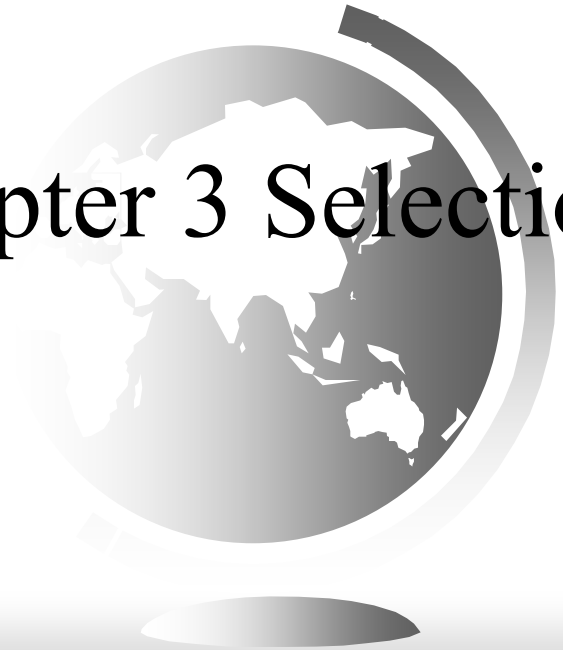


Chapter 3 Selections

Motivations

If you input a negative value for radius in Listing 2.2, ComputeAreaWithConsoleInput.py, the program would print an invalid result. If the radius is negative, you don't want the program to compute the area. How can you deal with this situation?

Objectives

- To write Boolean expressions using relational operators (§3.2).
- To generate random numbers using the **random.randint(a, b)**, **random.randrange(a, b)**, or **random.random()** functions (§3.3).
- To program with Boolean expressions (**AdditionQuiz**) (§3.3).
- To implement selection control using one-way **if** statements (§3.4).
- To implement selection control using two-way **if-else** statements (§3.5).
- To implement selection control with nested **if** and multi-way **if-elif-else** statements (§3.6).
- To avoid common errors in **if** statements (§3.7).
- To program with selection statements (§§3.8–3.9).
- To combine conditions using logical operators (**and**, **or**, and **not**) (§3.10).
- To use selection statements with combined conditions (**LeapYear**, **Lottery**) (§§3.11–3.12).
- To write expressions that use the conditional expressions (§3.13).
- To simplify selection statements using match-case statements (§3.14).
- To understand the rules governing operator precedence and associativity (§3.15).

Boolean Data Types

Often in a program you need to compare two values, such as whether i is greater than j. There are six comparison operators (also known as relational operators) that can be used to compare two values. The result of the comparison is a Boolean value: true or false.

```
b = (1 > 2)
```

Relational Operators

Operator	*Name*
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

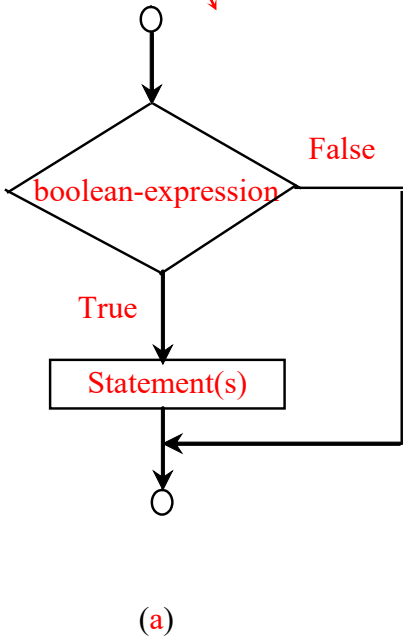
Problem: A Simple Math Learning Tool

This example creates a program to let a first grader practice additions. The program randomly generates two single-digit integers number1 and number2 and displays a question such as "What is 7 + 9?" to the student. After the student types the answer, the program displays a message to indicate whether the answer is true or false.

