

Découverte de la programmation

Avertissement

Ce document a été réalisé pour montrer quelques-unes des multiples fonctionnalités de la calculatrice TI-Nspire (CAS ou non CAS) et pour aider les débutants à prendre en main rapidement cette calculatrice.

Au cours des années, à la demande des utilisateurs, la calculatrice a bénéficié de nombreuses améliorations (nouvelles fonctionnalités, écran couleur, etc.). C'est particulièrement le cas en programmation où de nouvelles commandes sont apparues.

En conséquence, comme le document que vous avez sous les yeux a été réalisé avec la version 1.6 de l'O.S. en 2008, il peut y avoir des fonctionnalités nouvelles non abordées et quelques différences minimes dans les commandes ou dans les écrans.

Par exemple, dans l'application **Géométrie**, la commande : **menu** 7 : Points et droites, 2 : Point sur est remplacée, dans la version 3.2, par : **menu** 8 : Géométrie, 1 : Points et droites, 2 : Point sur.

Pour prendre en main la machine, nous vous conseillons de commencer par consulter le document « Outils de base », réalisé en 2012 sur la version 3.2, qui permet d'avoir une vue d'ensemble et de se familiariser avec les commandes de base (sauvegarder un fichier, etc.).

Vous disposez, pour vous aider à cette initiation à TI-Nspire, des documents suivants :

Titre	Application	Version TI-Nspire	Année
Outils de base	Multi-applications	3.2	2012
Fonctions numériques	Graphiques , Calculs, Éditeur mathématique	2.1	2010
AireRayon	Géométrie , Tableur & listes, Graphiques	2.1	2010
LeTriangle	Géométrie , Calculs	2.1	2010
TI-Nspire et la simulation	Tableur & listes , Calculs, Graphiques	1.6	2008
Données & statistique	Données & statistiques , Tableur & listes	1.6	2008
Programmation	Calculs , Tableur & listes	1.6	2008
Utilisation des bibliothèques	Calculs	1.6	2008
PublishView	PublishView	3.0	2011

De plus, deux fichiers sont destinés aux utilisateurs de TI-Navigator (version 3.0, de 2011).

Pour l'équipe T3 France, novembre 2012.

Découverte de la programmation sur la calculatrice

TI-*n*spire™/ TI-*n*spire™CAS

Ce document a été réalisé avec la version 1.6 de la calculatrice TI-Nspire™.

Introduction

La plate forme scientifique TI-*n*spire™ offre dans l'application **Calculs** la possibilité de créer des *programmes* et des *fonctions* pouvant éventuellement être réutilisés dans d'autres problèmes.

Un programme permet d'effectuer une suite d'opérations de façon automatique. Les données d'entrée sont des paramètres d'appels et sont entrés entre parenthèses lors de l'appel du programme. Le programme comporte également des instructions de traitement de ces données et enfin des instructions d'affichage des résultats obtenus.

Une fonction effectue comme le programme, des instructions, mais retourne un résultat destiné à une utilisation ultérieure. Généralement, la fonction retourne un argument unique dépendant des arguments utilisés, mais peut aussi en retourner plusieurs à condition de les placer dans une liste.

Premier exemple : PGCD

On se propose de calculer le pgcd de deux nombres par l'algorithme d'Euclide.

- 1 Diviser a par b , on obtient le reste r
- 2 Si $r = 0$, le pgcd est trouvé : $\text{pgcd}(a,b) = b$
- 3 Si $r \neq 0$, remplacer a par b , b par r et recommencer à partir de 1

Créer un nouveau classeur et choisir l'application **Calculs** :

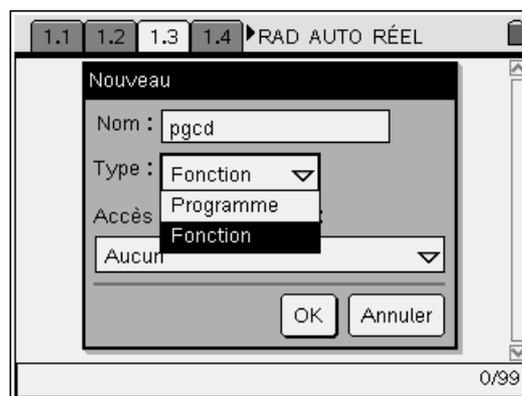
 puis choisir 6 : **Nouveau classeur**,
puis 1 : **Calculs**.

