Python ↔ R Handy Mental Mapping Cheat Sheet

Goal: quick “how do I think about this in the other language?” mappings. R idioms ↔ Python (NumPy/pandas).

Assignment + printing

R mindset / idiom	Python analogue
`x <- 1 x # or x = 1 print(x)`	`x = 1 x # in scripts print(x)`

1-indexing vs 0-indexing

R mindset / idiom	Python analogue
`x <- c(10,20,30) x[1] # 10 x[3] # 30`	`x = [10,20,30] x[0] # 10 x[2] # 30`

Slicing (inclusive vs exclusive end)

R mindset / idiom	Python analogue
`x <- 1:10 x[2:5] # 2,3,4,5 x[-1] # drop first`	`import numpy as np x = np.arange(1, 11) x[1:5] # 2,3,4,5 x[1:] # drop first`

Vectorization is default

R mindset / idiom	Python analogue
`x <- c(1,2,3) x^2 x + 10`	`import numpy as np x = np.array([1,2,3]) x**2 x + 10`

Recycling (R) ↔ broadcasting (NumPy)

R mindset / idiom	Python analogue
`c(1,2,3) + c(10,20) # 11,22,13 (recycles)`	`import numpy as np np.array([1,2,3]) + np.array([10,20]) # ValueError unless shapes broadcast. # Use np.resize / explicit broadcasting.`

Missing values

R mindset / idiom	Python analogue
`x <- c(1, NA, 3) is.na(x) mean(x, na.rm=TRUE)`	`import numpy as np x = np.array([1, np.nan, 3.0]) np.isnan(x) np.nanmean(x)`

Logical filtering

R mindset / idiom	Python analogue
`x <- 1:10 x[x %% 2 == 0]`	`import numpy as np x = np.arange(1, 11) x[x % 2 == 0]`

Apply family ↔ map/apply

R mindset / idiom	Python analogue
`lapply(list(1:3, 4:6), mean) sapply(1:5, function(i) i^2)`	`list(map(np.mean, [np.arange(1,4), np.arange(4,7)])) [i**2 for i in range(1,6)]`

Pipes (%>%) ↔ method chaining

R mindset / idiom	Python analogue
`library(dplyr) df %>% filter(x > 0) %>% group_by(g) %>% summarise(m=mean(y))`	`df[df['x'] > 0] .groupby('g', as_index=False) .agg(m=('y','mean'))`

Data frame column access

R mindset / idiom	Python analogue
`df$x # or subset(df, x > 0)`	`df['x'] # boolean mask df[df['x'] > 0]`

Factor ↔ categorical

R mindset / idiom	Python analogue
`g <- factor(c('A','B','A')) levels(g)`	`import pandas as pd g = pd.Categorical(['A','B','A']) g.categories`

Grouping: dplyr verbs ↔ pandas

R mindset / idiom	Python analogue
`df %>% group_by(g) %>% summarise(n=n(), m=mean(y))`	`df.groupby('g').agg(n=('g','size'), m=('y','mean')).reset_index()`

Reshaping: pivot_longer/wider ↔ melt/pivot

R mindset / idiom	Python analogue
`library(tidyr) pivot_longer(wide, cols=c(a,b), names_to='k', values_to='v') pivot_wider(long, names_from=k, values_from=v)`	`wide.melt(id_vars=['id'], value_vars=['a','b'], var_name='k', value_name='v') long.pivot(index='id', columns='k', values='v').reset_index()`

Formula modeling syntax (lm/glm) ↔ statsmodels

R mindset / idiom	Python analogue
`fit <- lm(y ~ x1 + x2 + g, data=df) summary(fit)`	`import statsmodels.formula.api as smf fit = smf.ols('y ~ x1 + x2 + C(g)', data=df).fit() fit.summary()`

Randomness: set seed

R mindset / idiom	Python analogue
`set.seed(42) rnorm(3)`	`import numpy as np rng = np.random.default_rng(42) rng.normal(size=3)`

In-place vs copy (gotcha)

R mindset / idiom	Python analogue
`df2 <- df # usually copy-on-modify behavior`	`df2 = df.copy() # make an explicit copy # beware chained assignment; prefer .loc`

Common “where is this function?” mental map

R mindset / idiom	Python analogue
`mean(), sd(), cor() # stats package (base) # dplyr verbs in dplyr`	`np.mean(), np.std() scipy.stats.* for tests pandas for wrangling seaborn/matplotlib for plots`