AdditionQuiz

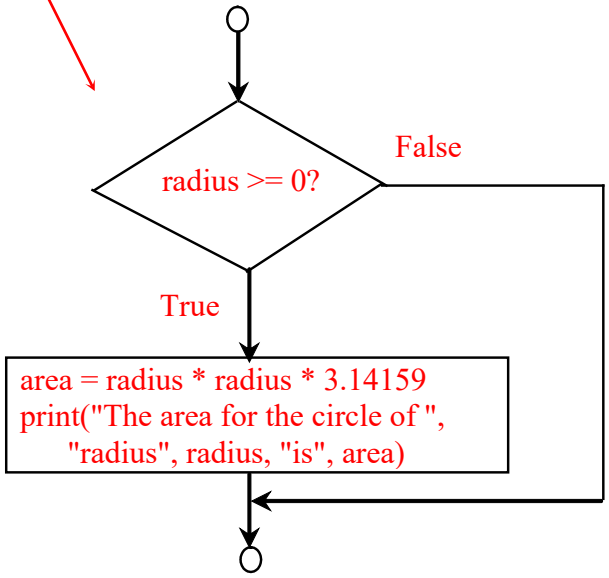
One-way `if` Statements

```
if boolean-expression:
    statement(s)
```



(a)

```
if radius >= 0:
    area = radius * radius * 3.14159
    print("The area for the circle of radius“,
          radius, "is“, area)
```



(b)

Note

```
if i > 0:
print("i is positive")
```

(a) Wrong

```
if i > 0:
    print("i is positive")
```

(b) Correct

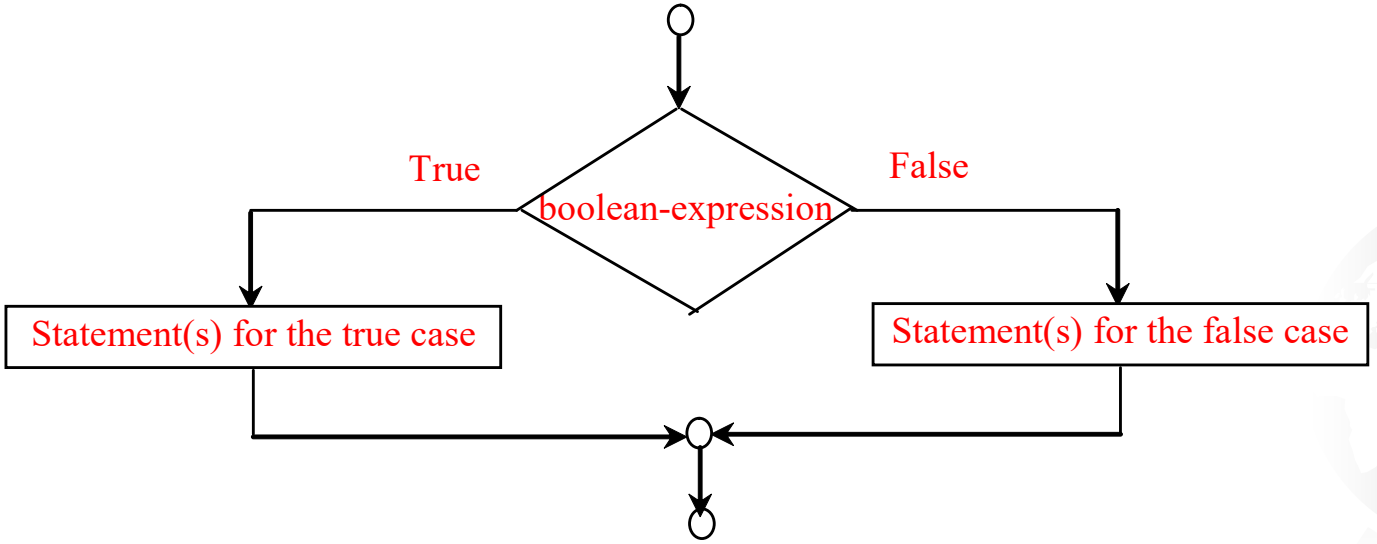
Simple if Demo

Write a program that prompts the user to enter an integer. If the number is a multiple of 5, print HiFive. If the number is divisible by 2, print HiEven.

SimpleIfDemo

The Two-way `if` Statement

```
if boolean-expression:
    statement(s)-for-the-true-case
else:
    statement(s)-for-the-false-case
```



`if...else` Example

```
if radius >= 0:
    area = radius * radius * math.pi
    print("The area for the circle of radius", radius, "is", area)
else:
    print("Negative input")
```

Problem: An Improved Math Learning Tool

This example creates a program to teach a first grade child how to learn subtractions. The program randomly generates two single-digit integers number1 and number2 with number1 >= number2 and displays a question such as "What is 9 – 2?" to the student. After the student types the answer, the program displays a message to indicate whether the answer is correct.