Appuyer sur la touche .

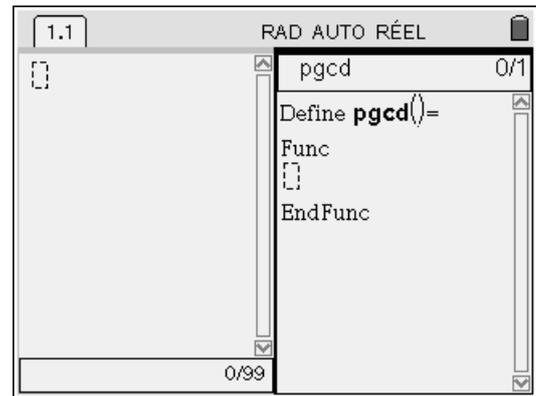
Choisir 8 : **Fonctions et Programmes** puis 1 : **Editeur de programmes** et enfin 1 : **Nouveau**.

Remplir les champs proposés dans la fenêtre qui s'ouvre sans oublier de préciser l'accès à la bibliothèque (**LibPub** permet d'utiliser le programme dans d'autres classeurs en le rendant visible du catalogue, **LibPriv** ne le rend pas visible du catalogue. **Aucun** ne permet d'utiliser le programme que dans le classeur courant).

Utiliser ensuite la touche  puis valider par **OK**.



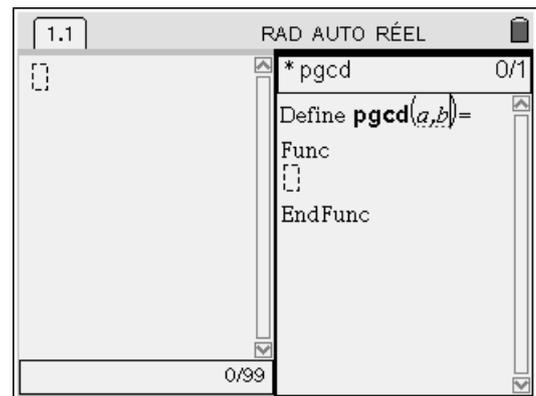
L'écran se partage automatiquement en deux parties verticales, la fenêtre de droite permet d'éditer le programme ou la fonction, celle de gauche de le tester.



Entrée des instructions du programme ou de la fonction

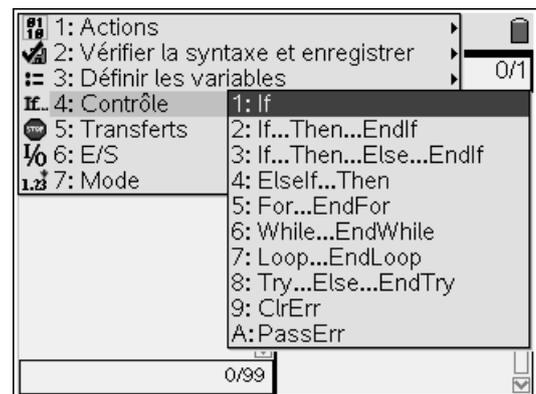
Les nombres *a* et *b* sont à mettre en arguments de la fonction, utiliser le pavé de navigation pour placer le curseur dans la parenthèse et entrer les arguments *a* et *b* séparés par une virgule.

Placer le curseur entre les lignes **Func** et **End Func** pour éditer le corps du programme.

