Chapter 2 Elementary Programming

Objectives

- To write programs that perform simple computations
- To obtain input from a program's user by using the **input** function
- To use identifiers to name variables
- To assign data to variables
- To use the operators +, -, *, /, //, %, and **
- To write and evaluate numeric expressions
- To use augmented assignment operators
- To perform numeric type conversion and rounding with the **int** and **round** functions

Trace a Program Execution

```
# Assign a radius
radius = 20                # radius is now 20

# Compute area
area = radius * radius * 3.14159

# Display results
print("The area for the circle of radius " , radius,
     " is" , area)
```

Reading Input from the Console

1. Use the input function

```
variable = input("Enter data: ")
```

2. Use the eval function

```
var = eval(variable)
```

Identifiers

- An identifier is a sequence of characters that consists of letters, digits, and underscores (_).
- An identifier must start with a letter or an underscore. It cannot start with a digit.
- An identifier cannot be a reserved word. (See Appendix A, "Python Keywords," for a list of reserved words.) Reserved words have special meanings in Python, which we will discuss later.
- An identifier can be of any length.

Expression

```
x = 1                 # Assign 1 to variable x
radius = 1.0          # Assign 1.0 to variable radius

# Assign the value of the expression to x
x = 5 * (3 / 2) + 3 * 2

y = 13
x = y + 1       # Assign the addition of y and 1 to x
area = radius * radius * 3.14159 # Compute area
```

Assignment Statements

```
x = 1              # assign 1 to x
x = x + 1          # assign (current
                   # value of x) + 1
                   # to x

i = j = k = 1
```

Simultaneous Assignment

```
var1, var2, ..., varn = exp1, exp2, ..., expn
```

```
x, y = y, x # Swap x with y
```

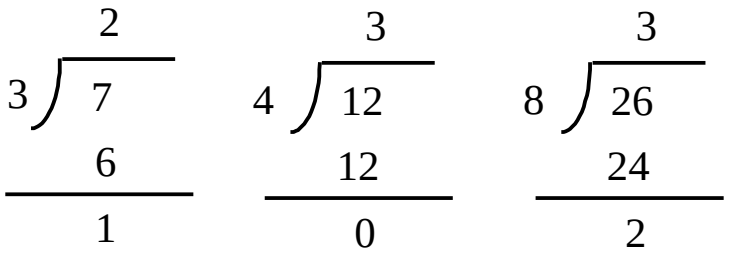
Numerical Data Types

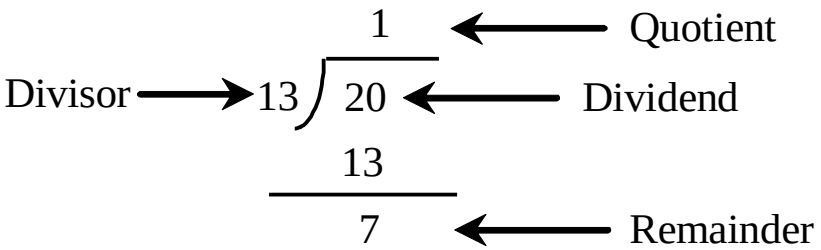
- integer: e.g., 3, 4
- float: e.g., 3.0, 4.0

Numeric Operators

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 – 0.1	33.9
*	Multiplication	300 * 30	9000
/	Float Division	1 / 2	0.5
//	Integer Division	1 // 2	0
**	Exponentiation	4 ** 0.5	2.0
%	Remainder	20 % 3	2