SubtractionQuiz

Multiple Alternative if Statements

```
if score >= 90.0:
    grade = 'A'
else:
    if score >= 80.0:
        grade = 'B'
    else:
        if score >= 70.0:
            grade = 'C'
        else:
            if score >= 60.0:
                grade = 'D'
            else:
                grade = 'F'
```

(a)

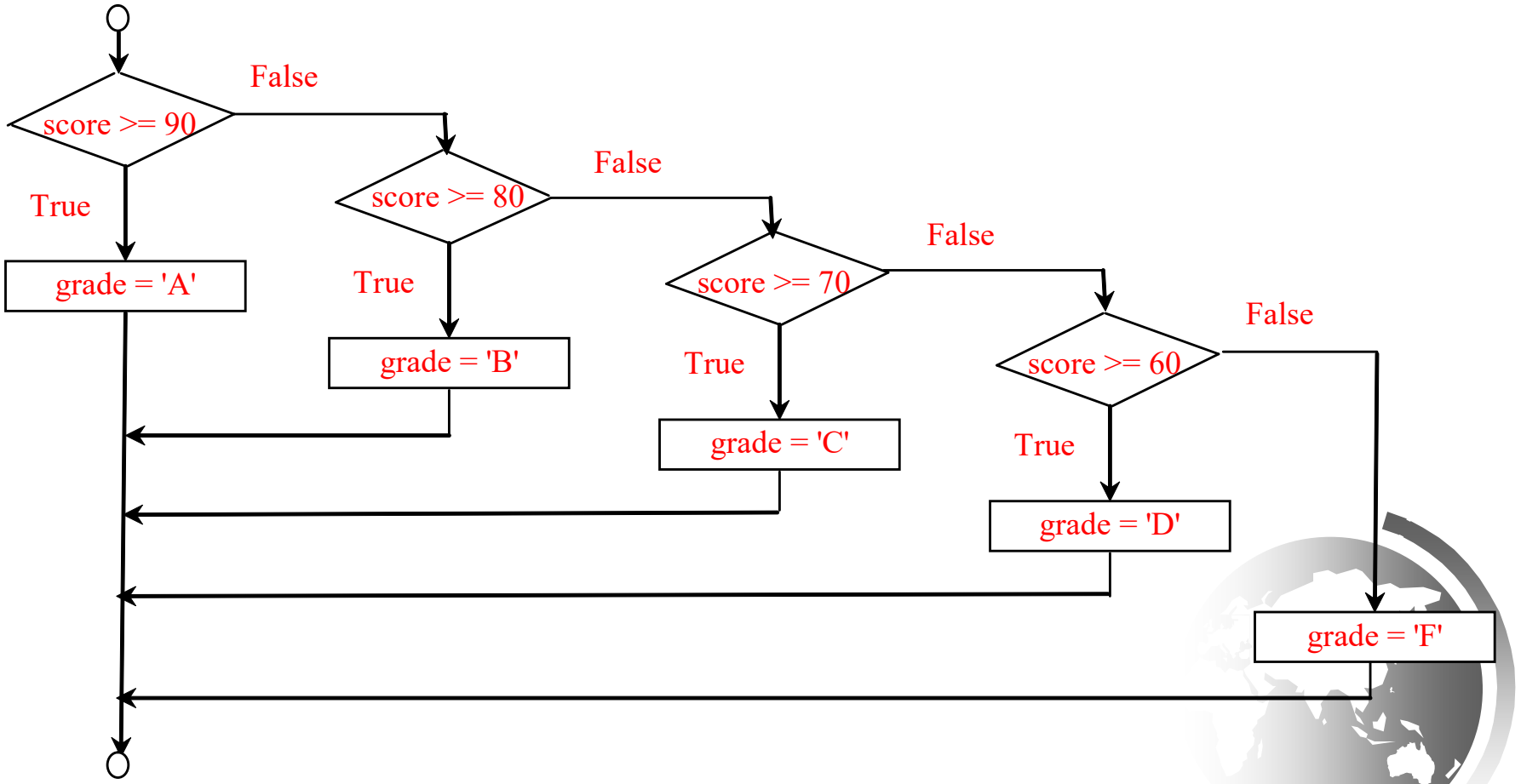
Equivalent

This is better

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

(b)