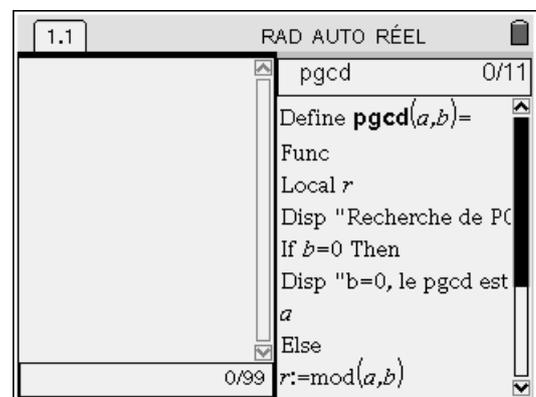


Celui-ci peut être totalement rédigé sous forme de texte dès lors que la syntaxe des instructions est connue. Mais celles-ci s'obtiennent aussi en utilisant les raccourcis accessibles en appuyant sur la touche **menu** puis en sélectionnant l'instruction souhaitée dans la rubrique correspondante.

Par exemple pour introduire l'instruction de contrôle **If...Then...Else...EndIf** dans le programme, il suffit d'appuyer sur la touche **menu** puis de choisir ensuite 4 : **Contrôle** puis enfin 3 : **If...Then...Else...EndIf**.



L'instruction est ensuite recopiée dans l'éditeur de programmes, on utilise le pavé de navigation pour compléter les champs correspondants aux différentes étapes de l'algorithme.



Recopier totalement dans l'éditeur de programme, l'ensemble des instructions ci-contre correspondant à l'algorithme d'Euclide.

Remarque : Sur l'unité nomade, lors de l'écriture des crochets [], penser à utiliser la touche de direction vers la gauche pour écrire à l'intérieur.

Par ailleurs, lors de l'écriture entre crochets, les paramètres sont à séparer par une virgule par exemple Disp[a,b] La virgule disparaît pour laisser place à un espace vide après enregistrement du programme.

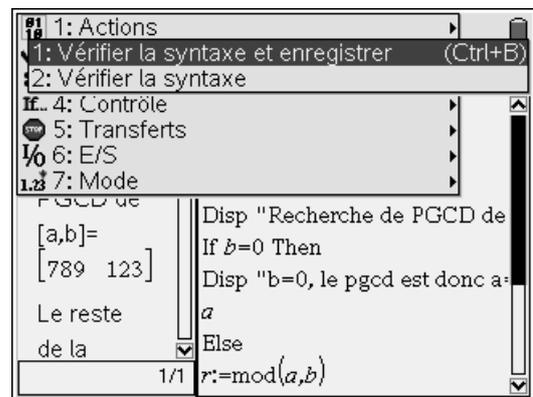
```

Define pgcd(a,b)=
Func
Local r
Disp "Recherche de PGCD de [a,b]=", [a b]
If b=0 Then
Disp "b=0, le pgcd est donc a=", a
Else
r:=mod(a,b)
Disp "Le reste de la division de a par b est r=", r
Disp [a b], " --> ", [b r]
pgcd(b,r)
EndIf
EndFunc
    
```

Vérifier la syntaxe et enregistrer la fonction

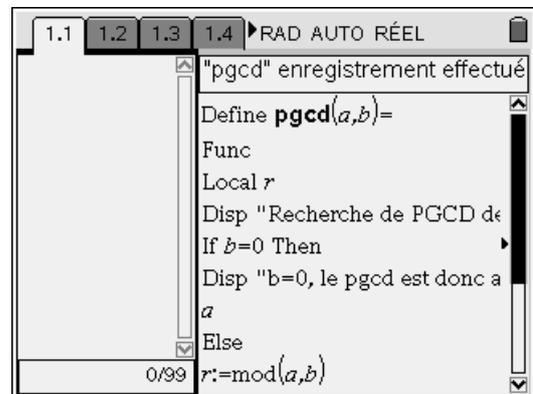
Appuyer ensuite sur la touche , puis choisir 2 : **VÉRIFIER LA SYNTAXE ET ENREGISTRER**. Choisir à nouveau 1 : **VÉRIFIER LA SYNTAXE ET ENREGISTRER**.

