# Lecture 2

niceguy

2023-07-28

## Functions

```
# Syntax of Functions
f <- function(x){
  x**2 + 1
}
# equivalent to f(x) = x^2 + 1
cat(f(5))
```

```
## 26
```

We can also use intermediate variables

```
g <- function(x){
  y <- 2*x
  y**2 - x
}
cat(g(6))
```

```
## 138
```

Now this allows us to write a more advanced function for finding square roots for

$$Ax^2 + Bx + C = 0$$

```
sq.roots <- function(A, B, C){
  disc <- B**2 - 4*A*C
  if(disc > 0){
    r1 <- (-b-sqrt(disc))/(2*a)
    r2 <- (-b+sqrt(disc))/(2*a)
    cat(r1, r2)
  }
  else if(disc == 0){
    cat(-b/(2*a))
  }
}
```

There *is* homework, but there is no way I'm doing it. Prof Guerzhoy also mentioned how a function *returning* a string (e.g. `"Hi"`) is different from a function *printing* a string (e.g. `cat("Hi")`).

## Vectors

A vector is an ordered $n$-tuple of elements of the same type.

```r
v <- c(TRUE, FALSE, TRUE, TRUE) # c returns a vector
v[1]
```

```
## [1] TRUE
```

```r
v[2]
```

```
## [1] FALSE
```

```r
v[3]
```

```
## [1] TRUE
```

```r
v[4]
```

```
## [1] TRUE
```

There are also functions

```r
v <- c(2, -50, 2, 4)
sort(v)
```

```
## [1] -50   2   2   4
```

```r
length(v)
```

```
## [1] 4
```

```r
max(v)
```

```
## [1] 4
```

```r
min(v)
```

```
## [1] -50
```

```r
v <- c(2, -1, 2, 2, 5, 1)
unique(v)
```

```
## [1]  2 -1  5  1
```

Similar to C, we have and `&`, or `|` and not `!`. We can now do funky things like

```r
v <- c(5, 2, -1, 1)
v[v>0]
```

```
## [1] 5 2 1
```

```r
v>=0 & v<= 3
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```r
v[v>=0 & v<= 3]
```

```
## [1] 2 1
```

```r
cat(v, 3)
```

```
## 5 2 -1 1 3
```