

Calcul 1

Commande :

```
n:0
```

Résultat :

```
0
```

Calcul 2

Commande :

```
s:0
```

Résultat :

```
0
```

Calcul 3

Commande :

```
while n < 10 do (n : n + 1, s : s + n)
```

Calcul 4

Commande :

```
s
```

Résultat :

```
55
```

Calcul 5

Commande :

```
s:0
```

Résultat :

```
0
```

Calcul 6

Commande :

```
for n from 0 thru 10 do (s : s + n)
```

Calcul 7

Commande :

```
s
```

Résultat :

```
55
```

Calcul 8

Commande :

```
s:0
```

Résultat :

```
0
```

Calcul 9

Commande :

```
for n from 0 thru 10 do s : s + n
```

Calcul 10

Commande :

```
s
```

Résultat :

```
55
```

Calcul 11

Commande :

```
s:0
```

Résultat :

```
0
```

Calcul 12

Commande :

```
for n step 2 from 0 thru 10 do s : s + n
```

Calcul 13

Commande :

```
s
```

Résultat :

```
30
```

Calcul 14

Commande :

```
f(x):=if x < 0 then 0 else 1
```

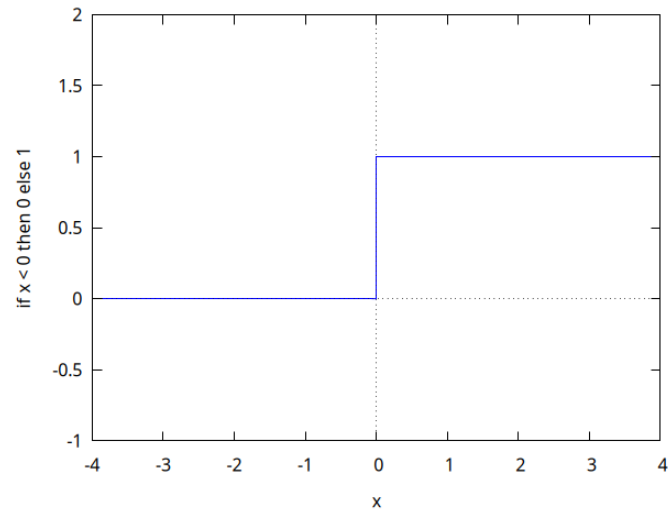
Résultat :

```
f(x):=if x < 0 then 0 else 1
```

Calcul 15

Commande :

```
wxplot2d(f(x),[x, -4,4], [y, -1, 2])
```



Calcul 16

Commande :

```
f(x):=if x < 0 then 0
```

Résultat :

```
f(x):=if x < 0 then 0
```

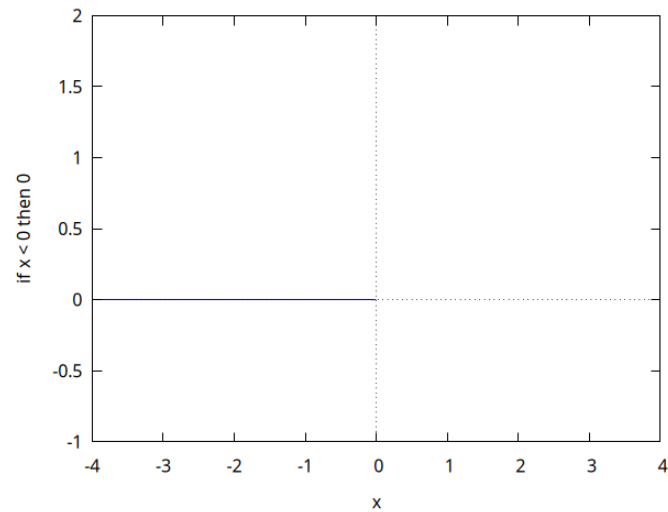
Calcul 17

Commande :

```
wxplot2d(f(x),[x, -4,4], [y, -1, 2])
```

Affichage :

plot2d: expression evaluates to non-numeric value somewhere in plotting range.



