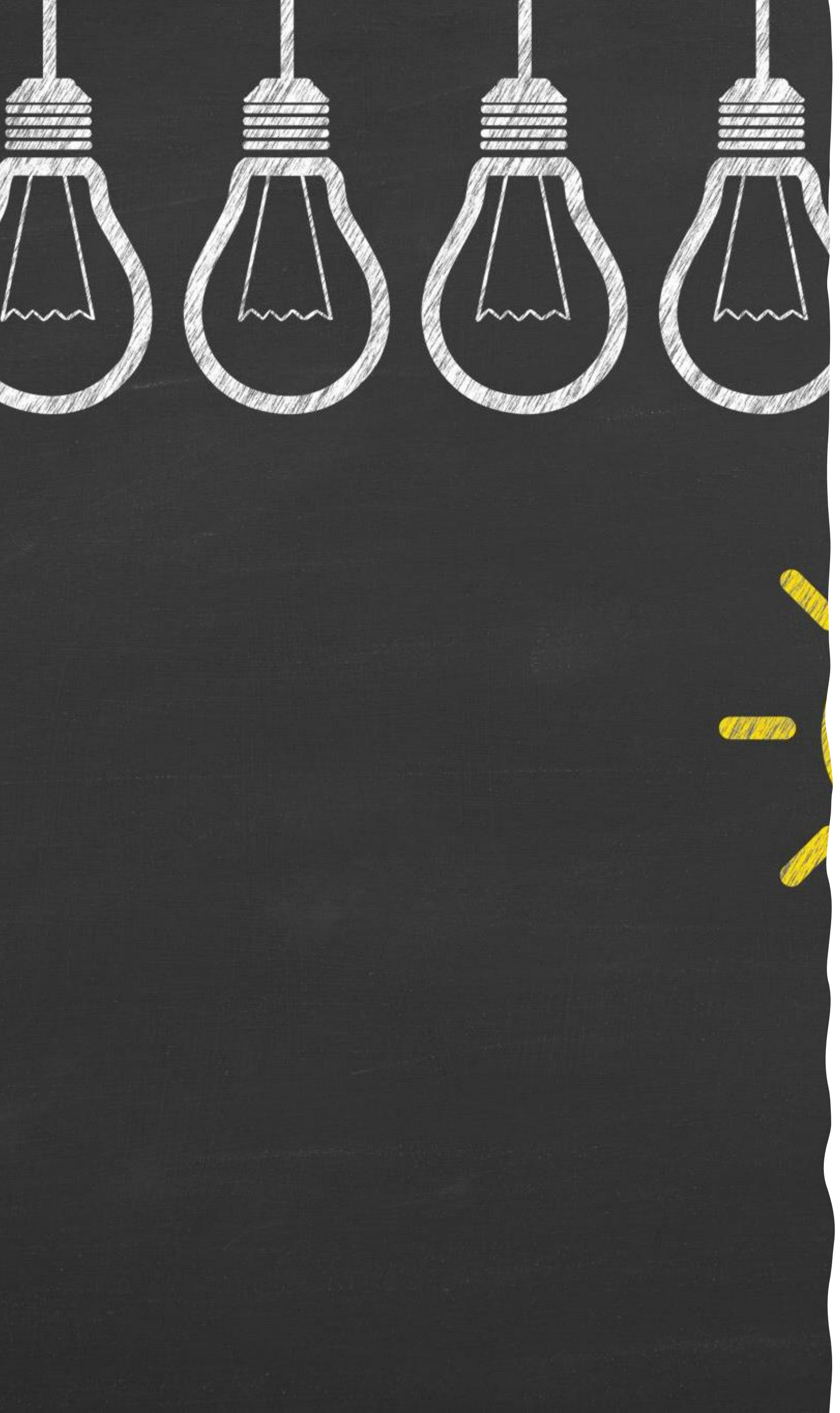


CMSC 105 Elementary Programming

Acknowledgement: These slides are adapted from slides provided with “Introduction to Programming Using Python, Liang (Pearson 2013)” and slides shared by Dr. Jory Denny

