

- La fonction `\FPseed=\x` permet de modifier le système de génération de nombre aléatoire : `\x` peut prendre `\time`, `\the\day`, `\the\month`, `\the\year`
`\FPeval\compteur}{clip(\the\month*\the\time*1.1)}`
`\FPseed=\time`

- La fonction `\FPeval{x}{...}` permet de faire des calculs et d'affecter le résultat à la variable `\x`.

- Pour intégrer la variable `\x` dans un autre calcul, il faut la faire suivre d'une paire d'accollades ou de `\space`.
`\FPeval\x{5+2} \FPeval\y{\x}*3} \FPeval\z{\y\space /7}`
7.000000000000000000 21.000000000000000000 3.000000000000000000

- La fonction `\FPrandom{x}` calcule un nombre aléatoire. Pour l'afficher `\FPprint{x}`
`\x= 0.218418296993904885`

- La fonction `\FPtrunc{y}{x}{a}` transfère le nombre `\x` avec `a` chiffres après la virgule à la variable `\y`, on l'affiche avec `\numprint{y}`
`a=0⇒\y=0 a=1⇒\y=0,2 a=5⇒\y=0,218 41 a=8⇒\y=0,218 418 29`

- On peut cumuler les fonctions `\FPeval` et `\FPtrunc`.
Exemple `\FPeval{x}{trunc(a+random*b,c)}`
`\x` sera compris entre `a` et `a+b`
, séparateur
`c` : nombre de chiffre après la virgule

- `\FPeval{y}{trunc(1+random*10,1)}\numprint{y}`
`\FPeval{w}{trunc(21+random*6,0)}\numprint{w}`
`\FPeval{z}{trunc(1+random*8,9)}\numprint{z}`
10,5
25
5,493 563 542

- La fonction `\FPround{y}{x}{a}` transfère le nombre `\x` avec `a` chiffres après la virgule à la variable `\y` et arrondi, on l'affiche avec `\numprint{y}`
`a=0⇒\y=0 a=1⇒\y=0,2 a=5⇒\y=0,218 42 a=8⇒\y=0,218 418 30`

- La fonction `\FPclip{y}{x}` transfère le nombre `\x`, sans les zéros inutiles, à la variable `\y`.
Exemple : `\FPeval{y}{clip(10.000)}` affiche 10

- Les fonctions `\FPadd{x}{a}{b}`, `\FPdiv{x}{a}{b}`, `\FPmul{x}{a}{b}`, `\FPsub{x}{a}{b}` attribue à la variable `\x` la somme de `a+b`, le quotient de `a/b`, le produit de `a*b`, et le reste de `a-b`

- La fonction `FPset{x}{y}` attribue à la variable `\x` (macro ou chaîne) la valeur `\y`

- `\FPabs{x}{a}` renvoie la valeur absolue de `a`
`\FPneg{y}{a}` renvoie l'opposé de `a`

— La commande `\FPiflt{\x}{\y} {instruction 1} \else {instruction 2} \fi` vérifie si $x < y$ dans ce cas l'instruction 1 est lancée sinon c'est l'instruction 2.

La commande `\FPifeq{\x}{\y} {instruction 1} \else {instruction 2} \fi` vérifie si $x = y$ dans ce cas l'instruction 1 est lancée sinon c'est l'instruction 2.

La commande `\FPifgt{\x}{\y} {instruction 1} \else {instruction 2} \fi` vérifie si $x > y$ dans ce cas l'instruction 1 est lancée sinon c'est l'instruction 2.

La commande `\FPifneg{\x} {instruction 1} \else {instruction 2} \fi` vérifie si $x < 0$ dans ce cas l'instruction 1 est lancée sinon c'est l'instruction 2.

La commande `\FPifpos{\x} {instruction 1} \else {instruction 2} \fi` vérifie si $x > 0$ dans ce cas l'instruction 1 est lancée sinon c'est l'instruction 2.

La commande `\FPifzero{\x} {instruction 1} \else {instruction 2} \fi` vérifie si $x = 0$ dans ce cas l'instruction 1 est lancée sinon c'est l'instruction 2.

La commande `\FPifint{\x} {instruction 1} \else {instruction 2} \fi` vérifie si x est entier dans ce cas l'instruction 1 est lancée sinon c'est l'instruction 2.

La commande `\ifFPtest {instruction 1} \else {instruction 2} \fi` répète le dernier test.

— Les commandes `\FPmin{\x}{\y}{\z}`, `\FPmax{\x}{\y}{\z}` attribue à la valeur x le minimum, maximum entre y et z .

— La commande `\FPlsolve{\x}{a}{b}` cherche la valeur réelle x pour résoudre l'équation $a*x+b=0$.

La commande `\FPqsolve{\x}{\y}{a}{b}{c}` cherche les valeurs réelles x et y pour résoudre l'équation $a*x^2+b*x+c=0$

La commande `\FPcsolve{\x}{\y}{\z}{a}{b}{c}{d}` cherche les valeurs réelles x , y et z pour résoudre l'équation $a*x^3+b*x^2+c*x+d=0$

La commande `\FPqqsolve{\w}{\x}{\y}{\z}{a}{b}{c}{d}{e}` cherche les valeurs réelles w , x , y et z pour résoudre l'équation $a*x^4+b*x^3+c*x^2+d*x+e=0$

— La commande `\FPe` affiche la valeur de $e=2.718281828459045235$

— La commande `\FPpi` affiche la valeur de π .