Si aucune erreur de syntaxe n'est à corriger dans le programme, l'enregistrement est effectué. En cas d'erreur, le curseur d'édition est renvoyé à l'endroit du programme où une erreur est détectée.



Appuyer sur la touche  pour valider.

Si tout se passe bien, un message inscrit en haut du texte du programme vous informe que l'enregistrement a été effectué.



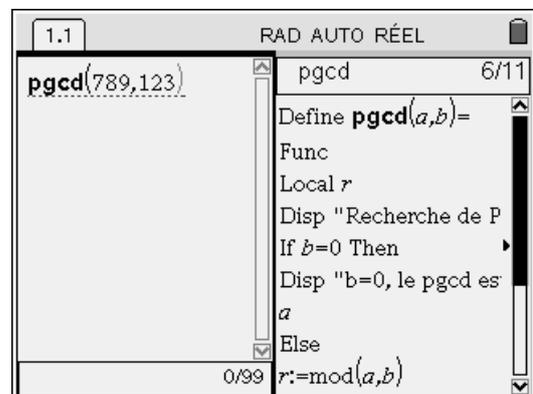
Pour utiliser le programme :

Appuyer sur la touche   pour changer de fenêtre. Lorsque celle-ci est active, elle est entourée d'un rectangle noir.

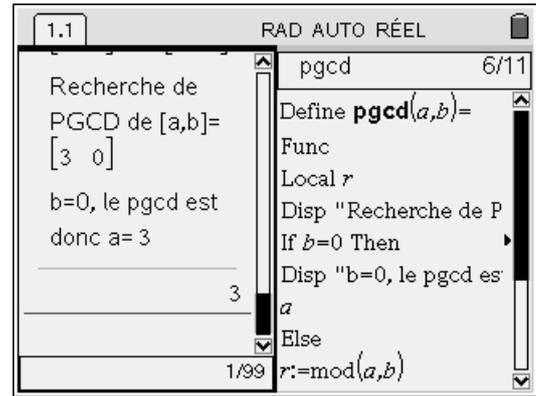
Il est alors possible d'utiliser le programme comme une instruction de calcul « intégrée » à l'unité nomade.

On souhaite ici, déterminer le pgcd des nombres 798 et 123, on tape donc l'instruction **pgcd(798,123)** puis on valide l'instruction en appuyant sur la touche .

Il est également possible de rappeler le programme **pgcd** en appuyant sur la touche  puis en complétant les parenthèses par les nombres *a* et *b*.



On obtient l'ensemble des quotients et restes des divisions euclidiennes successives, l'utilisation du pavé de navigation permet d'observer l'ensemble des résultats.



Remarque : Si l'on ne souhaite pas obtenir les affichages intermédiaires, on pourra adopter la fonction suivante qui donne directement le pgcd de deux nombres a et b .

```

Define pgcd(a,b)=
Func
If b=0 Then
  a
Else
  pgcd(b,mod(a,b))
EndIf
EndFunc
    
```

Second exemple : Pile ou face

On se propose d'écrire une fonction qui simule le lancer d'une pièce de monnaie. Au nombre aléatoire généré, on associera les nombres (0 : Pile) et (1 : Face) Dans un premier temps, la fonction **lancer** comptera le nombre de piles et faces, dans une seconde étape, la fonction **tirages** effectuera un tirage de n pièces que l'on répètera N fois. On exploitera ensuite cette fonction dans le tableur afin de réaliser un graphe rapide pour superposer aux données, une courbe de Gauss théorique.

Un programme et une fonction sont des éléments similaires. La différence essentielle, réside dans le fait que le résultat renvoyé par une fonction peut-être utilisé dans une représentation graphique ou un tableur.