Outline

Control Flow & Boolean Values

Relational Operators

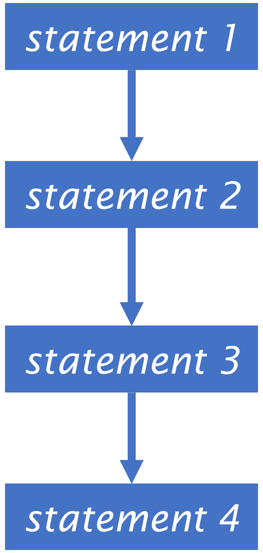
if-else Statements

Multiple Conditions

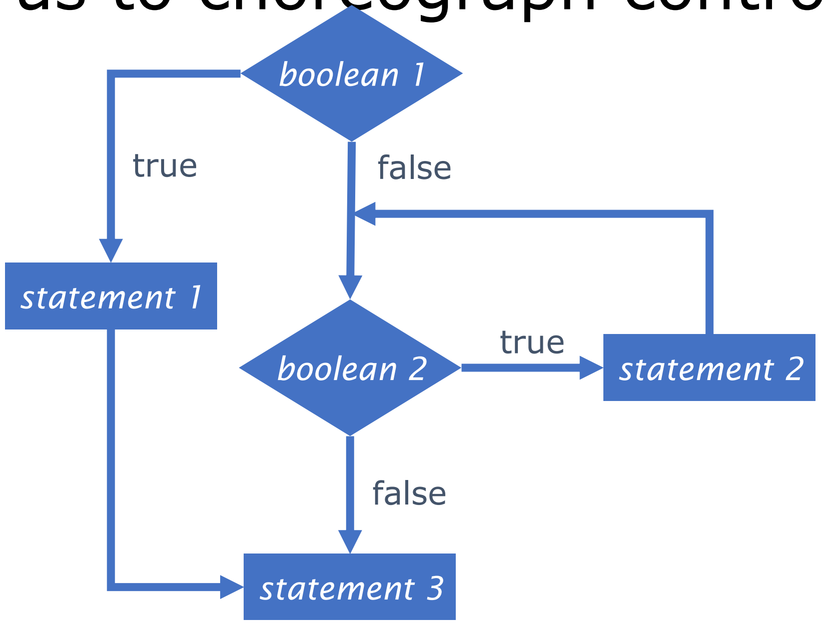
Control Flow

Notation
- Block – statement of code
- Diamond – conditional
- Open circle – start/end of algorithm

- Control flow.
  - Sequence of statements that are actually executed in a program.
  - Conditionals and loops: enable us to choreograph control flow.



straight-line control flow



control flow with conditionals and loops

Boolean values

True and **False** are the only Boolean values.

The type is shown as **<class 'bool'>**.

```
>>> type(True)
```

Boolean values

True and **False** are the only Boolean values.

The type is shown as **<class 'bool'>**.

```
>>> type(True)
<class 'bool'>
```

Boolean values

True and **False** are the only Boolean values.

The type is shown as **<class 'bool'>**.

```
>>> type(True)
<class 'bool'>
>>> type(False)
```

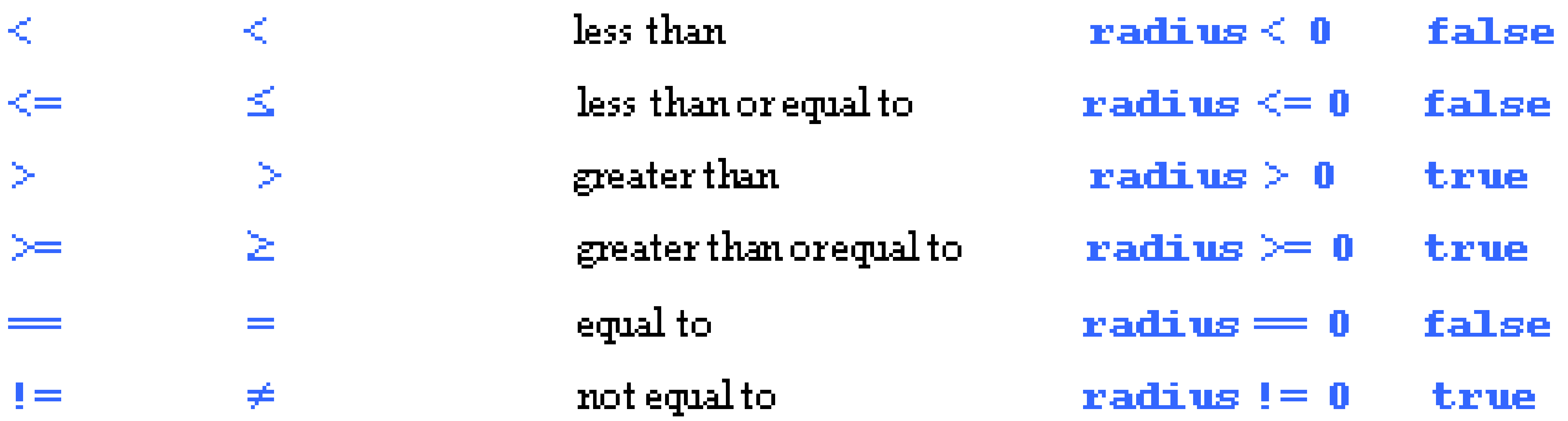
Boolean values

True and **False** are the only Boolean values.

The type is shown as **<class 'bool'>**.

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

Relational Operators



Python Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
<	<	less than	radius < 0	false
<=	≤	less than or equal to	radius <= 0	false
>	>	greater than	radius > 0	true
>=	≥	greater than or equal to	radius >= 0	true
==	=	equal to	radius == 0	false
!=	≠	not equal to	radius != 0	true

Logical Operators

Python Operator	**Math Symbol**	**Description**
`not`	$\neg$	logical negation
`and`	$\wedge$	logical conjunction
`or`	$\vee$	logical disjunction

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 25
>>> y = 30
```

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 25
>>> y = 30
>>> x < y
```

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 5
>>> y = 10
>>> x < y
True
```

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 5
>>> y = 10
>>> x < y
True
>>> x > y
```

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 5
>>> y = 10
>>> x < y
True
>>> x > y
False
```

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 5
>>> y = 10
>>> x < y
True
>>> x > y
False
>>> x == y
```

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 5
>>> y = 10
>>> x < y
True
>>> x > y
False
>>> x == y
```

Checking if x is equal to y! Symbol == checks for equality

Relational Expression

A Boolean value is the result of a *relational expression*:

```
>>> x = 5
>>> y = 10
>>> x < y
True
>>> x > y
False
>>> x == y
False
```

Example

- Check if a number is
    - Divisible by 2 and 3
    - Divisible by 2 or 3

```
Line 1    x = eval(input("Enter a number: "))