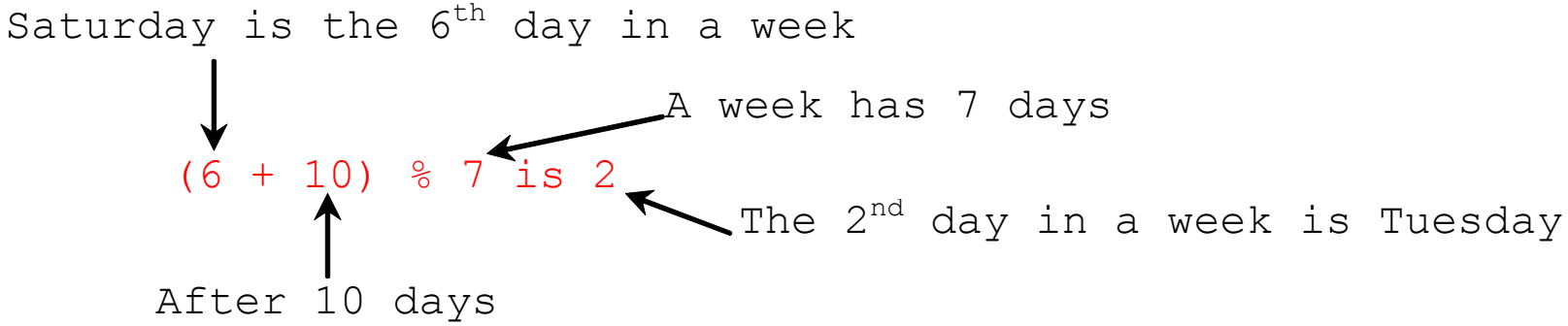
The % Operator





Remainder Operator

Remainder is very useful in programming. For example, an even number % 2 is always 0 and an odd number % 2 is always 1. So you can use this property to determine whether a number is even or odd. Suppose today is Saturday and you and your friends are going to meet in 10 days. What day is in 10 days? You can find that day is Tuesday using the following expression:



Problem: Displaying Time

Write a program that obtains hours and minutes from seconds.

Arithmetic Expressions

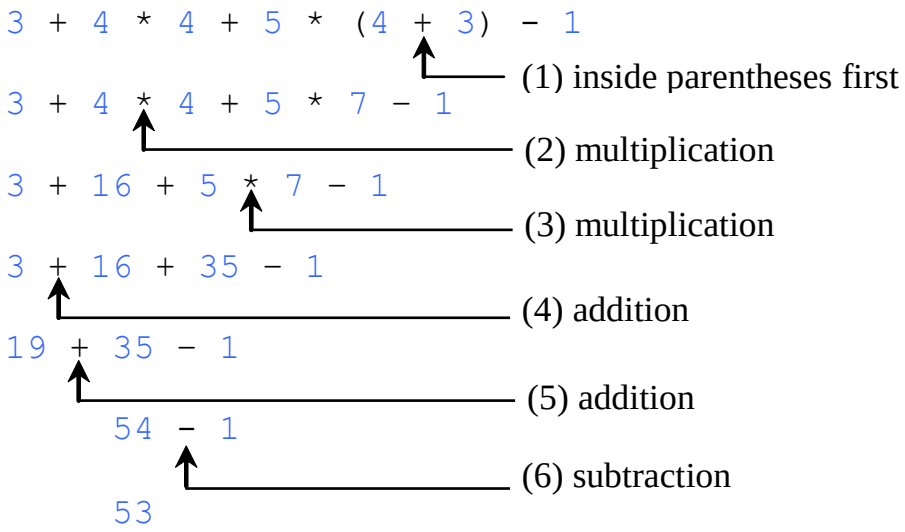
$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

is translated to

(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)

How to Evaluate an Expression

Although Python has its own way to evaluate an expression behind the scene, the result of a Python expression and its corresponding arithmetic expression are the same. Therefore, you can safely apply the arithmetic rule for evaluating a Python expression.



Augmented Assignment Operators

Operator	*Example*	*Equivalent*
`+=`	`i += 8`	`i = i + 8`
`-=`	`f -= 8.0`	`f = f - 8.0`
`*=`	`i *= 8`	`i = i * 8`
`/=`	`i /= 8`	`i = i / 8`
`%=`	`i %= 8`	`i = i % 8`

Type Conversion and Rounding

datatype(value)

i.e., int(4.5) => 4
float(4) => 4.0
str(4) => “4”