Calcul 18

Commande :

```
f(x):= if x < 1 then 1 else x*f(x-1)
```

Résultat :

```
f(x):=if x < 1 then 1 else x*f(x-1)
```

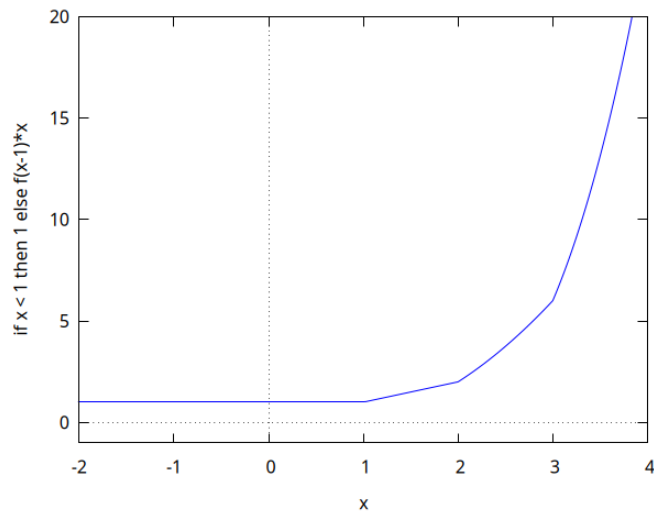
Calcul 19

Commande :

```
wxplot2d(f(x),[x, -2,4], [y, -1, 20])
```

Affichage :

plot2d: some values will be clipped.



Calcul 20

Commande :

```
true and flase
```

Résultat :

flase

Calcul 21

Commande :

```
true or flase
```

Résultat :

true

Calcul 22

Commande :

```
(true and flase) or true
```

Résultat :

```
true
```

Calcul 23

Commande :

```
x:true
```

Résultat :

```
true
```

Calcul 24

Commande :

```
y:false
```

Résultat :

```
false
```

Calcul 25

Commande :

```
x and y
```

Résultat :

```
false
```

Calcul 26

Commande :

```
x or y
```

Résultat :

```
true
```

Calcul 27

Commande :

```
kill(x, y)
```

Résultat :

```
done
```

Calcul 28

Commande :

```
g(x) := block([temp], temp:1, while x > 1 do(temp:x*temp, x:x-1), return(temp))
```

Résultat :

```
g(x):=block([temp],temp:1,while x > 1 do (temp:x*temp,x:x-1),return(temp))
```

Calcul 29

Commande :

```
g(5)
```

Résultat :

```
120
```

Calcul 30

Commande :

```
g(x) := block([temp], temp:1, while x > 1 do(temp:x*temp, x:x-1), temp)
```

Résultat :

```
g(x):=block([temp],temp:1,while x > 1 do (temp:x*temp,x:x-1),temp)
```

Calcul 31

Commande :

```
g(5)
```

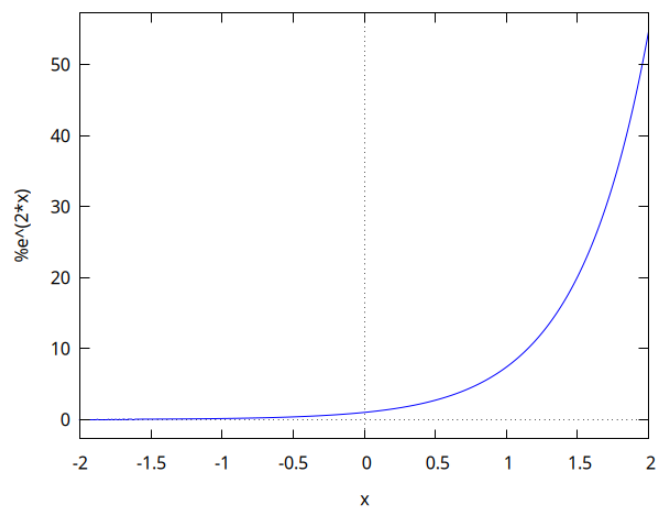
Résultat :

```
120
```

Calcul 32

Commande :