— La commande `\FPexp{\x}{a}` attribue à la variable x la valeur e^a

- La commande `\FPln{\x}{a}` attribue à la variable `\x` la valeur $\ln(a)$
- La commande `\FPpow{\x}{a}{b}` attribue à la variable `\x` la valeur a^b
- La commande `\FProot{\x}{a}{b}` attribue à la variable `\x` la valeur $a^{(1/b)}$.
On peut écrire `\FPeval{\x}{root(b,a)}`
- La commande `\FPpascal{\x}{a}` attribue à `\x` la ligne a du triangle de pascal.
- La commande `\FPSin{\x}{a}` attribue à `\x` la valeur du sinus de a qui est exprimé en radians.
`\FPeval{\x}{sin(30*\FPpi/180)}\FPprint{\x}`
- La commande `\FPCos{\x}{a}` attribue à `\x` la valeur du cosinus de a qui est exprimé en radians.
- La commande `\FPSincos{\x}{\y}{a}` attribue à `\x` la valeur du sinus de a qui est exprimé en radians et `\y` la valeur du cosinus de a qui est exprimé en radians.
- La commande `\FPTan{\x}{a}` attribue à `\x` la valeur de la tangente de a qui est exprimé en radians.
- La commande `\FPCot{\x}{a}` attribue à `\x` la valeur de la cotangente de a qui est exprimé en radians.
- La commande `\FPTancot{\x}{\y}{a}` attribue à `\x` la valeur de la tangente de a qui est exprimé en radians et `\y` la valeur de la cotangente de a .
- La commande `\FParcsin{\x}{a}` attribue à `\x` la valeur de l'arc sinus(a), `\x` sera exprimé en radians.

La commande `\FParccos{\x}{a}` attribue à `\x` la valeur de l'arc cosinus(a), `\x` sera exprimé en radians.

La commande `\FParcsincos{\x}{\y}{a}` attribue à `\x` la valeur de l'arc sinus(a), `\x` sera exprimé en radians, à `\y` la valeur de l'arc cosinus(a), `\y` sera exprimé en radians.

La commande `\FParctan{\x}{a}` attribue à `\x` la valeur de l'arc tangente(a), `\x` sera exprimé en radians.

La commande `\FParccot{\x}{a}` attribue à `\x` la valeur de l'arc cotangente(a), `\x` sera exprimé en radians.

La commande `\FParctancot{\x}{\y}{a}` attribue à `\x` la valeur de l'arc tangente(a), `\x` sera exprimé en radians, à `\y` la valeur de l'arc cotangente(a), `\y` sera exprimé en radians.

- `\FPupn` : le principe est de prendre chaque opérande puis d'appliquer l'opérateur sur les deux derniers opérandes dès qu'un symbole mathématique apparaît. Il s'agit du même principe que la notation polonaise inversée (NPI ou RPN) utilisée avec `pstricks` mais avec une différence notable les opérations se font avec l'opérande avant l'opérateur puis le précédent donc de droite à gauche (et non pas de gauche à droite comme avec la NPI).

Dans les trois exemples suivants, le calcul est le même : $((3 + 2) - 4) \times 5 = 5$

`\FPupn{x{5 4 3 2 4 + - *} = 5}`

5	5	5	5	5	5	5
4	4	4	4	1	1	
3	3	5	5	×	×	
2	2	-	-			
+	+	×	×			
-	-					
×	×					

neg : donne l'opposé

`\FPupn{x{3 2 + 4 - 5 * neg clip} = 5}`

3	3	5	5	-1	-1	-5	-5	5
2	2	4	4	5	5	neg	neg	
+	+	-	-	×	×			
4	4	5	5	neg	neg			
-	-	×	×					
5	5	neg	neg					
×	×							
neg	neg							

swap : permute les éléments

`\FPupn{x{3 2 + 4 - swap 5 * clip} = 5}`

3	3	5	5	4	4	1	1	5
2	2	4	4	5	5	5	5	
+	+	swap	swap	-	-	×	×	
4	4	-	-	5	5			
swap	swap	5	5	×	×			
-	-	×	×					
5	5							
×	×							

copy : duplique le résultat

`\FPupn{x{3 copy * clip}}` donne 9 équivalent à `\FPupn{x{3 3 * clip}}`

pop : efface l'élément précédent

`\FPupn{x{3 2 + 3 4 pop swap - 5 * clip}}` équivaut à

`\FPupn{x{3 2 + 3 swap - 5 * clip}}` équivaut à

`\FPupn{x{3 3 2 + - 5 * clip}}` équivaut à

`\FPupn{x{3 copy 2 + - 5 * clip}}` équivaut à

`\FPupn{x{5 3 copy 2 + - * clip}}`