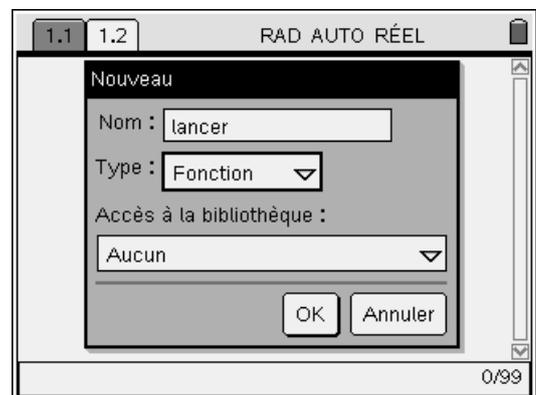
Créer un nouveau classeur classeur et choisir l'application Calculs :

puis choisir 6 : **Nouveau classeur**, puis 1 : **Calculs**.

Choisir 9 : **Fonctions et Programmes** puis 1 : **Éditeur de programmes** et enfin 1 : **Nouveau**.

Compléter le champ correspondant au nom de la fonction, appuyer sur la touche pour passer au champ suivant et préciser qu'il s'agit d'une fonction. Appuyer à nouveau sur pour donner l'accès à la bibliothèque public (visible depuis le catalogue et accessible à tous les classeurs) ou privé.

Appuyer une nouvelle fois sur pour mettre le mot OK en surbrillance et confirmer en appuyant sur la touche .



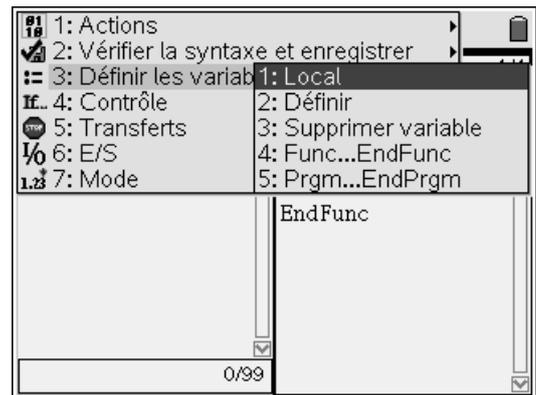
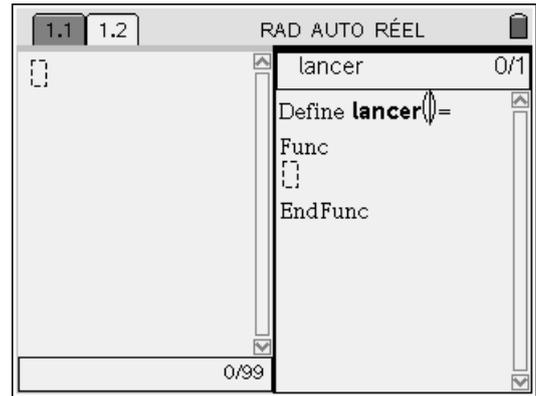
L'écran se partage verticalement en deux parties l'éditeur de **fonctions/programmes** est entouré d'un rectangle noir, matérialisant ainsi sa sélection.

Utiliser ensuite le pavé de navigation (flèche vers le bas), pour placer le curseur entre les balises **Func...EndFunc**

Le résultat du lancer de pièce est contrôlé par une boucle de variable *i*. Cette variable étant aussi susceptible d'être utilisée par le système, afin d'éviter les conflits, nous précisons qu'il s'agit d'une variable locale.

Appuyer sur la touche , puis 3 : **Définir les variables** et choisir 1 : **Local**, puis compléter l'instruction comme indiqué par le texte de la fonction.

