	Python	R
Arithmatic Operators		
Assignment : Defining a number	a = 10 ; b = 25	a <- 10 ; b <- 25
Addition	a + b	a + b
Subtraction	a - b	a - b
Multiplication	a * b	a * b
Division	a / b	a / b
Power : a^b	a ** b	a ^ b
Remainder	a % b	a %% b
Integer Division	a // b	a %/% b
Logical Operators		
Short-Circuit logical AND	a and b	a && b
Short-Circuit logical OR	a or b	a || b
Element-wise logical AND	a and b	a & b
Element-wise logical OR	a or b	a | b
Logical NOT	! a	! a
Relational Operators		
Equal	a == b	a == b
Less than	a < b	a < b
Greater than	a > b	a > b
Less than or equal	a <= b	a <= b
Greater than or equal	a >= b	a >= b
Not Equal	a != b	a != b
Root and Logarithm		
Square root	math.sqrt(a)	sqrt(a)
Logarithm, base e	math.log(a)	log(a)
Logarithm, base 10	math.log10(a)	log10(a)
Exponential function	exp(a)	exp(a)
Round off	math.round(a)	round(a)
Generate random numbers		
Uniform Distribution	random.uniform((10,))	runif(10)
Uniform numbers between a and b	random.uniform(a,b,(10,))	runif(10, min = a, max = b)
Normal Distribution	random.standard_normal((10,))	rnorm(10)
Vectors		
Sequences		
1,2,3,.....,10	range(1,11)	seq(10) or 1:10
1,4,7,10	arange(1,11,3)	seq(1,10,by = 3)
10,7,4,1	arange(10,0,-3)	seq(from= 10, to= 1,by =-3)
Reverse	a[: : -1]	rev(a)
Concatenation		
Concatenate two vectors	concatenate((a, a))	c(a,a)
Add elements 1,2,3,4	concatenate((range(1,5), a), axis = 1)	c(1:4 , a)
Repeating		
1 2 3 , 1 2 3	concatenate((a, a))	rep(a,times=2)
1 1 1, 2 2 2, 3 3 3	a.repeat(3)	rep(a,each=3)
1 , 2 2 , 3 3 3	a.repeat(a)	rep(a,a)
Maximium & Minimum		
Pairwise max	maximum(a,b)	pmax(a,b)
max of all values in two vectors	concatenate((a, b)) . max()	max(a,b)
Vector Multiplication		
Multiply two vectors	a * a	a * a
Vector dot product a . b	dot(a , b)	

	Python	R
Matrices		
Defining a Matrix		
Define a matrix	a = array([[2,3] , [4,5]])	matrix(c(2,3,4,5) , dim=c(2,2)) rbind(c(2,3) ,c(4,5))
Concatenation (matrices) ; rbind and cbind		
Bind rows	concatenate((a,b) , axis = 0) vstack((a,b))	rbind(a,b)
Bind columns	concatenate((a,b) , axis = 1) hstack((a,b))	cbind(a,b)
Array creation		
0 filled array	zeros((3,3))	matrix(0,3,3)
Any number filled array	array([[9,9] , [9,9]])	matrix(9,3,3)
Identity Matrix	identity(3)	diag(1,3)
Indexing and accessing elements (Python: slicing)		
Element 2,3 (row,col)	a[1,2]	a[2,3]
First row	a[0,]	a[1,]
First column	a[: , 0]	a[,1]
Remove one column	a.take([0,2,3],axis = 1)	a[,-2]
Clipping : replace element	a[: , 0] = 99	a[,1] <- 99
Transpose and inverse		
Transpose	a.conj() . transpose()	t(a)
Determinant	linalg.det(a)	det(a)
Inverse	linalg.inv(a)	solve(a)
Rank	rank(a)	rank(a)
Sum (Python : Numpy)		
Sum of each column	a.sum(axis=0)	apply(a,2,sum)
Sum of each row	a.sum(axis=1)	apply(a,1,sum)
Sum of all elements	a.sum()	sum(a)
Cumulative sum (Columns)	a.cumsum(axis=0)	apply(a,2cumsum)
Sorting		
Sort all elements flat	a.ravel() .sort()	t(sort(a))
Sort each column	a.sort(axis=0)	apply(a,2,sort)
Sort each row	a.sort(axis=1)	t(apply(a,2,sort))
Sort, return indices	a.ravel() .argsort()	order(a)
Maximun and Minimum		
max in each column	a.max(0)	apply(a,2,max)
max in each row	a.max(1)	apply(a,1,max)
max in array	a.max()	max(a)
Matrix - and elementwise - multiplication		
Elementwise multiplication	a * b or multiply(a,b)	a * b
Dot product	matrixmultiply(a,b)	a %*% b
Outer product	outer(a,b)	outer(a,b) or a %o% b
Cross product	cross(a,b)	crossprod(a,b)
Matrix Size : Dimensions		
Matrix dimensions	a.shape	dim(a)
Number of columns	a.shape[1]	ncol(a)
Number of elements	a.size	prod(dim(a))
Number of dimensions	a.ndim	

	Python	R
Data Frames (Python : Pandas)		
Creating a Dataframe		
Dataframe	DF = pd.DataFrame({'a': [10, 5, 2], 'b': [20, 5, 30]})	a = c(10,5,2) b = c(20,5,30) DF = data.frame(a,b)
Import csv file as dataframe	DF = pd.read_csv("file.csv")	DF <- read.table("file.csv")
Accessing elements		
Display first 5 rows	df.head()	head(df,5)
Select first column	df['a']	df$a or df[1]
Common functions		
Summary	df.describe	summary(df)
Check datatypes	df.dtypes	str(df)
Structure	df.shape	str(df)
Programming		
Reading from a file	f = load("data.txt")	f <- read.table("data.txt")
Import library functions	from pylab import *	library(mtcars)
Comment	#	#
Print	print a	print(a)
Script file extension	.py or .ipynb(Jupyter notebook)	.R
Conditionals :		

If -else statement	if m > 0 :
	a = 100
	else :
	a = 0
		if (m > 0){
		a = 100
		}
		else {
		a = 0
		}

Loops :
For loop
	for i in range(1,6) :
	print(i)
	print(i*5)
		for (i in 1:5){
		print(i)
		print(i*5)
		}

Functions		

Function : Even or odd	def even_odd(x):
	if x%2 == 0:
	y = 'Even'
	else :
	y = 'Odd'
	return y
		even_odd = function(x){
		if (x %% 2 == 0){
		y = "Even"
		}
		else{
		y = "Odd"
		}
		return(y)
		}