Line 2    divBy2 = x % 2 == 0
Line 3    divBy3 = x % 3 == 0

Line 4    print(x, "divisible by 2 and 3: ", divBy2 and divBy3)
Line 5    print(x, "divisible by 2  or 3: ", divBy2  or divBy3)
```

Exercise

- Let a user enter a year, and output whether or not it is a leap year. A year is a leap year if it is
  - Divisible by 4 but not by 100
  - OR
  - Divisible by 400
- Do not use any `if` statements, only Boolean expressions

Operator Precedence

1. +, - (unary +/-)
2. ** (exponentiation)
3. not
4. *, /, //, % (multiplication/division)
5. +, - (addition/subtraction)
6. <, <=, >, >= (comparison)
7. ==, != (equality)
8. and
9. or
10. =, +=, -=, *=, /=, //=, %= (Assignment operator)

Operator Precedence and Associativity

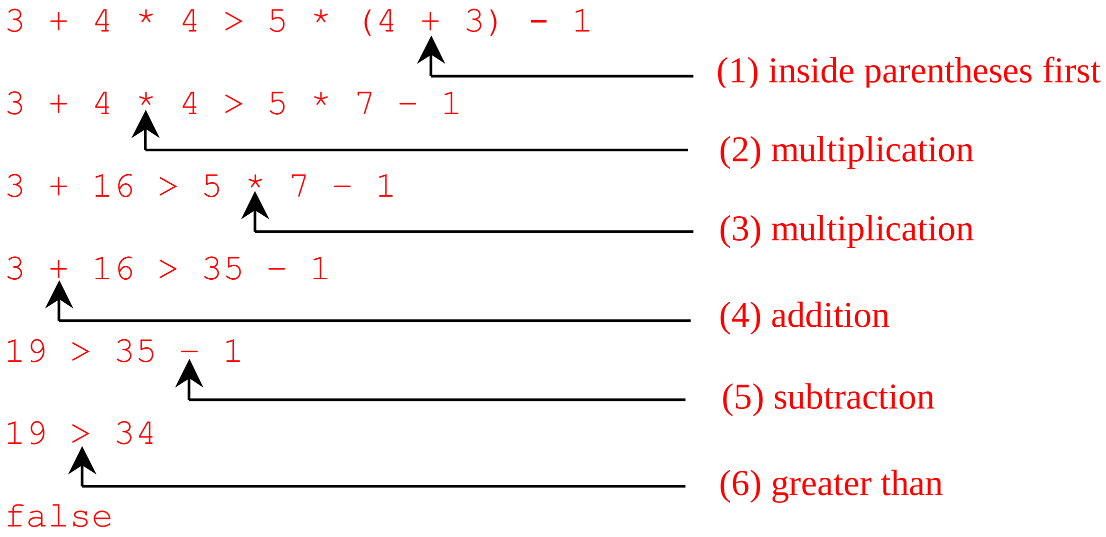
- The expression in the parentheses is evaluated first.

- 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.

Example

- Applying the operator precedence and associativity rule, the expression
  `3 + 4 * 4 > 5 * (4 + 3) - 1`
  is evaluated as follows:



```
3 + 4 * 4 > 5 * (4 + 3) - 1
3 + 4 * 4 > 5 * 7 – 1
3 + 16 > 5 * 7 – 1
3 + 16 > 35 – 1
19 > 35 – 1
19 > 34
false
```

(1) inside parentheses first

(2) multiplication

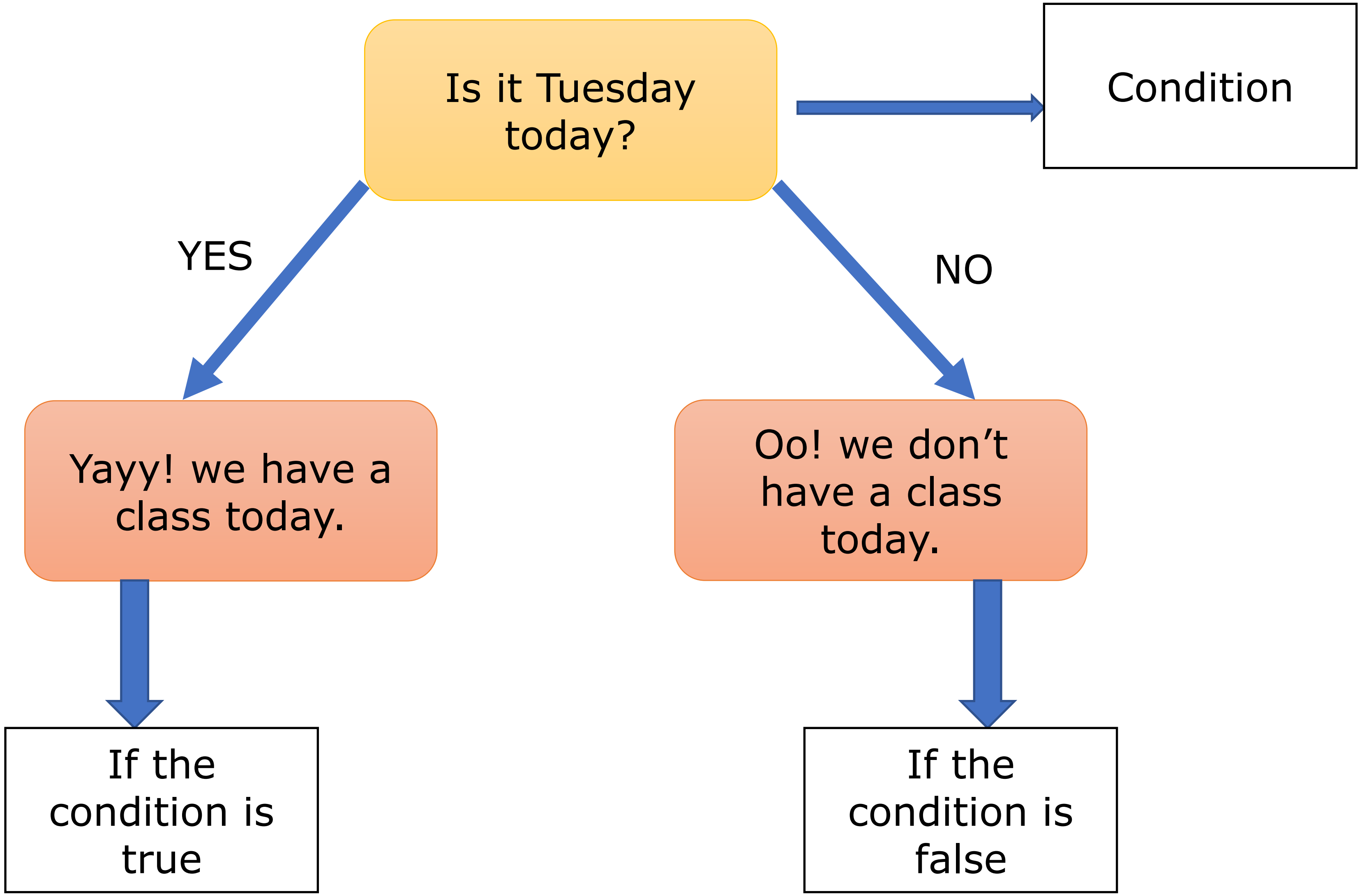
(3) multiplication

(4) addition

(5) subtraction

(6) greater than

if-else Statement



Is it Tuesday today?

Condition

YES

NO

Yayy! we have a class today.

Oo! we don’t have a class today.

If the condition is true

If the condition is false

The **if** statement

This statement is also known as a *conditional* statement. It has two basic forms:

```
if condition:
   consequent
```

The **if** statement

This statement is also known as a *conditional* statement. It has two basic forms:

```
if condition:
  consequent
```

```
if condition:
  consequent
else:
  alternate
```

The **if** statement

This statement is also known as a *conditional* statement. It has two basic forms:

```
if condition:
   consequent
```

```
if condition:
   consequent
else:
   alternate
```

The consequent and alternate are groups of one or more statements.

The **if** statement

This statement is also known as a *conditional* statement. It has two basic forms:

```
if condition:
    consequent
```

They must be indented!

```
if condition:
    consequent
else:
    alternate
```

The **if** statement

This statement is also known as a *conditional* statement. It has two basic forms:

```
if condition:
⇨ consequent
```

They must be indented!

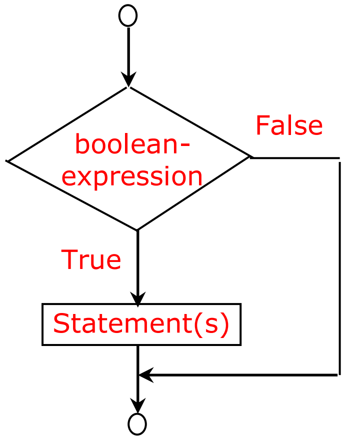
The standard in Python is to use 4 spaces.

```
if condition:
⇨ consequent
else:
⇨ alternate
```

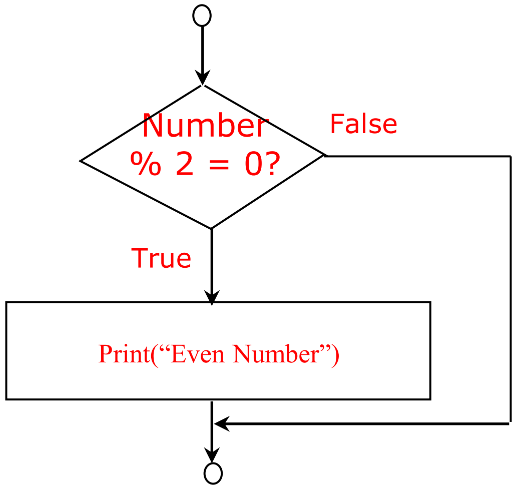
One-way if Statements

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