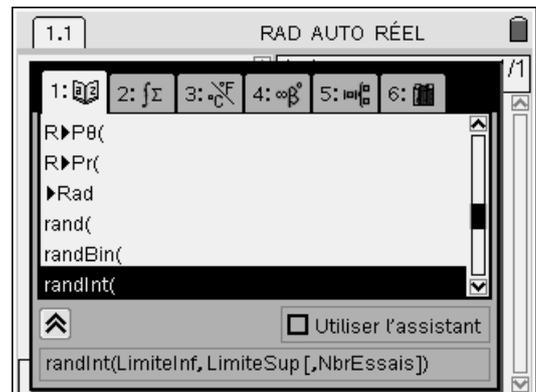
Les instructions de contrôle se saisissent en appuyant sur  puis 4 : **Contrôle**.



Compléter le texte de la fonction comme indiqué ci-contre.

```
Define lancer(n)=
Func
Local i,np
np:=0
For i,1,n
  If randInt(0,1)=0 Then
    Disp "Pile"
    np:=np+1
  Else
    Disp "Face"
  EndIf
EndFor
Disp np,"Piles"
Disp n-np,"Faces"
np
EndFunc
```

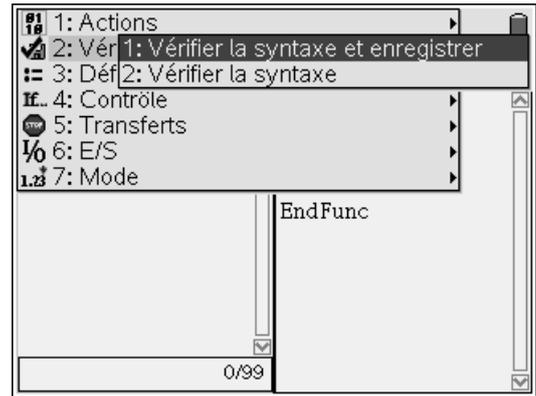
L'instruction **randint()** générant un nombre aléatoire entier peut être obtenue à partir du catalogue  puis . Observer la syntaxe de l'instruction dans la partie grisée.



Vérifier la syntaxe et enregistrer.

Pour valider l'écriture de la fonction, appuyer sur la touche  puis choisir 2 : **Vérifier la syntaxe et enregistrer**. Choisir une seconde fois 1 : **Vérifier la syntaxe et enregistrer**.

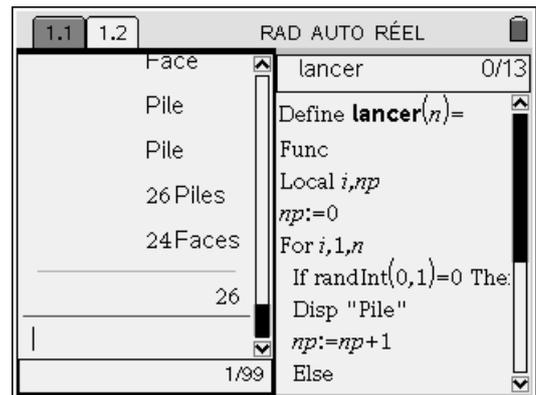
Valider par  pour confirmer.



Utiliser la fonction.

Utiliser les touches   pour sélectionner la partie gauche de la feuille de calcul.

Appuyer sur la touche  pour rappeler la variable **lancer()** et choisir un nombre d'essais, puis appuyer sur la touche  pour réaliser le calcul.



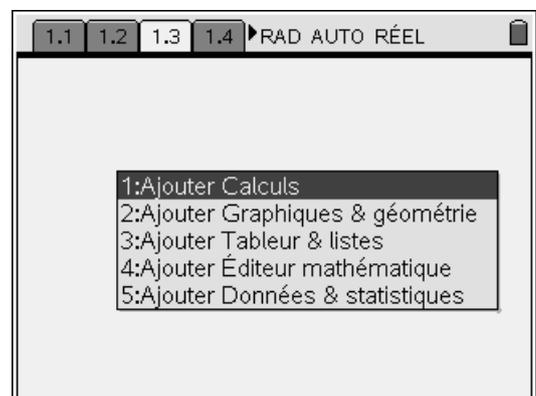
Amélioration de la fonction lancer()

