BIOS611-HW3

Yangzhenyu Gao

2025-09-12

Problem 1

The flag `-i` in Docker tells the container to keep STDIN open even if not attached.
The flag `-t` allocates a pseudo-TTY (terminal).
The abbreviation **tty** comes from *teletypewriter*, a device from the mid-20th century used to send typed input to computers.
This is much younger than the scientific acceptance of atoms, which dates back about 120 years (early 20th century, with modern physics and chemistry).

Problem 2

```
x <- 0
y <- 0

z1 <- local({
  x <- 10
  y <- 15
  x + y
})

x
```

```
## [1] 0
```

```
y
```

```
## [1] 0
```

```
z2 <- {
  x <- 10
  y <- 15
  x + y
}

x
```

```
## [1] 10
```

```
y
```

```
## [1] 15
```

```
z1
```

```
## [1] 25
```

```
z2
```

```
## [1] 25
```

Explanation:
- `local({...})` evaluates code in a temporary environment; `x` and `y` are not available outside.
- `{...}` evaluates in the current environment; `x` and `y` remain defined after execution.

Problem 3

```
list_of_functions <- list()
for(i in 1:10){
  list_of_functions <- c(list_of_functions, function() i)
}
for(f in list_of_functions){
  print(f())
}
```

```
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
```

The above prints 10 ten times because each function closes over the variable `i`, which after the loop ends equals 10.

To fix this, use local function above:

```r
list_of_functions <- list()
for(i in 1:10){
  list_of_functions[[i]] <- local({
    j <- i
    function() j
  })
}

for(f in list_of_functions){
  print(f())
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

Problem 4

Blocks {} can look identical but mean different things:
- In a **function**, {} is the function body, evaluated only when the function is called.
- In a **loop**, {} is the loop body, evaluated each time the loop iterates.
The difference is in scope and timing of evaluation.

Problem 5

```r
with_list <- function(lst, expr){
  env <- list2env(lst, parent = parent.frame())
  eval(substitute(expr), envir = env)
}

q <- 7
result <- with_list(list(a=10, b=100), {
  a + b + q
})

result
```

```
## [1] 117
```

```r
exists("a")
```

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

Explanation:
- `list2env` creates an environment from the list.
- `eval(substitute(expr), envir=env)` evaluates the expression inside that environment.
- After running, `x` and `y` are not defined in the global environment.

Problem 6

```r
`%mid%` <- function(a,b) (a+b)/2
z <- c(1,2,3) %mid% c(9,9,9)
z
```

```
## [1] 5.0 5.5 6.0
```

Explanation:
- R allows custom infix operators of the form `%name%`.
- Backticks are needed to define `%mid%` as a function name.
- `z` is the elementwise average: `c(5, 5.5, 6)`.

Problem 7

- **apt**: System package manager for Debian/Ubuntu Linux; installs software at the OS level.
- **pip**: Python package manager; installs Python libraries from PyPI.
- **install.packages**: R's function for installing packages from CRAN.

Problem 10

- My github repo: https://github.com/YangzhenyuGao/611-Data-Science-Project