```
if number % 2 == 0:
    print(“Even Number”)
```



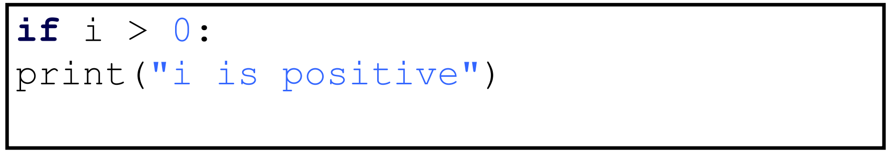
(a)



(b)

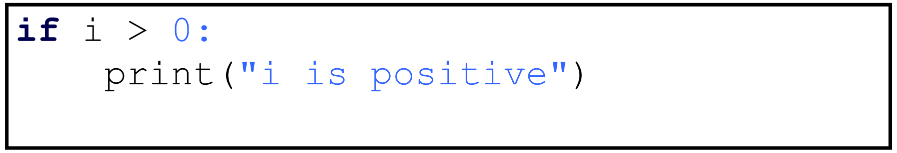
Note

- Indentation matters



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

(a) Wrong

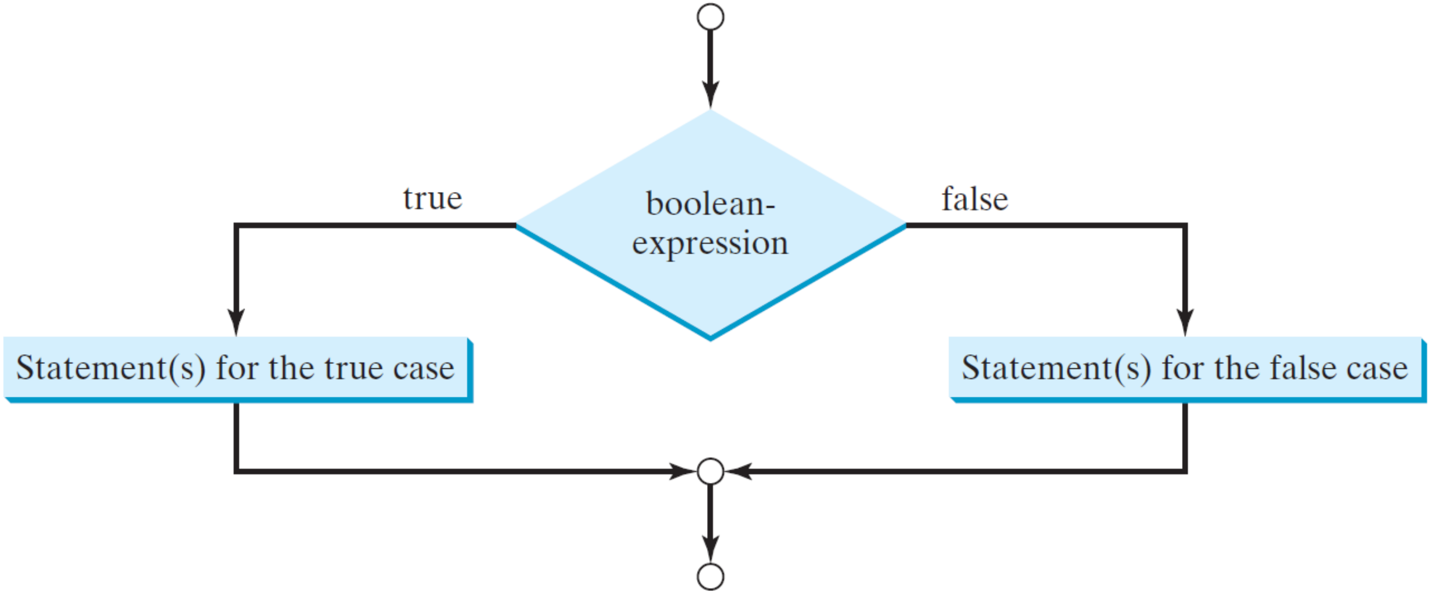


```python
if i > 0:
    print("i is positive")
```

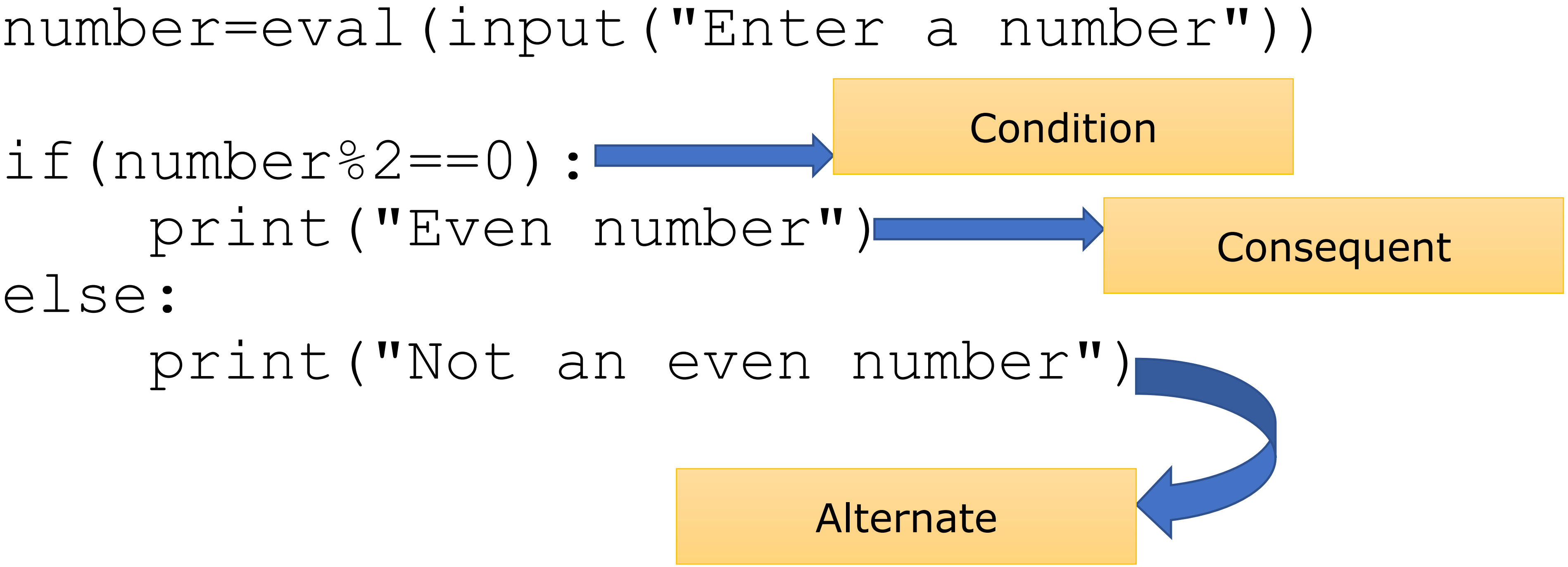
(b) Correct

The Two-way if Statement

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


true
boolean-expression
false
Statement(s) for the true case
Statement(s) for the false case

if-else Example