La fonction **tirages()** ci-dessous réalise le même travail, mais simule n lancers. L'instruction **frequency()**, comptera le nombre de piles ou faces obtenus et enregistrera le résultat dans une liste. La fonction **tirages** placée dans un tableau nous permettra ainsi de réaliser N tirages de n pièces de monnaie. On pourra ainsi construire l'histogramme des données et le comparer à une courbe de Gauss théorique. L'avantage de cette méthode est d'être particulièrement rapide pour un grand nombre de valeurs.

Elle sera donc à privilégier en classe pour initier les élèves à la notion d'échantillonnage et de fluctuation d'échantillonnage en statistiques.

Création de la fonction tirages()

Appuyer sur la touche   pour insérer une nouvelle page du classeur et choisir 1 : **Calculs**.

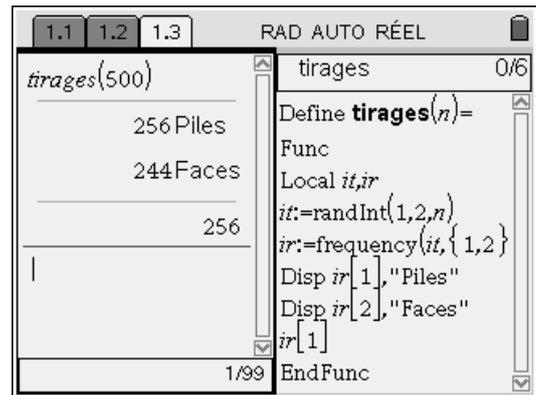


Créer une nouvelle fonction dans l'éditeur de fonctions

Écrire la fonction **tirages()** dont le texte figure ci contre.

```
Define tirages(n)=
Func
Local it,ir
it:=randInt(1,2,n)
ir:=frequency(it,{1,2})
Disp ir[1],"Piles"
Disp ir[2],"Faces"
ir[1]
EndFunc
```

Tester la fonction dans la partie gauche de la feuille de calcul.



Utilisation de la fonction dans un tableur.

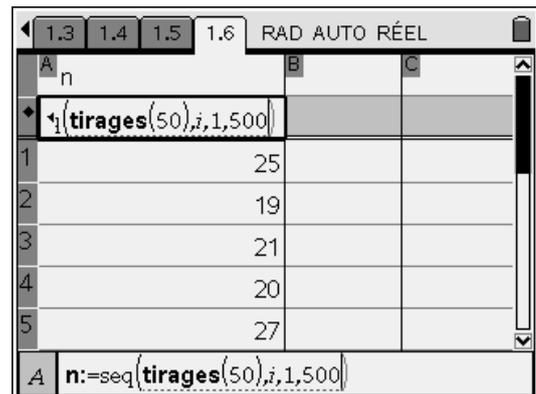
Appuyer sur la touche **ctrl** **I** pour insérer une nouvelle page du classeur et choisir 3 : **Ajouter Tableur & Listes**.

Placer le curseur dans la colonne **A** à côté de la lettre **A** et nommer la colonne **n**.

Dans cette cellule sous la lettre **A**, écrire la formule de calcul **seq(tirages(50),i,1,500)**.

Remarque : dans la cellule d'édition d'une formule applicable à toute la colonne, le signe = est automatiquement recopié par la machine.

La variable **tirages** peut être rappelée en appuyant sur la touche **storVar**.



