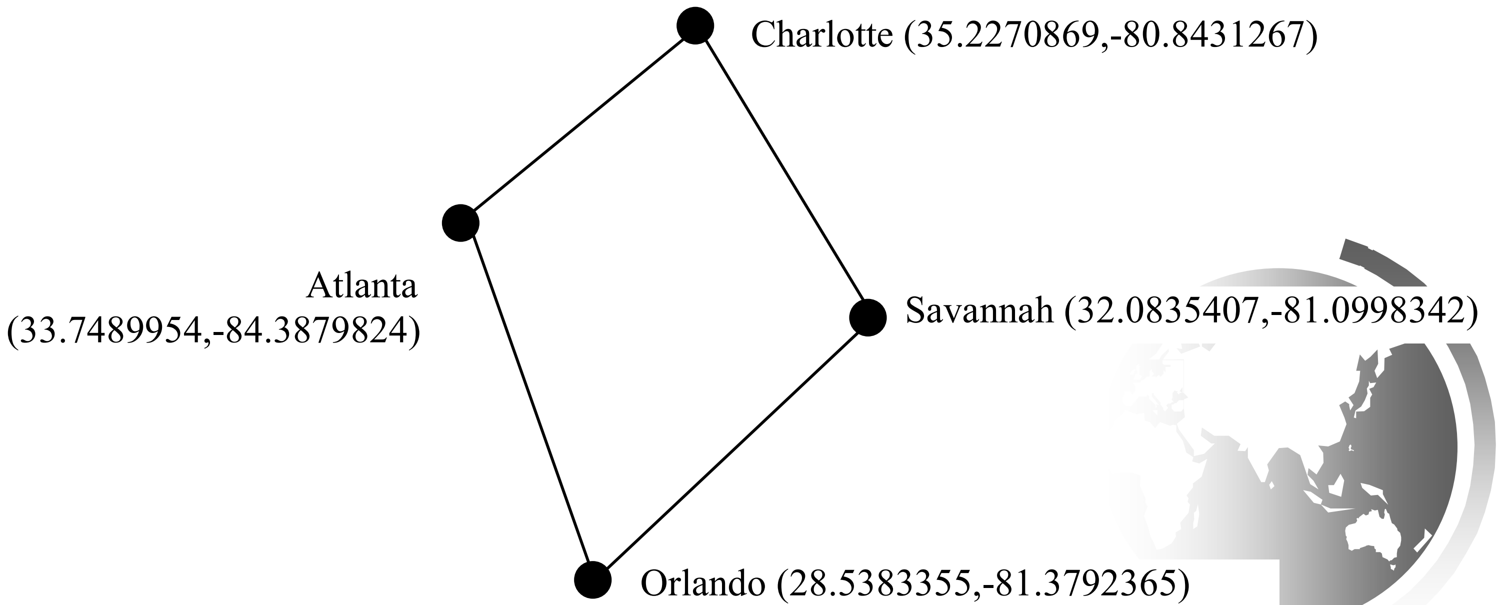
Chapter 4
Mathematical Functions, Strings, and Objects

Motivations

Suppose you need to estimate the area enclosed by four cities, given the GPS locations (latitude and longitude) of these cities, as shown in the following diagram. How would you write a program to solve this problem? You will be able to write such a program after completing this chapter.


Charlotte (35.2270869,-80.8431267)
Atlanta (33.7489954,-84.3879824)
Savannah (32.0835407,-81.0998342)
Orlando (28.5383355,-81.3792365)

Objectives

- To solve mathematics problems by using the functions in the **math** module (§4.2).
- To represent and process strings and characters (§§4.3-4.4).
- To encode characters using ASCII and Unicode (§§4.3.1-4.3.2).
- To use the **ord** function to obtain a numerical code for a character and the **chr** function to convert a numerical code to a character (§4.3.3).
- To represent special characters using the escape sequence (§4.3.4).
- To invoke the **print** function with the **end** argument (§4.3.5).
- To convert numbers to a string using the **str** function (§4.3.6).
- To use the + operator to concatenate strings (§4.3.7).
- To read strings from the keyboard (§4.3.8).
- To solve the lottery problem using strings (**§**4.4).
- To introduce objects and methods (**§**4.5).
- To introduce the methods in the **str** class (**§**4.6).
- To program using characters and strings (**GuessBirthday**) (§4.7.1).
- To convert a hexadecimal character to a decimal value (**HexDigit2Dec**) (§4.7.2).
- To format numbers and strings using the **format** function (**§**4.8).