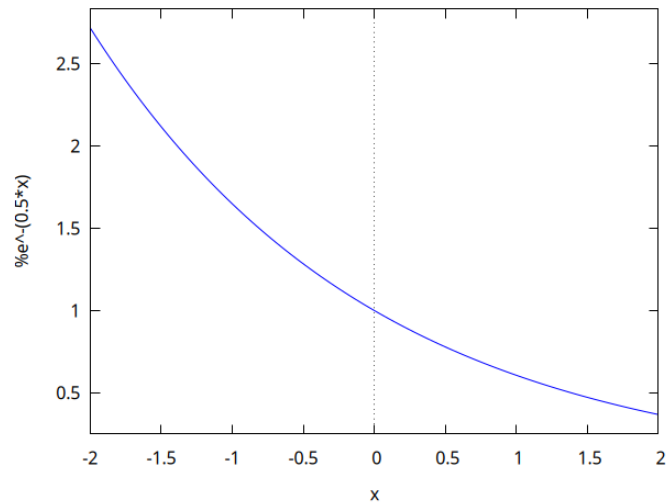
```
wxplot2d(exp(2*x), [x, -2, 2])
```



Calcul 33

Commande :

```
wxplot2d(exp(-0.5*x), [x, -2, 2])
```



Calcul 34

Commande :

```
test(x,y):=block([], if x > y then print(x, "est plus grand que ", y) else print(x, "nest pas plus grand que",  
y), return("Commande excute"))
```

Résultat :

```
test(x,y):=block([],  
if x > y then print(x,"est plus grand que ",y)  
else print(x,"nest pas plus grand que",y),  
return("Commande excute "))
```

Calcul 35

Commande :

```
test(4,3)
```

Affichage :

```
4 est plus grand que 3
```

Résultat :

```
Commande exécutée
```

Calcul 36

Commande :

```
x:4
```

Résultat :

```
4
```

Calcul 37

Commande :

```
display(x)
```

Affichage :

```
x = 4
```

Calcul 38

Commande :

```
kill(y)
```

Résultat :

```
done
```

Calcul 39

Commande :

```
assume(y > 0)
```

Résultat :

```
[y > 0]
```

Calcul 40

Commande :

```
test(y^2+1,-y)
```

Affichage :

```
y^2+1 est plus grand que -y
```

Résultat :

```
Commande exécutée
```

Calcul 41

Commande :

```
forget(y>0)
```

Résultat :

```
[y > 0]
```

Calcul 42

Commande :

```
test(y^2+1,-y)
```

Résultat :

```
Commande exécutée
```

Calcul 43

Commande :

```
assume(y < 0)
```

Résultat :

$[y < 0]$

Calcul 44

Commande :

```
test(y^2+1,-y)
```

Résultat :

Commande exécutée

Calcul 45

Commande :

```
facts()
```

Résultat :

$[0 > y]$

Calcul 46

Commande :

```
forget(all)
```

Résultat :

$[true]$

Calcul 47

Commande :

```
facts()
```

Résultat :

$[0 > y]$

Calcul 48

Commande :

```
declare(n, integer)
```

Résultat :

done

Calcul 49

Commande :

```
assume(n>0)
```

Résultat :

$[redundant]$

Calcul 50

Commande :

```
cos(n*%pi)
```

Résultat :

1

Calcul 51

Commande :

```
kill(all)
```

Résultat :

done

Calcul 52

Commande :

```
cos(n*%pi)
```

Résultat :

$\cos(\pi n)$

Calcul 53

Commande :

```
1+2
```

Résultat :

3