Flowchart



animation

Trace if-else statement

Suppose score is 70.0

The condition is false

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

animation

Trace if-else statement

Suppose score is 70.0

The condition is false

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

animation

Trace if-else statement

Suppose score is 70.0

The condition is true

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

animation

Trace if-else statement

Suppose score is 70.0

grade is C

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

animation

Trace if-else statement

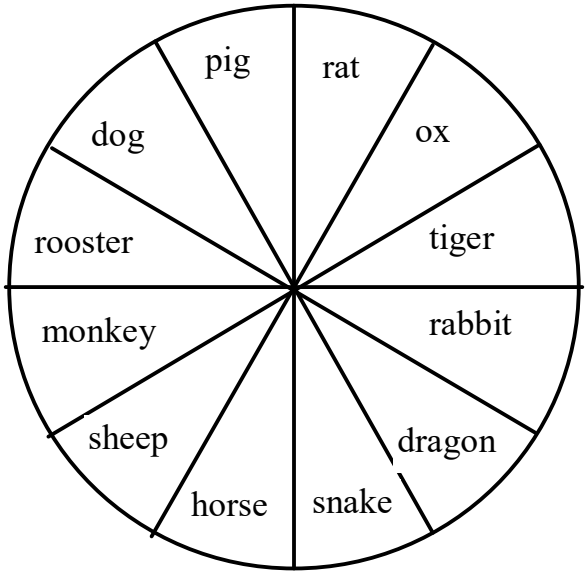
Suppose score is 70.0

Exit the if statement

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

Example

Now let us write a program to find out the Chinese Zodiac sign for a given year. The Chinese Zodiac sign is based on a 12-year cycle, each year being represented by an animal: rat, ox, tiger, rabbit, dragon, snake, horse, sheep, monkey, rooster, dog, and pig, in this cycle.



year % 12 =

0: monkey
1: rooster
2: dog
3: pig
4: rat
5: ox
6: tiger
7: rabbit
8: dragon
9: snake
10: horse
11: sheep

ChineseZodiac

Common Errors

Most common errors in selection statements are caused by incorrect indentation. Consider the following code in (a) and (b).

```
radius = -20

if radius >= 0:
    area = radius * radius * 3.14
print("The area is", area)
```

(a) Wrong

```
radius = -20

if radius >= 0:
    area = radius * radius * 3.14
    print("The area is", area)
```

(b) Correct

TIP

```
if number % 2 == 0:
    even = True
else:
    even = False
```

(a)

Equivalent

This is shorter

```
even = number % 2 == 0
```

(b)

Problem: Body Mass Index

Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters. The interpretation of BMI for people 16 years or older is as follows:

BMI	Interpretation
Below 18.5	Underweight
18.5-24.9	Normal
25.0-29.9	Overweight
Above 30.0	Obese

ComputeAndInterpretBMI

Problem: Computing Taxes

The US federal personal income tax is calculated based on the filing status and taxable income. There are four filing statuses: single filers, married filing jointly, married filing separately, and head of household. The tax rates for 2009 are shown below.

Marginal Tax Rate	Single	Married Filing Jointly or Qualified Widow(er)	Married Filing Separately	Head of Household
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351– $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,525 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 - $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

Problem: Computing Taxes, cont.

```
if status == 0:
    # Compute tax for single filers
elif status == 1:
    # Compute tax for married filing jointly
elif status == 2:
    # Compute tax for married filing separately
elif status == 3:
    # Compute tax for head of household
else:
    # Display wrong status
```

ComputeTax

Logical Operators

Operator	Description
`not`	`logical negation`
`and`	`logical conjunction`
`or`	`logical disjunction`

Truth Table for Operator not

p	not p	Example (assume age = 24, gender = 'F')
`True`	`False`	not (age > 18) is False, because (age > 18) is True.
`False`	`True`	not (gender == 'M') is True, because (grade == 'M') is False.

Truth Table for Operator and

p1	p2	p1 and p2	Example (assume age = 24, weight = 140)
False	False	False	
False	True	False	(age > 18) and (weight >= 140) is False, because weight > 140) is False.
True	False	False	
True	True	True	(age > 18) and (weight <= 140) is True, because (age > 18) and (weight <= 140) are both True.

Truth Table for Operator or

p1	p2	p1 or p2	Example (assume age = 24, weight = 140)
False	False	False	(age > 34) or (weight >= 150) is False, because (age > 34) and (weight >= 150) are both False.
False	True	True	
True	False	True	(age > 14) or (weight < 140) is true, because (age > 14) is True.
True	True	True	

Examples

Here is a program that checks whether a number is divisible by 2 and 3, whether a number is divisible by 2 or 3, and whether a number is divisible by 2 or 3 but not both:

TestBooleanOperators

Problem: Determining Leap Year?