round(4.629) => 5
round(4.629,2) => 4.63

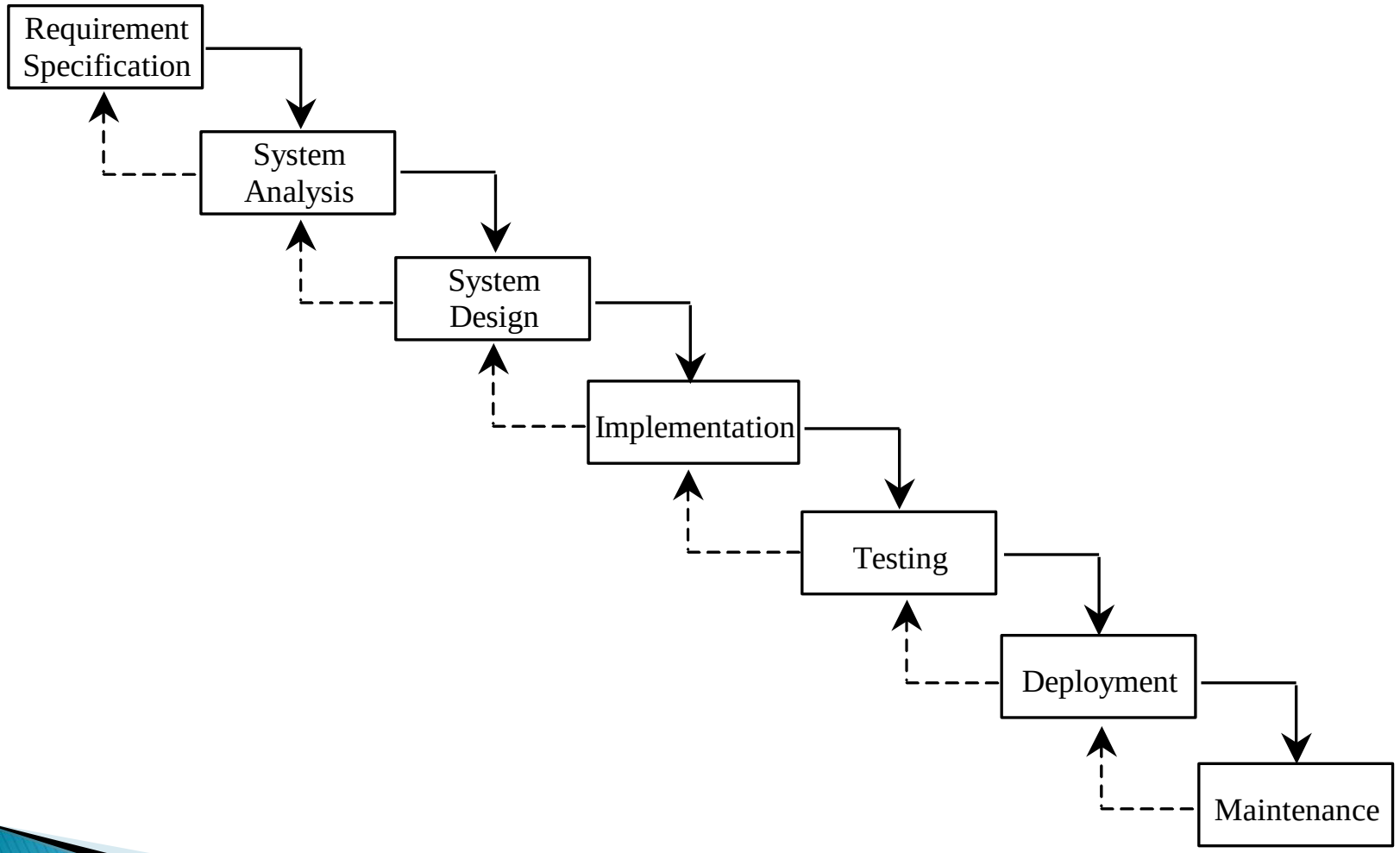
Problem: Displaying Current Time

Write a program that displays current time in GMT in the format hour:minute:second such as 1:45:19.

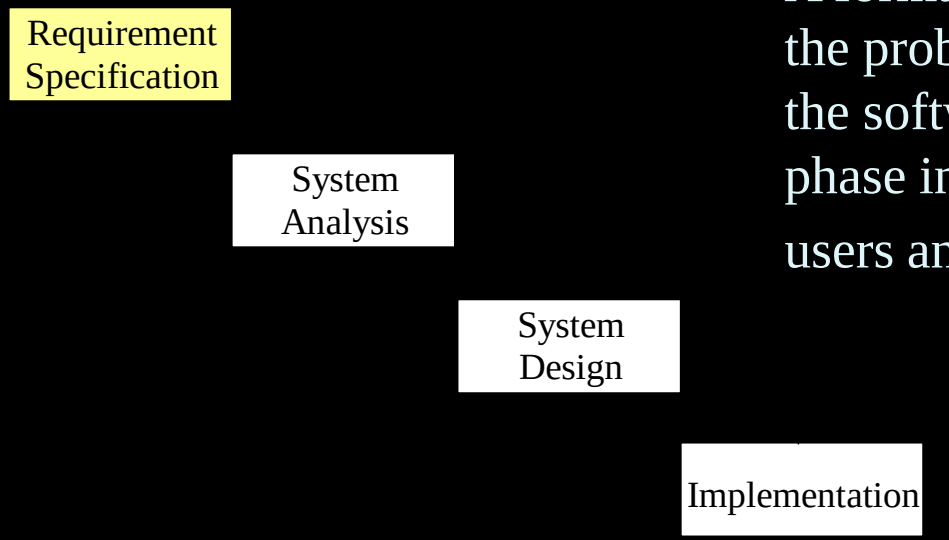
The time.time() function returns the current time in seconds with millisecond precision since the midnight, January 1, 1970 GMT. (1970 was the year when the Unix operating system was formally introduced.) You can use this function to obtain the current time, and then compute the current second, minute, and hour as follows.

