eps = <4.336808689942018e-19+4.336808689942018e-19j> , one + eps = <1+1j>
eps = <2.168404344971009e-19+2.168404344971009e-19j> , one + eps = <1+1j>

```

№6

```

<import math
x=math.pi/2
eps=10**(-5)
s=x
t=x
i=2
while abs(t/s)>eps:
    t=(-x**2/((2*i-1)*(2*i-2)))*t
    s=s+t
    i=i+1
import math
a=math.fabs(s-math.sin(x))/math.sin(x)
print ('s=',s, 'tmax=',t, 'a=',a, 'i=', i)
>

```

dz1.py
dz2.py
hw.py

```

1 import math
2 x=math.pi/2
3 eps=10**(-100)
4 s=x
5 t=x
6 i=2
7 while abs(t/s)>eps:
8     t=(-x**2/((2*i-1)*(2*i-2)))*t
9     s=s+t
10    i=i+1
11 import math
12 a=math.fabs(s-math.sin(x))/math.sin(x)
13 print ('s=',s, 'tmax=',t, 'a=',a, 'i=', i)
14

```

C:\windows\SYSTEM32\cmd.exe

```

s= 1.0000000000000002 tmax= -3.482030331464222e-102 a= 2.220446049250313e-16 i= 41

-----
(program exited with code: 0)
Для продолжения нажмите любую клавишу . . .

```

sum	t max	ошибка	степень eps
0.9999999999939768	-6.688035109811466e-106.023181953196399e-12	6.023181953196399e-12	-8
1.00000000000000437	6.066935731106194e-12	4.374278717023117e-14	-10
1.00000000000000002	3.4140529700331598e-31	2.220446049250313e-16	-30
1.00000000000000002	6.382145039734999e-51	2.220446049250313e-16	-50

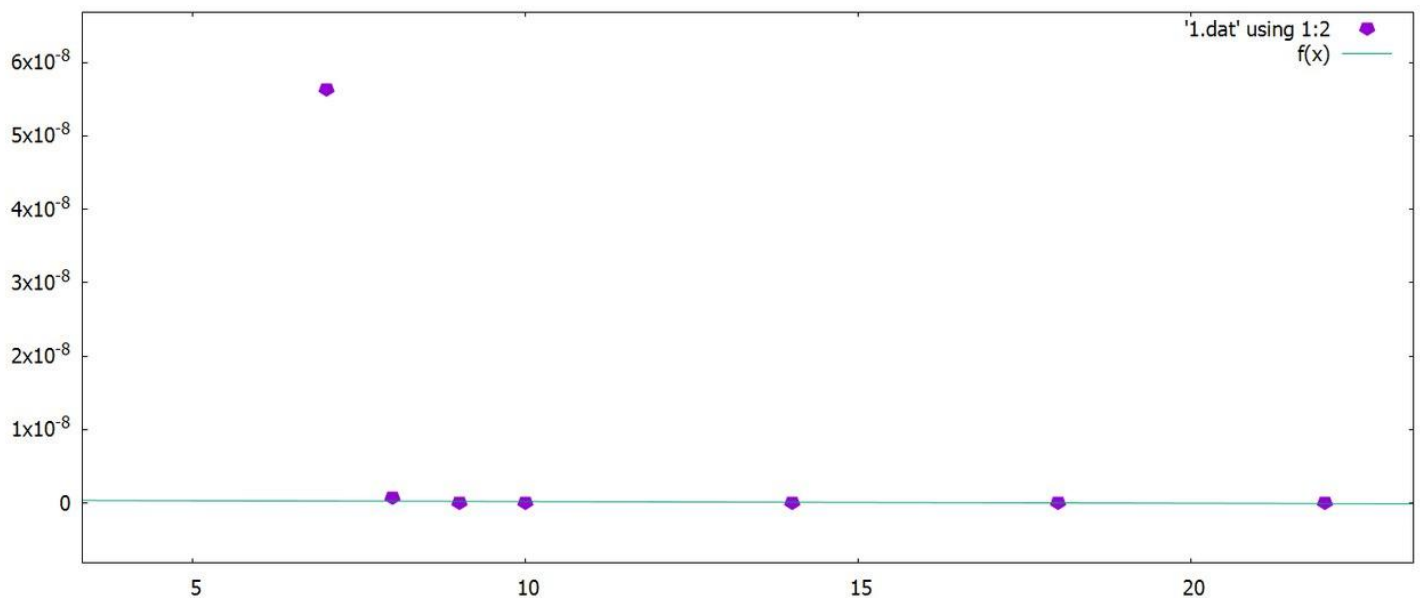
- Из данной таблицы мы видим, что при достаточно малых значениях x алгоритм сходится к 1, Значение тем ближе к 0, чем меньше точность.
- После того, как наша точность превысила машинную, значение суммы ряда остается постоянным(при $x=\pi/2 =1.00000000000000002$)
- Без использования формулы приведения ($\sin(x+2n\pi)=\sin(x)$) существует диапазон x, для которых алгоритм сходится, но не к 0, а к 1. Что видно из данной таблицы.

sum	делитель pi (pi/k)
0.9980267284282714	50
0.9995065603657319	100
0.9999950652018581	1000

- При применении формул приведения алгоритм также сходится при малым x .
- "Плохая" версия алгоритма

