# Controle Structures - Solutions

## Solution 1: Conditional Statements (if-else)

```
income = float(input("Enter your income: "))
tax_status = input("Enter your tax status: ")

if income > 50000:
    tax = income * 0.25
else:
    tax = income * 0.15

print(f"Your tax amount is: ${tax:.2f}")
```

---

## Solution 2: Nested if-else with Multiple Conditions

```
age = int(input("Enter your age: "))
income = float(input("Enter your income: "))
debt = float(input("Enter your debt amount: "))

if age >= 21:
    if income > 30000:
        if debt <= 10000:
            print("You are eligible for the loan.")
        else:
            print("You are not eligible for the loan due to high debt.")
    else:
        print("You are not eligible for the loan due to low income.")
else:
    print("You are not eligible for the loan due to age.")
```

---

## Solution 3: While Loop for Savings Target

```
savings_target = float(input("Enter your savings target: "))
balance = 0
months = 0
monthly_savings = 500

while balance < savings_target:
    balance += monthly_savings
    months += 1

print(f"It will take {months} months to reach your savings target.")
```

---

## Solution 4: Loan Repayment Simulation with While Loop

```
loan_balance = 50000
monthly_payment = 1000
monthly_interest_rate = 0.03
total_interest_paid = 0
```

```python
months = 0

while loan_balance > 0:
    monthly_interest = loan_balance * monthly_interest_rate
    total_interest_paid += monthly_interest
    loan_balance = loan_balance + monthly_interest - monthly_payment
    months += 1

    if loan_balance < monthly_payment:
        total_interest_paid += loan_balance * monthly_interest_rate
        loan_balance = 0
        months += 1

print(f"It will take {months} months to repay the loan.")
print(f"Total interest paid over the life of the loan:
${total_interest_paid:.2f}")
```

---

## Solution 5: For Loop Over a List

```python
stock_prices = [100, 105, 98, 110, 120]
total_gain_loss = 0

for i in range(1, len(stock_prices)):
    total_gain_loss += stock_prices[i] - stock_prices[i-1]

print(f"Total gain/loss: ${total_gain_loss:.2f}")
```

---

## Solution 6: For Loop with Condition

```python
numbers = [1, 2, 3, 4, 5, 6]
for number in numbers:
    if number % 2 == 0:
        print(f"The square of {number} is {number ** 2}")
```

---

## Solution 7: Nested For Loop with Matrix

```python
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

for row in matrix:
    row_sum = 0
    for item in row:
        row_sum += item
    print(f"Sum of row: {row_sum}")
```

---

## Solution 8: Range-based For Loop

```python
for i in range(1, 21):
    if i % 3 == 0:
        print(i)
```

---

## Solution 9: Break and Continue

```
while True:
    number = int(input("Enter a number: "))

    if number > 100:
        print("Success")
        break

    if number < 0:
        print("Invalid")
        continue
```

---

## Solution 10: List Comprehension

```
grades = [85, 92, 88, 76, 95]
updated_grades = [grade + 5 if grade < 90 else grade for grade in grades]
print(updated_grades)
```

# More exercises on **list comprehension**

---

Solutions to additional list comprehensions exercises

## Solution 1: Convert Temperatures

Convert the list of temperatures from Celsius to Fahrenheit using list comprehension.

```
celsius = [0, 20, 37, 100]
fahrenheit = [(9/5) * temp + 32 for temp in celsius]
print(fahrenheit)
```

**Output:**

```
[32.0, 68.0, 98.60000000000001, 212.0]
```

---

## Solution 2: Filter and Square Odd Numbers

Use list comprehension to create a new list of squares of only the odd numbers.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
squared_odds = [num ** 2 for num in numbers if num % 2 != 0]
print(squared_odds)
```

**Output:**

```
[1, 9, 25, 49, 81]
```

---

## Solution 3: Extract Vowels from a String

Use list comprehension to extract vowels from the string.

```
text = "List comprehensions are very powerful!"
vowels = [char for char in text if char.lower() in 'aeiou']
print(vowels)
```

**Output:**

```
['i', 'o', 'e', 'e', 'e', 'i', 'o', 'a', 'e', 'e', 'o', 'e', 'u']
```

---

## Solution 4: Flatten a Nested List

Use list comprehension to flatten a list of lists into a single list.

```
nested_list = [[1, 2, 3], [4, 5], [6, 7, 8, 9]]
flattened = [item for sublist in nested_list for item in sublist]
print(flattened)
```

**Output:**

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

---

## Solution 5: Dictionary from List

Use list comprehension to convert a list of tuples into a dictionary.

```
countries = [("USA", 331002651), ("India", 1380004385), ("China", 1439323776)]
country_population = {country: population for country, population in
countries}
print(country_population)
```

**Output:**

```
{'USA': 331002651, 'India': 1380004385, 'China': 1439323776}
```

```
In [ ]:
```