Answers to Practice Problems — Lesson 5

Answer 1: Budget picker

Prompt: With `prices = [11, 4, 8, 3, 12]` and `budget = 9`, print the cheapest item within budget and its index.

```
prices = [11, 4, 8, 3, 12]
budget = 9

best_price = float('inf')
best_idx = -1

for i, p in enumerate(prices):
    if p <= budget and p < best_price:
        best_price = p
        best_idx = i

if best_idx == -1:
    print("Nothing affordable.")
else:
    print("Cheapest within budget: index", best_idx, "price", best_price)
```

Output

```
Cheapest within budget: index 3 price 3
```

Answer 2: Two-attribute route

Prompt: Routes are `(name, time, transfers)`. Use `score = -time - 4*transfers`. Print each score and the best route.

```
routes = [("A", 18, 0), ("B", 14, 1), ("C", 12, 2), ("D", 16, 0)]

def route_score(time_min, transfers):
    return -time_min - 4*transfers

best = None
best_s = -float('inf')
```

```
for name, t, k in routes:
    s = route_score(t, k)
    print("Route", name, "score =", s)
    if s > best_s:
        best_s = s
        best = name

print("Best route:", best, "with score", best_s)
```

Output

```
Route A score = -18
Route B score = -18
Route C score = -20
Route D score = -16
Best route: D with score -16
```

Answer 3: Snack tradeoff

Prompt: With `snacks = [(name, fun, weight), ...]`, use `score = fun - 0.75*weight`. Print the best snack.

```
snacks = [
    ("chips", 7, 2.0),
    ("apple", 5, 0.4),
    ("cookie", 6, 0.8),
    ("soda", 4, 1.0),
]

def snack_score(fun, w):
    return fun - 0.75*w

best_name = None
best_s = -float('inf')

for name, fun, w in snacks:
    s = snack_score(fun, w)
    print(name, "score =", s)
    if s > best_s:
        best_s = s
        best_name = name

print("Best snack:", best_name, "with score", best_s)
```

Output

```
chips score = 5.5
apple score = 4.7
cookie score = 5.4
soda score = 3.25
Best snack: chips with score 5.5
```

Answer 4: Filter then best (chores)

Prompt: Chores `[("trash", 3), ("wipe", 2), ("sweep", 5), ("boards", 1)]`, with `time_limit = 3`. Print the most time-consuming feasible chore.

```
chores = [("trash", 3), ("wipe", 2), ("sweep", 5), ("boards", 1)]
time_limit = 3

best_name = None
best_time = -1

for name, minutes in chores:
    if minutes <= time_limit and minutes > best_time:
        best_time = minutes
        best_name = name

if best_name is None:
    print("No chore fits the time limit.")
else:
    print("Do:", best_name, "(", best_time, "min )")
```

Output

```
Do: trash ( 3 min )
```

Answer 5: Challenge — General selector

Prompt: Write `best_choice(options, score_fn)` that returns the element with maximum `score_fn(item)`. Test on routes and snacks.

```
def best_choice(options, score_fn):
    best_item = None
    best_s = -float('inf')
    for item in options:
```

```
        s = score_fn(item)
        if s > best_s:
            best_s = s
            best_item = item
    return best_item, best_s

# Test 1: routes (name, time, transfers) with score = -time - 5*transfers
routes = [("A", 18, 0), ("B", 14, 1), ("C", 12, 2), ("D", 16, 0)]
route_fn = lambda r: -r[1] - 5*r[2]
best_route, s1 = best_choice(routes, route_fn)
print("Best route:", best_route, "score =", s1)

# Test 2: snacks (name, price, fun) with score = fun - price
snacks = [("chips", 2.5, 7), ("apple", 1.2, 5), ("cookie", 1.8, 6)]
snack_fn = lambda it: it[2] - it[1] # fun - price
best_snack, s2 = best_choice(snacks, snack_fn)
print("Best snack:", best_snack, "score =", s2)
```

Output

```
Best route: ('D', 16, 0) score = -16
Best snack: ('chips', 2.5, 7) score = 4.5
```