Controlling Program Flow - MAD 102 Week 3 Notes

Hia Al Saleh

September 18th, 2024

Contents

1 The Selection Structure **3**

2 Boolean Expression **3**

3 Booleans and Equality Operator **3**

4 Relational Operators **4**

5 Controlling Program Flow **4**
5.1 If Statement 4

6 Flow Charting **5**

7 Dual Outcome **5**

8 Multiple Outcomes **6**

9 Detecting Ranges **6**

10 Describing Complex Conditions **7**

11 Logical Operators: `and`, `or`, and `not` **7**

12 Truth Tables **7**

13 Decision Tables **7**

14 Binary Trees **8**

15 Resulting Code **8**

16 Nested Conditional Statements **8**

17 Membership Operators 8

18 Identity Operator 9

19 Ternary Operation 9

1 The Selection Structure

- A selection structure evaluates a condition, which is an expression that's true or false.
- This allows you to specify different courses of action based on the evaluation:
  - Do one thing if true.
  - Do something else if not (false).
- A branch is a sequence of statements that are only executed if a specific condition is met. This branch may never get executed in your program.

2 Boolean Expression

- A selection structure depends on a condition.
- This is an expression describing the relationship between two values that's evaluated when it appears in program code.
- A Boolean expression evaluates to true or false.
- Named after George Boole, who developed an extensive system of logic based on true and false conditions and their consequences.

3 Booleans and Equality Operator

- Booleans are used to compare values.
  - Are you old enough to drive?
  - Is the correct username entered?
  - Did I successfully retrieve the information from the server?
- To see if an answer is equivalent to an expected value, use the equality operator `==`.
- This returns a value of either true or false.
- **NOTE:** Many students confuse the mathematical operator `=` with the equality operator `==`. Remember that `=` is the assignment operator in coding.

4 Relational Operators

Common relational operators include:

- `==`: Checks if two values are equal.
- `<`: Less than.
- `>`: Greater than.
- `<=`: Less than or equal to.
- `>=`: Greater than or equal to.
- `!=`: Not equal to.

The `==` operator is a common relational operator. Here are some additional relational operators used in programming:

- `age < 60`: Checks if the age is less than 60.
- `hours > 40`: Checks if the hours are greater than 40.
- `region == "Ontario"`: Checks if the region is equal to Ontario.
- `status != "denied"`: Checks if the status is NOT denied.
- `quantity <= 10`: Checks if the quantity is 10 and under (includes 10).
- `grade >= 90`: Checks if the grade is greater than or equal to 90.

NOTE: The relational operator for checking if two values are equal is `==`. A single equals sign `=` is the assignment operator.

5 Controlling Program Flow

- The most common way of controlling a program's flow is to determine what a program will do based on decisions.
- These decisions can be simple or complex.
- The decision process is facilitated using an `if` statement.

5.1 If Statement

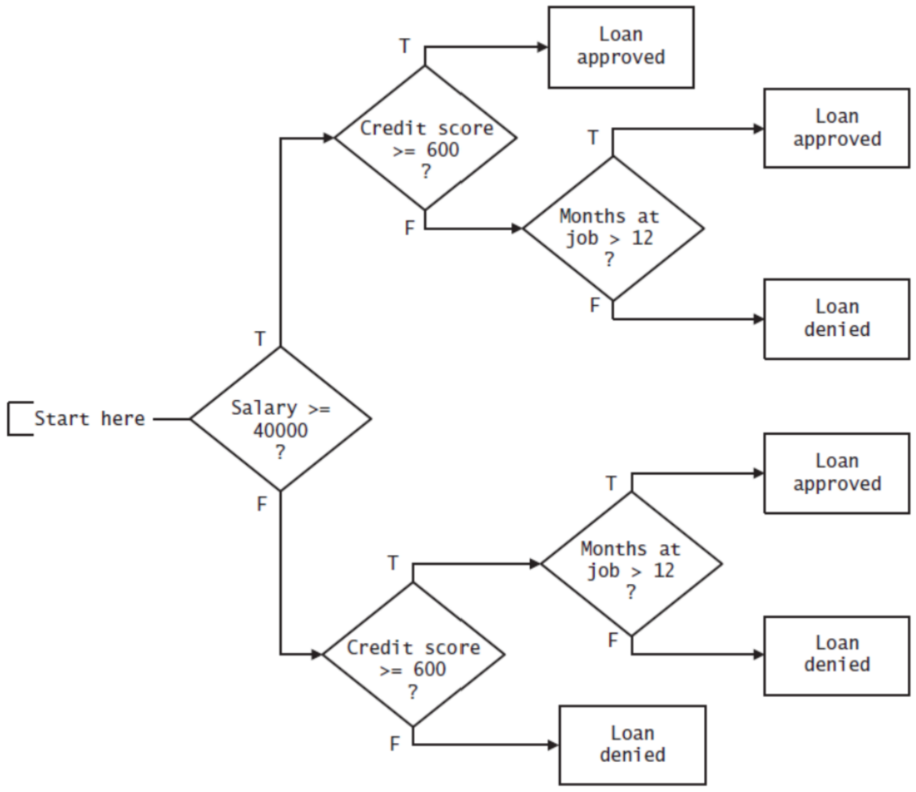
- The simplest selection structure is one in which an action is taken if a condition evaluates to true, but no action is taken if the condition evaluates to false.
- This is a single-outcome section.

```
# Python syntax
if condition:
    # action if condition is true
else:
    # action if condition is false
```