This program first prompts the user to enter a year as an int value and checks if it is a leap year.

A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.

(year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

LeapYear

Problem: Lottery

Write a program that randomly generates a lottery between 0 and 99, prompts the user to enter a number in the same range, and determines whether the user wins according to the following rule:

- If the user input matches the lottery in exact order, the award is $10,000.
- If the user input matches the lottery, the award is $3,000.
- If one digit in the user input matches a digit in the lottery, the award is $1,000.

Lottery

Conditional Expressions

```
if x > 0:
    y = 1
else:
    y = -1
```

is equivalent to

```
y = 1 if x > 0 else -1
```

expression1 **if** boolean-expression **else** expression2

Conditional Operator

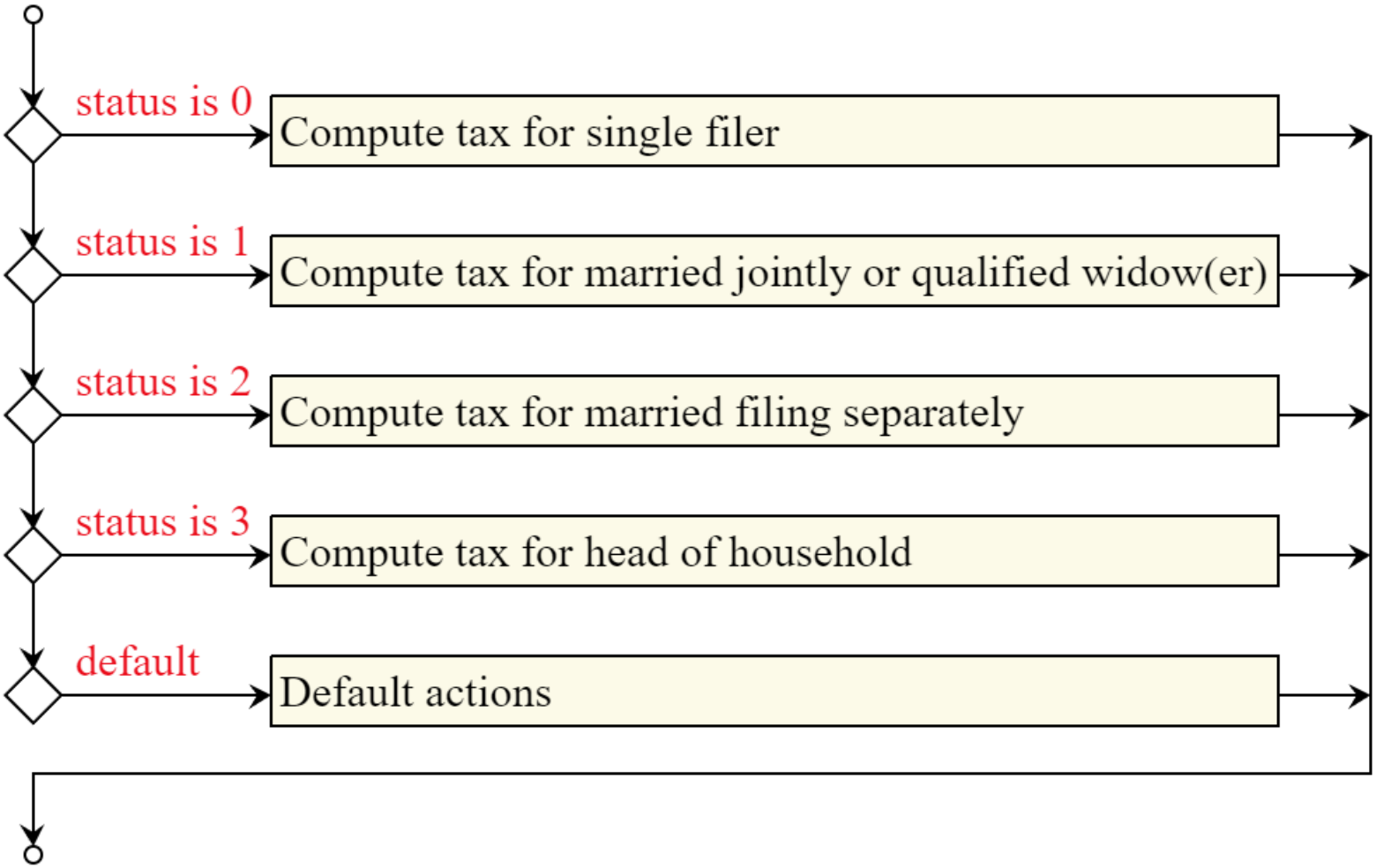
```
if number % 2 == 0:
    print(number, "is even")
else:
    print(number, "is odd")
```

```
print(number, "is even" if (number % 2 == 0)
   else "is odd")
```