```
number=eval(input("Enter a number"))

if(number%2==0):
    print("Even number")
else:
    print("Not an even number")
```

Condition

Consequent

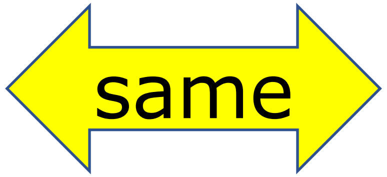
Alternate

Exercise

- Write a program that reads the total exam score and displays the grades. Here are the conditions:
  - If score is greater than or equal to 90, then display grade ‘A’
  - Display grade ‘B’, otherwise

Multiple Conditions

```
if condition1:
  consequent1
else:
  if condition2:
    consequent2
  else:
    if condition3:
      consequent3
    else:
      alternate3
```

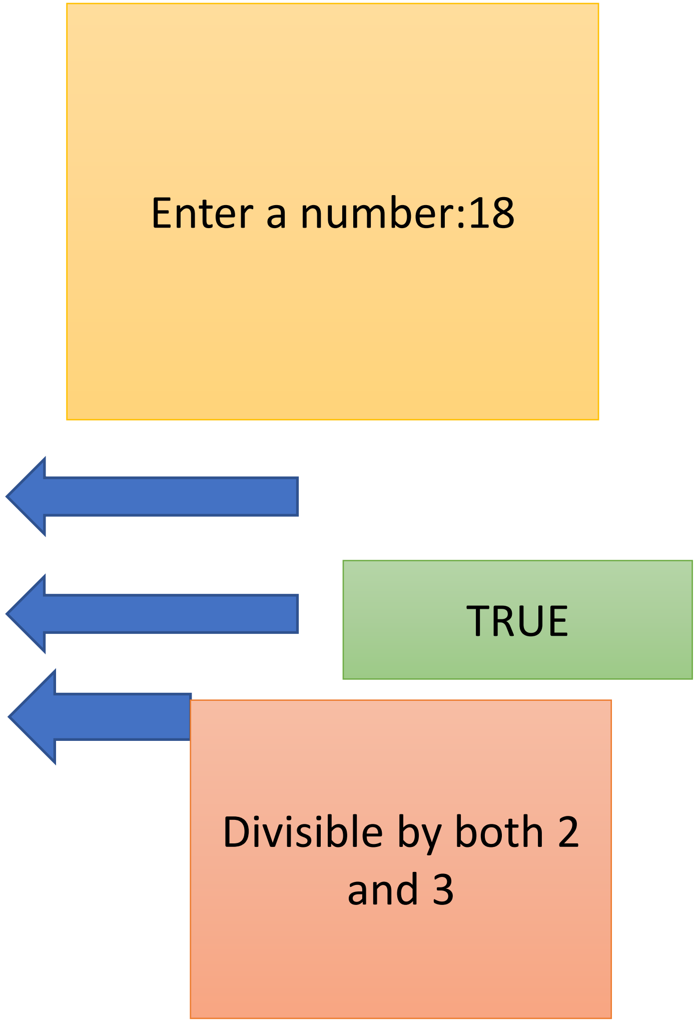

same

```
if condition1:
  consequent1
elif condition2:
  consequent2
elif condition3:
  consequent3
else:
  alternate3
```

Example

- Check if a number is
  - Divisible by 2 and 3
  - Divisible by 2 or 3