ShowCurrentTime

Software Development Process



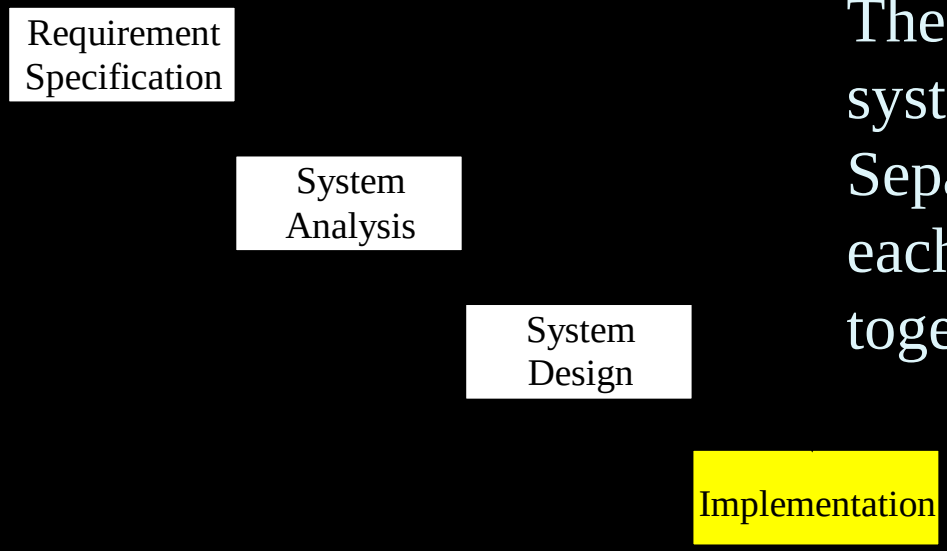
Requirement Specification



A formal process that seeks to understand the problem and document in detail what the software system needs to do. This phase involves close interaction between users and designers.

Most of the examples in this book are simple, and their requirements are clearly stated. In the real world, however, problems are not well defined. You need to study a problem carefully to identify its requirements.

Implementation



The process of translating the system design into programs. Separate programs are written for each component and put to work together.

This phase requires the use of a programming language like Python. The implementation involves coding, testing, and debugging.

Problem: Computing Loan Payments

This program lets the user enter the interest rate, number of years, and loan amount, and computes the monthly payment.

$$monthlyPayment = \frac{loanAmount \times monthlyInterestRate}{1 - \frac{1}{(1 + monthlyInterestRate)^{numberOfYears \times 12}}}$$

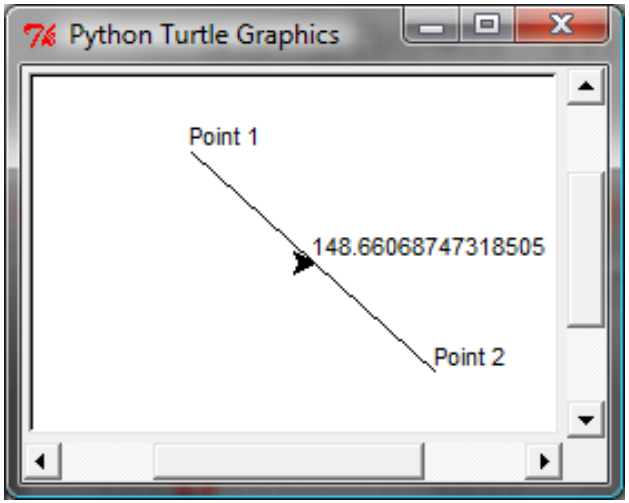
Case Study: Computing Distances

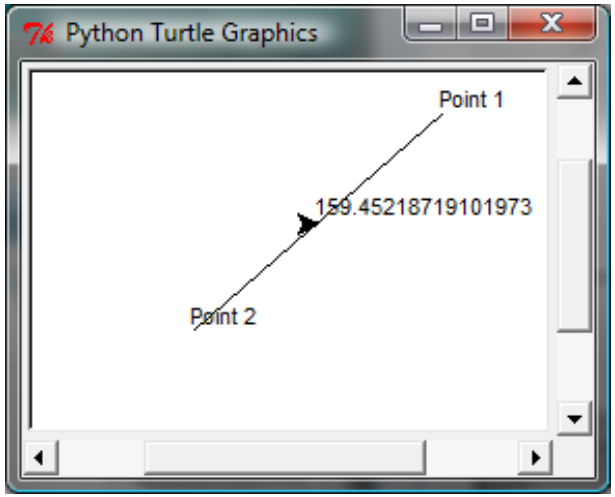
This program prompts the user to enter two points, computes their distance, and displays the points.

ComputeDistance

Case Study: Computing Distances

This program prompts the user to enter two points, computes their distance, and displays the points and their distances in graphics.





ComputeDistanceGraphics