Dicts — The Full Picture

// Creating dicts

```
{"key": value}        # literal
{}                    # empty dict
dict()                # empty dict
dict(key=value)       # keyword args
dict(zip(keys, vals)) # from two lists
d["key"] = value      # add or update
```

// Reading values

```
d["key"]              # KeyError if missing
d.get("key")          # None if missing
d.get("key", default) # default if missing
"key" in d            # True if key exists
"key" not in d        # True if missing
```

// Modifying

```
d["key"] = value      # add or update
d.update(other)       # merge dict in
d.update(key=value)   # keyword update
d.setdefault("k", v)  # add if missing
d.pop("key")          # remove, return val
d.pop("key", default) # safe remove
d.popitem()           # remove last pair
d.clear()             # empty the dict
```

// Inspecting

```
len(d)                # number of pairs
d.keys()              # view of all keys
d.values()            # view of all values
d.items()             # view of all pairs
val in d.values()     # check a value
d.copy()              # independent copy
```

// Iterating

```
for key in d:         # keys only
for key in d.keys():  # explicit
for val in d.values():# values only
for k, v in d.items():# pairs — most common
```

// Nested dicts

```
roster = {
  "Raven": {"score": 85, "active": True},
  "Wolf":  {"score": 74, "active": False}
}
roster["Raven"]           # inner dict
roster["Raven"]["score"]  # inner value

for name, data in roster.items():
    print(name, data["score"])
```

// Safe access patterns

```
# check before lookup
if "key" in d:
    print(d["key"])

# get with default
d.get("key", "fallback")

# safe pop
d.pop("key", None)

# safe outer + inner
d.get("outer", {}).get("inner")
```

// Common mistakes

```
d["missing"]          # KeyError — use get()
d = d.update({...})   # None — don't assign
backup = d            # same dict — use .copy()
x in d                # keys only — use .values()
{"k":1, "k":2}        # last wins, no error
del d[k] in loop      # RuntimeError — new dict
d.pop("k")            # KeyError — add default
{}                    # dict — use set() for set
```