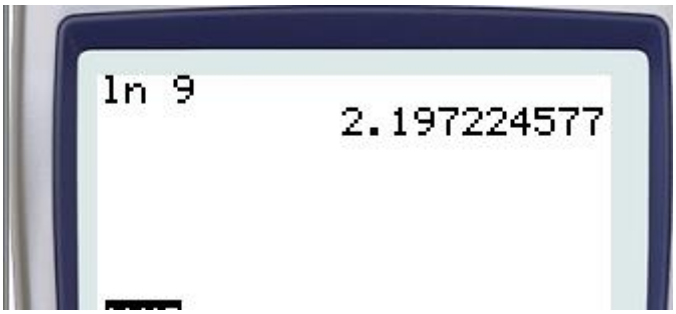
Mais comment font les calculatrices ou les ordinateurs pour calculer puis afficher le résultat d'un calcul ?

Prenons un exemple simple : ln(9)
Avec une calculatrice



Avec un langage de programmation comme python et sur un ordinateur :

```
*** Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500 32 bit (Intel)]
>>>
>>> log(9)
2.1972245773362196
>>>
```

Dans les 2 cas il y a 2 aspects principaux à considérer.
Le premier : Quel est l'algorithme qui permet de calculer un ln ?
Le deuxième : comment les valeurs sont-elles « enregistrées » dans la machine ?

Premier aspect
Pour un élève de terminale ou d'université, il connait des suites ou des séries qui convergent vers un ln, par exemple on a

Pour x ∈ [0,1[, $\ln(1+x) = \sum_{k=0}^{+\infty} (-1)^k \frac{x^{k+1}}{k+1}$

Pour x>1 on utilise $-\ln\left(\frac{1}{x}\right) = \ln(x)$

Et il y en a d'autres.
Ces formules pourraient être utilisées dans un algorithme.
Mais non, il y a mieux…
L'algorithme de Cordic !

On se donne x compris entre 1 et 10.
On affecte la valeur ln 10 à la variable y.
Pour i allant de 0 jusqu'à 10 (ou jusqu'à N), faire
$1 + 10^{-i} \rightarrow z$;
Tant que $xz \leq 10$ faire
$xz \rightarrow x$;
$y - \ln(z) \rightarrow y$;

Dans cet algorithme, y=ln(x) à la fin, et plusieurs valeurs de ln doivent être mémorisées au préalable, celles de $1 + 10^{-i}$.

Pour plus de détails voici un document complet de Mme ***Nicole Bopp***, mais attention la démonstration est loin d'être évidente.
http://numerisation.irem.univ-mrs.fr/ST/IST06003/IST06003.pdf

Cet algorithme, conçu vers 1960 par ***E.Volder***, qui était ingénieur, permettait de faire des calculs trigonométriques en temps réel par des calculateurs dans des situations variées comme les assistants de navigation en aviation.
C'est cet algorithme qui sera utilisé sur certains processeurs et sur les calculatrices scientifiques.

Deuxième aspect
Dans la machine, comment est représenté un nombre ?
Toute information est codée par une suite de chiffres binaires appelés bits.
Un octet est constitué de 8 bits.
Avec 4 octets soit 32 bits on peut coder à priori 2^{32}=4 294 967 296 objets différents, et avec 8 octets 2^{64}
Un nombre réel ne fait pas exception, il est codé sur 4 ou 8 octets selon une norme bien précise.
C'est la ***norme IEEE754***.
Cette norme dit comment on code et comment on décode, mais aussi explique la manière de faire les opérations élémentaires.
Voici un cours sur cette norme :
https://www.irisa.fr/sage/jocelyne/cours/precision/precision-2016.pdf
ou encore ce site qui permet de voir le codage d'un décimal avec cette norme :
http://www.binaryconvert.com/result_float.html?decimal=048046049

Avec 32 bits, on peut donc coder à priori 4 294 967 296 nombres décimaux différents.
Ainsi les autres nombres, qui ne peuvent pas être codés exactement, sont remplacés par leur arrondi, par exemple le plus proche parmi ceux qui sont codés exactement.
D'où des erreurs d'arrondis qui peuvent se propager lorsqu'il y a des boucles dans un algorithme.
Voilà un document à propos des arrondis sur la calculatrice TI-Nspire :
erreur-arrondi-TNS21.pdf
On s'aperçoit, entre autre, que l'associativité de l'addition n'est plus vérifiée sur les arrondis.
Ou encore celui-ci, par ***H.Lehning***, qui explique comment on pourrait calculer ou contrôler les erreurs commises grâce à l'arithmétique des intervalles, ou par une méthode stochastique :
approximation-ordi.PDF

Si x est un nombre à coder, il est compris entre 2 nombres codés exactement par la norme IEEE754.
Ainsi, on remplace chaque valeur par l'intervalle correspondant, et au lieu de faire des opérations sur les valeurs exactes, on les fait sur les intervalles.
On comprend qu'après de multiples opérations, l'intervalle correspondant au résultat final puisse être peu précis. Il a l'avantage de mesurer précisément les erreurs.
Un cours sur l'arithmétique des intervalles :
http://perso.ens-lyon.fr/nathalie.revol/talks/EJCACF04.pdf

Pour résumer, pour afficher le résultat d'un calcul, même simple, il faut un algorithme performant qui donne « théoriquement » la valeur exacte du résultat et une norme judicieuse de codage des nombres décimaux qui permet de minimiser les erreurs d'arrondis et d'éviter des bugs.
Mais minimiser ce n'est pas exclure et la propagation des arrondis peut générer des erreurs qui peuvent être catastrophiques dans certains cas comme le signale *H.Lehning*.