match-case Statements

```
case status:
  case 0:  compute taxes for single filers
  case 1:  compute taxes for married file jointly
  case 2:  compute taxes for married file separately
  case 3:  compute taxes for head of household
  case _:
          print("Errors: invalid status")
          sys.exit(1)
```

`match-case` Statement Flow Chart



`match-case` Statement Rules

The case-expression must yield a number, a string, a Boolean value.

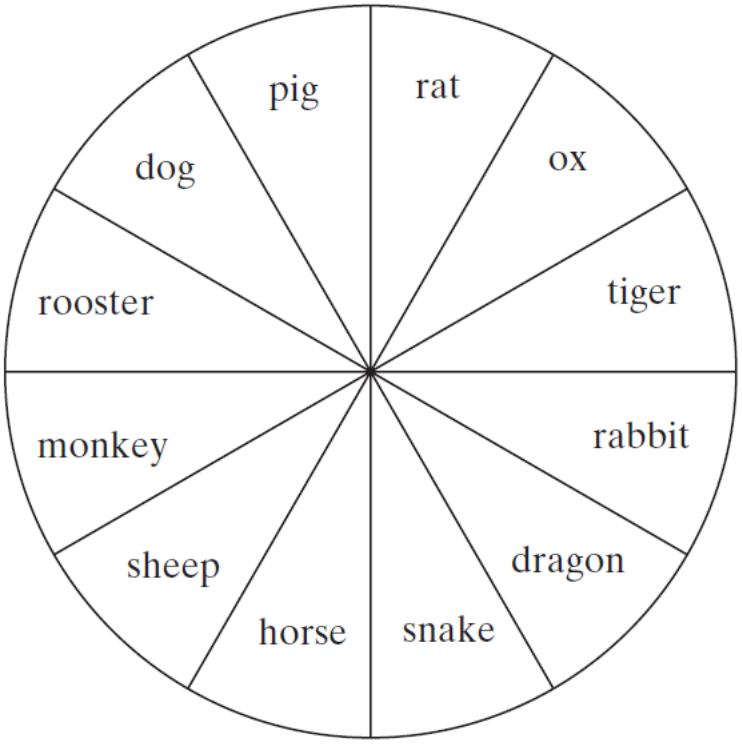
The value1, ..., and valueN must have the same data type as the value of the case-expression. The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression. Note that value1, ..., and valueN are constant expressions, meaning that they cannot contain variables in the expression, such as 1 + x.

```
switch case-expression:
  case value1:  statement(s)1
  case value2: statement(s)2
  ...
  case valueN: statement(s)N
  case _: statement(s)-for-default;
}
```

The default case, which is optional, can be used to perform actions when none of the specified cases matches the expression.

Problem: Chinese Zodiac

Write a program that prompts the user to enter a year and displays the animal for the year.



`year % 12` =

0: monkey
1: rooster
2: dog
3: pig
4: rat
5: ox
6: tiger
7: rabbit
8: dragon
9: snake
10: horse
11: sheep

ChineseZodiacUsingMatchCase

Operator Precedence

- +, -
- **
- not
- *, /, //, %
- +, -
- <, <=, >, >=
- ==, !=
- and
- or
- =, +=, -=, *=, /=, //=, %= (Assignment operator)

Operator Precedence and Associativity

The expression in the parentheses is evaluated first. (Parentheses can be nested, in which case the expression in the inner parentheses is executed first.) When evaluating an expression without parentheses, the operators are applied according to the precedence rule and the associativity rule.

If operators with the same precedence are next to each other, their associativity determines the order of evaluation. All binary operators except assignment operators are left-associative.

Operator Associativity

When two operators with the same precedence are evaluated, the *associativity* of the operators determines the order of evaluation. All binary operators except assignment operators are *left-associative*.

a – b + c – d is equivalent to ((a – b) + c) – d

Assignment operators are *right-associative*. Therefore, the expression

a = b = c = 5 is equivalent to a = (b = (c = 5))