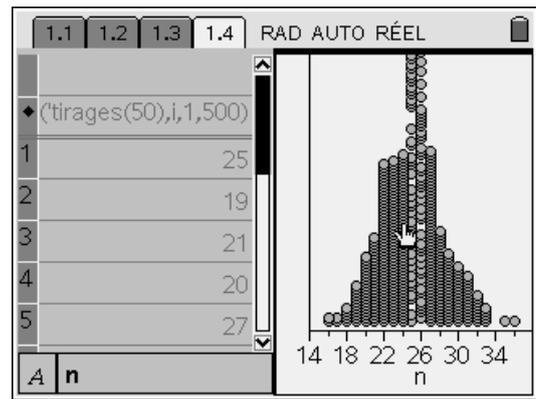
Seul le résultat du calcul est affiché, le mot pile ou face ne l'est pas.

Appuyer deux fois sur la touche de direction vers le haut pour sélectionner la colonne **A**.

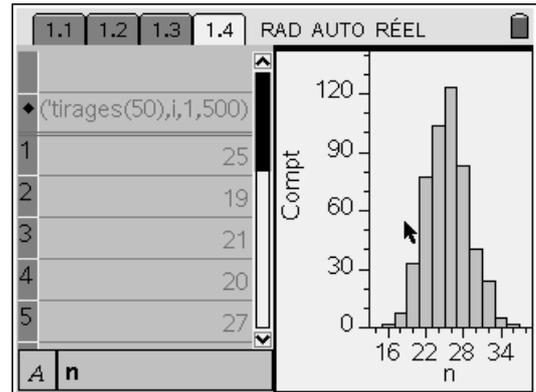
Appuyer sur la touche **menu** et choisir 3 : **Données**, puis 4 : **Graphe rapide**.



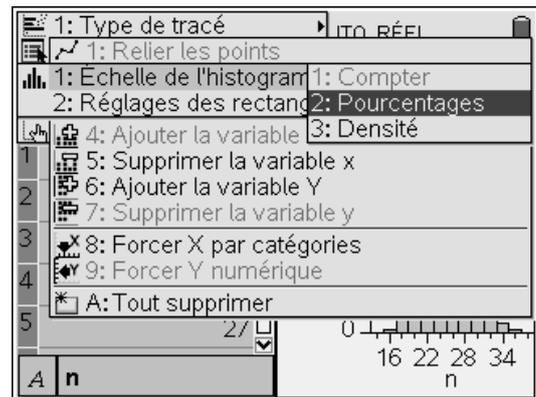
L'écran se partage en deux et les données sont représentées graphiquement.



Appuyer sur la touche $\text{\textcircled{menu}}$ puis choisir 1 : **Type de tracé** puis enfin 3 : **Histogramme**.



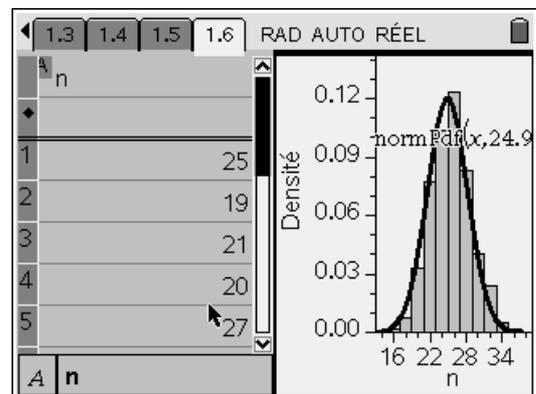
Pour afficher l'axe des ordonnées sous forme de densité, appuyer sur la touche $\text{\textcircled{menu}}$ puis choisir 2 : **Propriétés du tracé** puis à nouveau 2 : **Propriété de l'histogramme** et enfin 1 : **Echelle de l'histogramme**, choisir 3 : **Densité**.



Appuyer à nouveau sur la touche $\text{\textcircled{menu}}$ et choisir 4 **Analyse** puis enfin 9 : **Afficher la fonction Normale DdP**

Redimensionner la représentation si nécessaire.

Essai avec 500 valeurs.



Conclusion : La programmation d'une fonction avec un argument se révèle particulièrement intéressante pour être utilisée dans l'ensemble des applications de la plate forme Ti-nspire™.