# **# Programowanie w R – kurs podstawowy**

## # Podstawowe operacje na listach

# Lista może przechowywać dowolną ilość obiektów, dowolnego typu.

# Tworzenia listy

**a = 1:10**

**b = letters[1:10]**

**M = matrix(a, 2, 5)**

**RD=data.frame(a,b)**

**L = list(a, b, M, RD, 3)**

# Wyświetlmy L na podglądzie

[[ ]]

# Pojedynczy nawias kwadratowy […] służy do cięcia list i w przypadku pojedynczej wartości da listę jednoelementową.

**L[2]**

# zapiszmy ją jako y – powstanie nowa lista 1-elementowa, a nie wektor

**y=L[2]**

# Można się odwołać do kilku elementów – da nam krótsze listy

**w=L[1:3]**

**z=L[c(2, 4)]**

# Do elementów listy odwołujemy się podwójnym nawiasem kwadratowym [[…]]

**L[[2]]**

# zapiszmy jako z – powstanie wektor, a nie nowa lista 1-elementowa

**v=L[[2]]**

# Możemy odwołać się elementu w środku elementu listy

**L[[2]][1]**

# Możemy zmodyfikować jego wartość

**L[[2]][1] = "aa"**

# Nazywanie elementów listy

**names(L)= c("A1", "B2", "C3", "D4", "E5")**

**# Odwoływanie się elementów listy z użyciem nazwy**

**L["A1"]**

**L[["A1"]]**

# Odwoływanie się elementów listy z użyciem nazwy lub znakiem $

**L$A1**

# jest równoznaczne z podwójnym nawiasem [[ ]]

**L[["A1"]]**

# Dodawanie kolejnego elementu do listy

**L[[6]] = LETTERS[1:3]** # doda, ale bez nazwy elementu

**L[["F6"]] = LETTERS[1:3]**

Lub **L$F6 = LETTERS[1:3]** # doda od razu z nazwą

# Łączenie list i innych elementów - różnice

**L2 = c(L, a)** # dostawi elementy wektora

**L3 = c(L, RD)** # dostawi elementy RD czyli 2 wektory

**L4 = list(L, a)** # dostawi jako drugą listę i powstanie lista 2-elementowa

# Elementem listy może być inna lista

**L5 = list(L, 5, "Kot")**

# Odwołując się [] do pierwszego elementu dostaniemy listę 1-elem (lista)

**a=L5[1]**

# Odwołując się [[]] do pierwszego elementu dostaniemy listę 6-elem

**b=L5[[1]]**

# Odwołując się do drugiego elementu pierwszego elementu – lista 1-elem

**c= L5[[1]][2]**

# Odwołując się [[]] do drugiego elementu pierwszego elementu – wektor

**d= L5[[1]][[2]]**

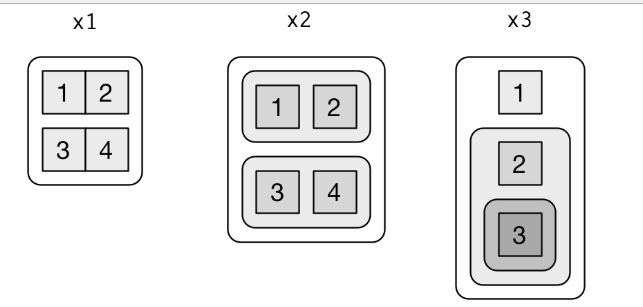
# Odwołując się [[]][2] do drugiego elementu drugiego elementu – liczba

**e**= **L5[[1]][[2]][2]**

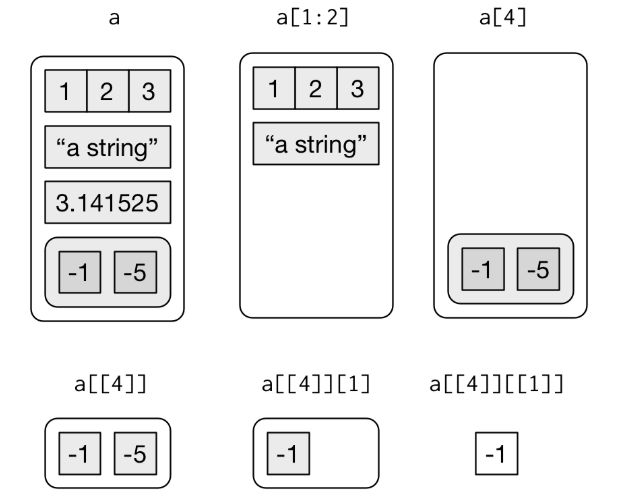
x1 <- **list**(**c**(1, 2), **c**(3, 4))

x2 <- **list**(**list**(1, 2), **list**(3, 4))

x3 <- **list**(1, **list**(2, **list**(3)))



a <- list(a = 1:3, b = "a string", c = pi, d = list(-1, -5))