NOTE: An indentation is a tab – it is not a series of spaces.

6 Flow Charting

- The diamond shape is the standard symbol for flow charting. It represents decision points with single outcomes.
- True Branch and False Branch lead to further actions based on evaluation.



7 Dual Outcome

- A dual outcome is a selection statement where you perform one set of instructions if a condition evaluates to true, otherwise perform a second set of instructions.

- Dual outcomes work under the principles of Boolean logic – something is either true or false – if true, do this; otherwise, do the other step.
- Dual outcomes use the keywords `if` and `else`.

```
if condition:
    # action if condition is true
else:
    # action if condition is false
```

8 Multiple Outcomes

- When there are more than a single or dual outcomes, use `if`, `elif`, and `else` to represent multiple conditions.
- The first condition is marked with `if`.
- The end condition is marked with `else`.
- All other conditions are marked with `elif`.

```
grade = 55

if grade >= 80:
    print("You have received an A")
elif grade >= 70:
    print("You have received a B")
elif grade >= 60:
    print("You have received a C")
elif grade >= 50:
    print("You have received a D")
else:
    print("You have received an F")
```

9 Detecting Ranges

- The order of your conditional statements allows you to check if a value is within a specified range.
- Each expression indicates the upper range.
- If you fall to the next condition and that evaluates to true, you must be within the specified range.

10 Describing Complex Conditions

- Often, two or more conditions are involved in a decision. Describe the relationship between the conditions.
- Example:
  - A student makes the dean's list for taking 12 credit hours AND having a grade point average of at least 3.5.
  - A movie theater offers a discount to anyone who is under 6 years old OR over 65 years old.
  - An employee gets a bonus vacation day for meeting a sales quota AND not being absent for a three-month period.

11 Logical Operators: `and`, `or`, and `not`

- Logical operators are used when evaluating two or more conditions.
- A complex condition occurs when two or more conditions must be evaluated for an action to take place.
- Example:
  - An employee is eligible for a discount on store items after working two months AND having a perfect attendance record.
- Conditions are joined with:
  - `and` - both conditions must be true.
  - `or` - at least one condition must be true.
  - `not` - negates the condition; true if the value is false.

12 Truth Tables

- Truth tables help sort out complex logical situations and make coding easier.
- A truth table expresses the results of combinations of conditions.

13 Decision Tables

- Decision tables are used for problems with multiple outcomes.
- They state all relevant conditions, true and false combinations of these conditions, and outcomes associated with each combination.

- The number of combinations is 2^n, where n is the number of conditions.
- Example:
  - A bank loan decision based on income, credit score, and employment duration.

14 Binary Trees

- Binary trees trace all combinations by splitting conditions into true and false paths.
- Paths lead to the next condition. Irrelevant conditions don't split into true and false paths.

15 Resulting Code

- The decision table can be translated into code.

```
# Resilting Code
student = ["Mal", "Jayne", "Washburn", "Zoe"]
name = input("Please enter a name:")

if name in student:
    print("Welcone - you are registered")
else:
    print("Welcome - you are not on our class list")
```

16 Nested Conditional Statements

- A branch's statement can hold additional `if-else` statements, known as nested statements.
- Indentation is important.

17 Membership Operators

- Membership operators return a Boolean value (true or false).
- They determine if a specified value is found in a container type (using `in`) or not (using `not in`).

```python
# Membership Operator
firstName = "Edgar"
lastName = "Smith"

anotherName = firstName

if firstName is anotherName:
    print("Same object")
else:
    print("Different object")
```

18 Identity Operator

- The identity operator (`is`) checks if two operands are bound to a single object.
- `is not` is the inverse.
- They do not compare values but check if two variables share the same memory address.

19 Ternary Operation

- A ternary operation is a conditional expression with three operands.
- Also known as a conditional expression.
- Difficult to read; should only be used for simple assignments.

```python
# Ternary Operation
grade = 50
result = "You have passed" if grade >= 50 else "You have
    failed"
print(result)
```