```
Line 1      number = eval(input("Enter a number:"))
Line 2      if(number%2 == 0 and number%3 == 0):
Line3           print("Divisible by both 2 and 3.")
Line 4      elif(number%2 == 0 or number%3 == 0):
Line 5          print("Divisible by 2 or 3.")
```



Enter a number:18

TRUE

Divisible by both 2 and 3

Example

- Check if a number is
  - Divisible by 2 and 3
  - Divisible by 2 or 3

Enter a number:22

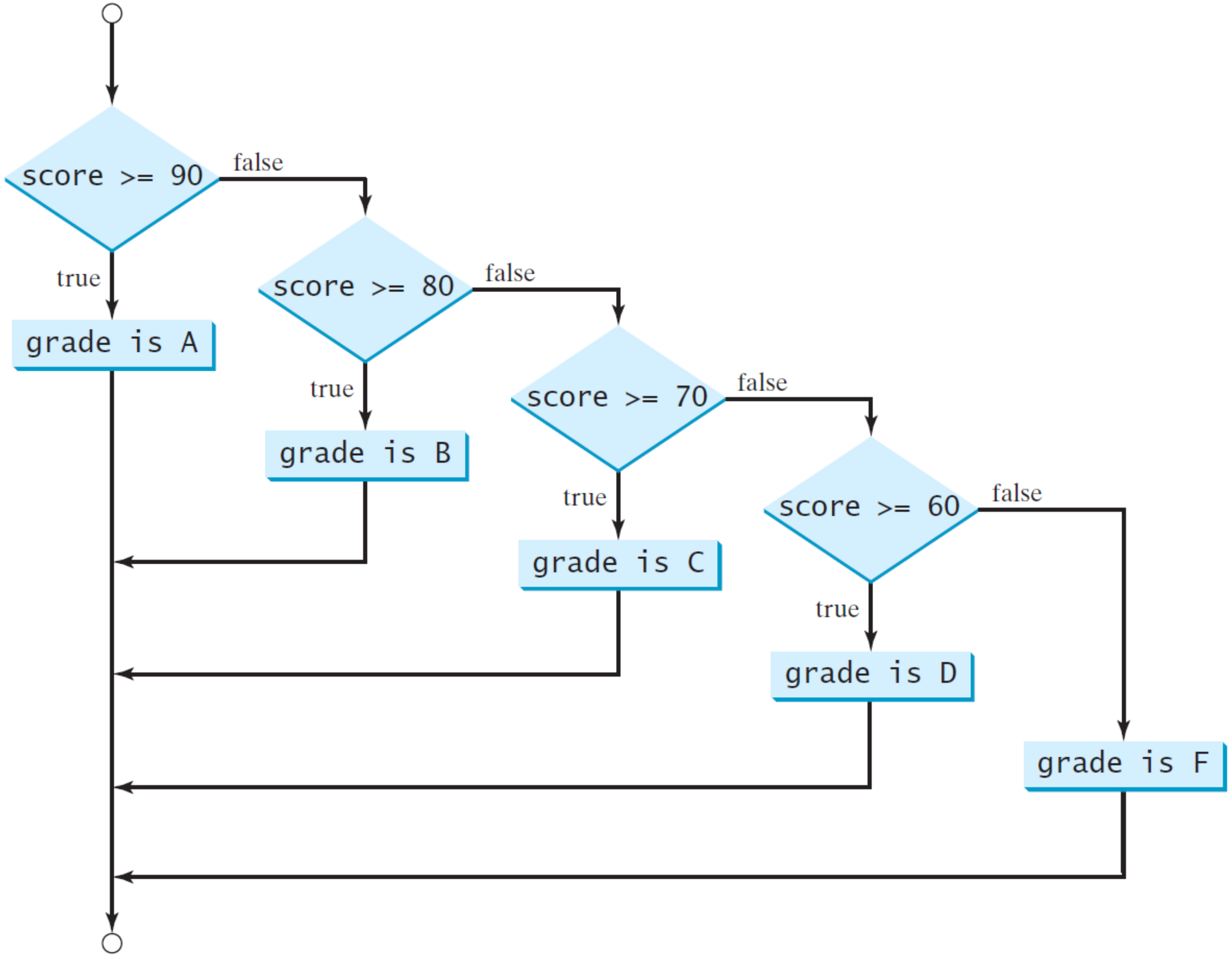
```
number = eval(input("Enter a number:"))
if(number%2 == 0 and number%3 == 0):
        print("Divisible by both 2 and 3.")
elif(number%2 == 0 or number%3 == 0):
        print("Divisible by 2 or 3.")
```

FALSE

TRUE

Divisible by 2 or 3

Multi-Way if-else Statements



Trace if-else statement

Suppose score is 72.3

```
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'
```

Condition is false

Trace if-else statement

Suppose score is 72.3

```
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'
```

Condition is false

Trace if-else statement

Suppose score is 72.3

```
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'
```

Condition is true

Trace if-else statement

Suppose score is 72.3

```
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'
```

Output “C”

Trace if-else statement

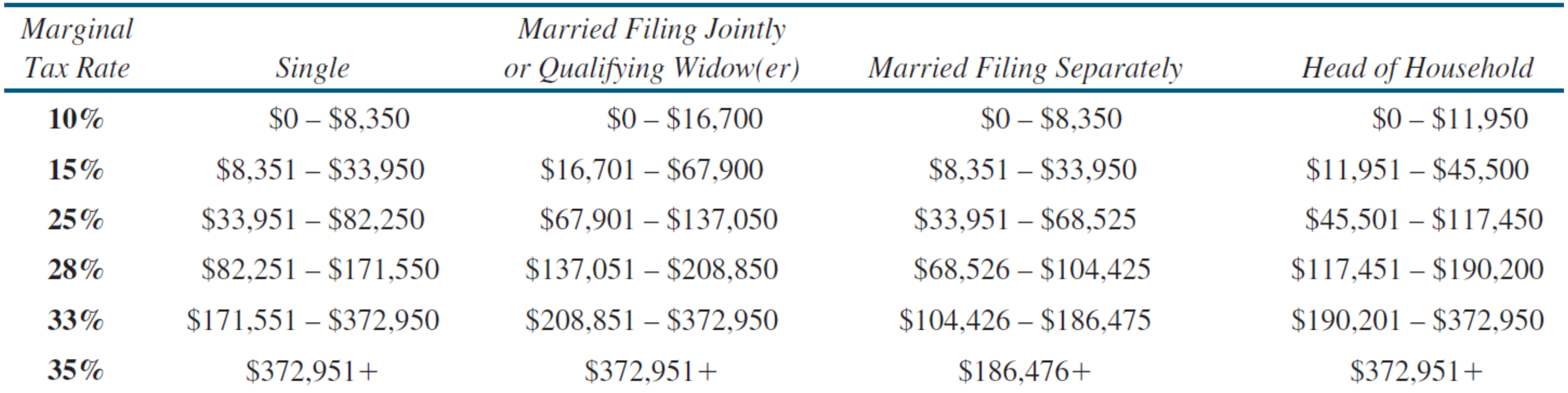
Suppose score is 72.3