```
<import math
x=math.pi*2
eps=10**(-50)
s=x
t=x
i=2
while abs(t/s)>eps:
    import math
    t=(-1)**(i-1)*(x**(2*i-1))/math.factorial((2*i-1))*t
    s=s+t
    i=i+1
import math
a=math.fabs(s-math.sin(x))/math.sin(x)
print ('s=',s, 'tmax=',t, 'a=',a, 'i=', i)
>
```

"Хороший"	"Плохой"
0.019757158783946512	0.019758418960677735
0.15643446504023087	0.1569781647954669
4.3878919606925197e-16	-1782193851.3401673



- Из данного графика мы видим, что при больших точностях функция ошибок от количества слагаемых выходит на константу

№7

```
<import math
import sys

a= float(input("a = "))
```

```

if a == 0:
    print ('не является квадратным')
    sys.exit()

b = float(input("b = "))
c = float(input("c = "))

D = b**2- 4*a*c
if D == 0:
    m=-b/2*a
    print ('x =', m)
    sys.exit()
if D > 0:
    n=(-b + math.sqrt(D))/ 2*a
    m=(-b - math.sqrt(D))/ 2*a
    print ('x1 =',n,'x2 =',m)
else:
    D<0
    D=complex(D**0.5)
    n=(-b+complex((D**0.5)))/2*a
    m=(-b-complex((D**0.5)))/2*a
    print ('x1 =',n,'x2 =',m)
>

```

dz1.py dz2.py hw.py kvadr.py kvad2.py

```

3
4 a= float(input("a = "))
5 if a == 0:
6     print ('не является квадратным')
7     sys.exit()
8
9 b = float(input("b = "))
10 c = float(input("c = "))
11
12 D = b**2- 4*a*c
13 if D == 0:
14     m=-b/2*a
15     print ('x =', m)
16     sys.exit()
17 if D > 0:
18     n=(-b + math.sqrt(D))/ 2*a
19     m=(-b - math.sqrt(D))/ 2*a
20     print ('x1 =',n,'x2 =',m)
21 else:
22     D<0
23     D=complex(D**0.5)
24     n=(-b+complex((D**0.5)))/2*a
25     m=(-b-complex((D**0.5)))/2*a
26     print ('x1 =',n,'x2 =',m)
27

```

C:\windows\SYSTEM32\cmd.exe

```

a = 3
b = 4
c = 5
x1 = (-3.2682595697431918+2.731740430256808j) x2 = (-8.731740430256806-2.731740430256808j)

-----
(program exited with code: 0)

Для продолжения нажмите любую клавишу . . .

```