**# Zadania**

# Zadanie 1. Utwórz zmienną x o wartości 100

**x = 100**

# Zadanie 2. Utwórz wektor V, składający się z liczb od 1 do 12

**V = 1:12**

# Zadanie 2. Utwórz z wektora V macierz M o wymiarach 3x4

**M=matrix(V, 3,4)**

# Zadanie 4. Utwórz listę o nazwie lista, składającą się ze wszystkich elementów utworzonych w Zad. 1, 2, 3

**lista = list(x, V, M)**

# Zadanie 5. Nadaj nazwy „zmienna”, „wektor” i „macierz” elementom listy lista

**names(lista)=c("zmienna", "wektor", "macierz")**

# Zadanie 6. Wytnij pierwszy i trzeci element listy lista. Zapisz jako listę lista2

**lista2 = lista[c(1,3)]**

# Zadanie 7. Zapisz drugi element listy lista jako wektor V1

**V1 = lista[[2]]**

**V1 = lista[["wektor"]]**

# Zadanie 8. Wyświetl element listy lista o nazwie macierz skorzystając z $

**lista$macierz**

# Zadanie 9. Dodaj do listy o nazwie lista czwarty element o nazwie litery, który zawiera wszystkie małe litery alfabetu. Wyświetl listę lista.

**lista$litery = letters**

**lista**

# Zadanie 10. Utwórz nową listę 2-elementową lista3 składającą się z listy lista2 i dowolnego wektora 5-elementowego

**lista3=list(lista2, 1:5)**

## # Zajęcia 8. Funkcje

# **---------------------Funkcje łączenia i tworzenia**

c()

cbind()

rbind()

matrix()

data.frame()

list()

seq()

rep()

sample()

rm()

# **---------------------Badanie i zmiana struktury danych**

names()

colnames()

rownames()

dim()

head()

tail()

nrow()

ncol()

str()

summary()

# **---------------------Edycja danych**

order()

merge()

split()

subset()

# **---------------------Sprawdzanie typu**

is.vector()

is.matrix()

is.data.frame()

is.factor()

# **---------------------Zmiana typu zmiennej**

as.character()

as.numeric()

as.integer()

as.factor()

# **---------------------?**

# jak wygląda funkcja subset()

subset(Studenci, subset = zaliczenie)

subset(Studenci, select = test1)

subset(Studenci, zaliczenie, test1) # zadziała

subset(Studenci, test1, zaliczenie) # nie zadziała!!!

subset(Studenci, select=test1, subset=zaliczenie) # zadziała

?subset

# to drop=FALSE od razu

# jak wygląda funkcja matrix()

matrix(1:12, 3, 4)

matrix(1:12, 4, 3)

matrix(1:12, **nrow** = 3)

matrix(1:12, **ncol** = 4)

matrix(1:12, 3, 4, byrow=TRUE)

?matrix

matrix(1:12, 3, 4,list(c("a","b","c"),c(1:4))) # błąd, bo najpierw jest byrow

matrix(1:12, 3, 4,,list(c("a","b","c"),c(1:4))) # dać przecinek i puste miejsce

matrix(1:12, 3, 4, dimnames=list(c("a","b","c"),c(1:4))) # dać dimnames

# **------------Tworzenie własnych funkcji**

# Definiowanie własnych funkcji przy pomocy składni

nazwa.funkcji = function(parametr1, parametr2, ...) {

...

}

# funkcja „pierwiastek”

pierwiastek = function(x) {

sqrt(x)

}

pierwiastek(16)

# funkcja „wektor n-elementowy” od 1 do n

wektor = function(n){

w=1:n

w

}

# funkcja „wektor n-elementowy”, ale elementy losowo pomieszane

wektor = function(n){

sample(1:n, n)

}

# funkcja „pitagoras”

pitagoras = function(a,b){

c=sqrt(a^2+b^2)

paste("Nasz trójkąt ma następującą długość boków",a,b,c)

}

# funkcja „suma kwadratów”

suma\_kwadratow = function(x,y) {

x^2+y^2

}

# funkcja „suma potęg”

suma\_poteg = function(x, y, potega\_x=2, potega\_y=2) {

x^potega\_x + y^potega\_y

}

# funkcja „minima” – 3 najmniejsze elementy wektora

minima = function(x){

posortowe=sort(x)

posortowe[1:3]

}

# funkcja „minima” - n najmniejszych elementów wektora

minima = function(x, n){

posortowe=sort(x)

posortowe[1:n]

}

minima = function(x, n=1) # n możemy ustawić na 1

minima = function(x, od, do){ # możemy ustawić 2 parametry

posortowe=sort(x)

posortowe[od:do]

}

minima = function(x, od=1, do=1){

posortowe=sort(x)

posortowe[od:do]

}

minima = function(x, od=1, do=length(x)){

posortowe=sort(x)

posortowe[od:do]

}

# funkcja „sortowanie kolumn” – ostatnia kolumna na początek

sort\_kol = function(df){

df[,c(ncol(df),1:(ncol(df)-1))]

}

# funkcja „n rzutów kostką”

kostka = function(n){

sample(1:6,n,replace=T)

}

# **-----------Zmienne lokalne i globalne**

# mamy zmienną y=10 w środowisku

y <- 10

f <- function(x) {

x + y

}

f(10) = 20

# mamy zmienną y=10 w środowisku, ale w funkcji ją zmieniamy

y <- 10

f <- function(x) {

y <- 5

x + y

}

# f(10) = 15

# nie mamy zmiennej y w środowisku, ale tworzymy ją w funkcji

f <- function(x) {

y <- 5

x + y

}

# f(5)=10, ale y nie pojawia się jako zmienna globalna

> y

BŁĄD: nie znaleziono obiektu 'y'