```
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'
```

Exit the block

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.



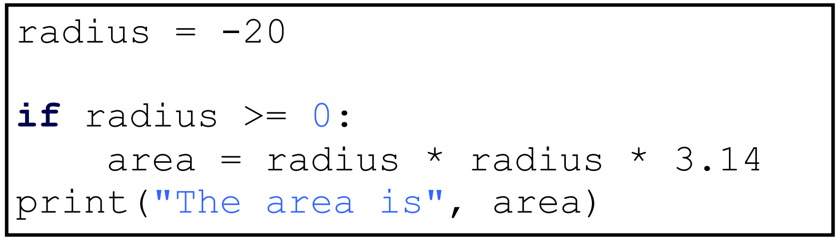
Marginal Tax Rate	*Single*	*Married Filing Jointly or Qualifying Widow(er)*	*Married Filing Separately*	*Head of Household*
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351 – $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $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
```

Common Errors

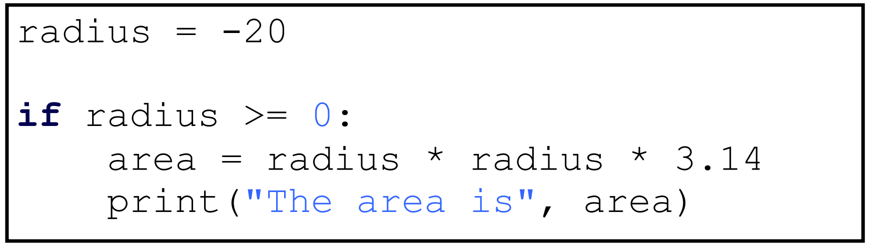
- Most common errors in selection statements are caused by incorrect indentation. Consider the following code in (a) and (b).



```
radius = -20

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

(a) Wrong



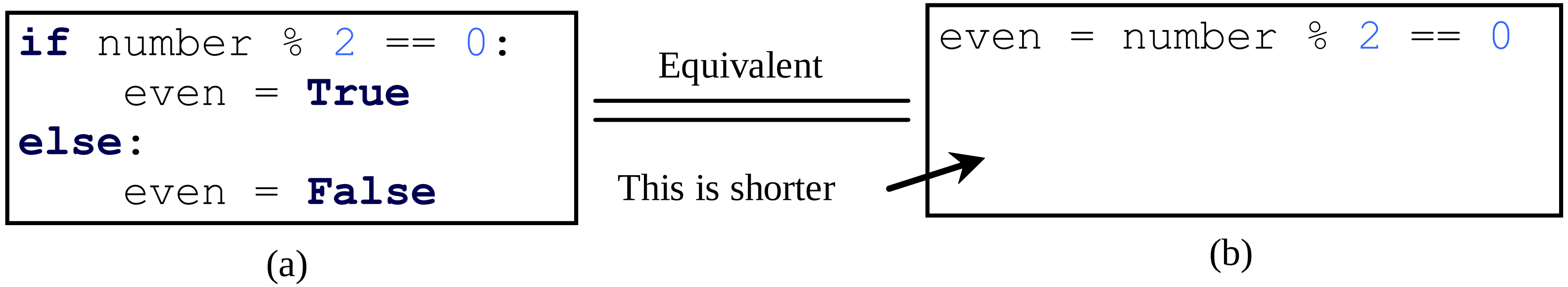
```
radius = -20

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

(b) Correct

TIP

- Use Boolean expressions when you can



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

(a)

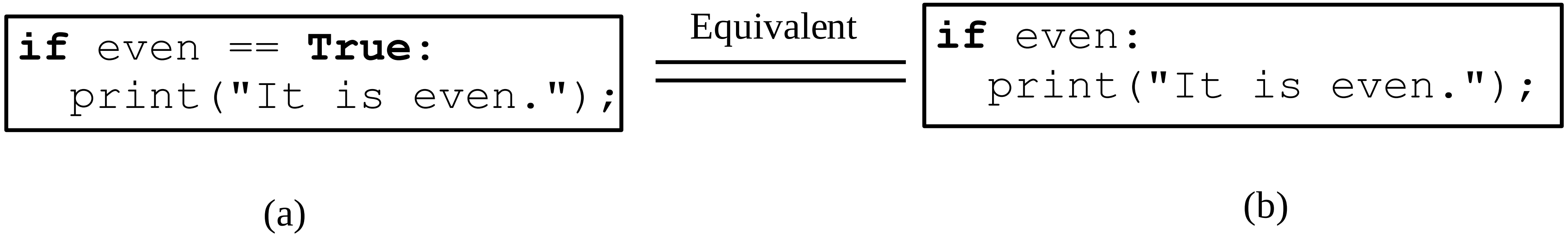
Equivalent

This is shorter

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

(b)

CAUTION



```
if even == True:
  print("It is even.");
```

Equivalent

```
if even:
  print("It is even.");
```

(a)

(b)

Exercise

- Write a program that reads a number and checks the following:
  - If the number is divisible by 2, display “Divisible by 2”

  - If the number is divisible by 5, display “Divisible by 5”

Exercise

- Extend the grading exercise by checking additional conditions:
    - If range of score is between 90 to 100, grade is ‘A’
    - If range of score is between 80 to 90, grade is ‘B’
    - If range of score is between 70 to 80, grade is ‘C’
    - Display, grade ‘D’ otherwise

Hint: Use **and, or** in if-else statements.

Thank you! Questions?