Python Built-in Functions

Function	**Description**	**Example**
`abs(x)`	Returns the absolute value for x.	`abs(-2)` is 2
`max(x1, x2, ...)`	Returns the largest among x1, x2, ...	`max(1, 5, 2)` is 5
`min(x1, x2, ...)`	Returns the smallest among x1, x2, ...	`min(1, 5, 2)` is 1
`pow(a, b)`	Returns a^b. Same as `a ** b`.	`pow(2, 3)` is 8
`round(x)`	Returns an integer nearest to x. If x is equally close to two integers, the even one is returned.	`round(5.4)` is 5 `round(5.5)` is 6 `round(4.5)` is 4
`round(x, n)`	Returns the float value rounded to n digits after the decimal point.	`round(5.466, 2)` is 5.47 `round(5.463, 2)` is 5.46

Built-in Functions

```
>>> max(2, 3, 4) # Returns a maximum number
4
>>> min(2, 3, 4) # Returns a minimu number
2
>>> round(4.51) # Rounds to its nearest integer
4
>>> round(4.4) # Rounds to its nearest integer
3
>>> abs(-3) # Returns the absolute value
3
>>> pow(2, 3) # Same as 2 ** 3
8
```

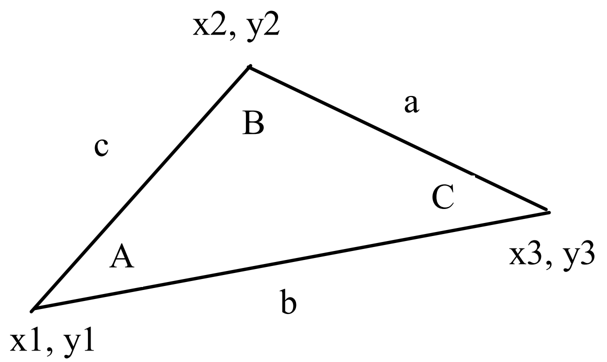
The math Functions

Function	Description	Example
fabs(x)	Returns the absolute value of the argument.	fabs(-2) is 2
ceil(x)	Rounds x up to its nearest integer and returns this integer.	ceil(2.1) is 3 ceil(-2.1) is -2
floor(x)	Rounds x down to its nearest integer and returns this integer.	floor(2.1) is 2 floor(-2.1) is -3
exp(x)	Returns the exponential function of x (e ** x).	exp(1) is 2.71828
log(x)	Returns the natural logarithm of x.	log(2.71828) is 1.0
log(x, base)	Returns the logarithm of x for the specified base.	log10(10, 10) is 1
sqrt(x)	Returns the square root of x.	sqrt(4.0) is 2
sin(x)	Returns the sine of x. x represents an angle in radians.	sin(3.14159 / 2) is 1 sin(3.14159) is 0
asin(x)	Returns the angle in radians for the inverse of sine.	asin(1.0) is 1.57 asin(0.5) is 0.523599
cos(x)	Returns the cosine of x. x represents an angle in radians.	cos(3.14159 / 2) is 0 cos(3.14159) is -1
acos(x)	Returns the angle in radians for the inverse of cosine.	acos(1.0) is 0 acos(0.5) is 1.0472
tan(x)	Returns the tangent of x. x represents an angle in radians.	tan(3.14159 / 4) is 1 tan(0.0) is 0
fmod(x, y)	Returns the remainder of x/y as double.	fmod(2.4, 1.3) is 1.1
degrees(x)	Converts angle x from radians to degrees	degrees(1.57) is 90
radians(x)	Converts angle x from degrees to radians	radians(90) is 1.57

MathFunctions

Problem: Compute Angles

Given three points of a triangle, you can compute the angles using the following formula:



```
A = acos((a * a - b * b - c * c) / (-2 * b * c))
B = acos((b * b - a * a - c * c) / (-2 * a * c))
C = acos((c * c - b * b - a * a) / (-2 * a * b))
```

ComputeAngles

Strings and Characters

A string is a sequence of characters. *String* literals can be enclosed in matching *single quotes* (') or *double quotes* ("). Python does not have a data type for characters. A single-character string represents a character.

```
letter = 'A' # Same as letter = "A"
numChar = '4' # Same as numChar = "4"
message = "Good morning"
# Same as message = 'Good morning'
```

NOTE

For consistency, this book uses double quotes for a string with more than one character and single quotes for a string with a single character or an empty string. This convention is consistent with other programming languages. So, it will be easy to convert a Python program to a program written in other languages.

Appendix B: ASCII Character Set

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.1 ASCII Character Set in the Decimal Index

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

ASCII Character Set, cont.

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.2 ASCII Character Set in the Hexadecimal Index

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	ff	cr	so	si
1	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	“	#	$	%	&	’	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	del

Functions ord and chr

```
>>> ch = 'a'
>>> ord(ch)
>>> 97
>>> chr(98)
>>> 'b'
```

Escape Sequences for Special Characters

Description	*Escape Sequence*	*Unicode*
Backspace	`\b`	`\u0008`
Tab	`\t`	`\u0009`
Linefeed	`\n`	`\u000A`
Carriage return	`\r`	`\u000D`
Backslash	`\\`	`\u005C`
Single Quote	`\'`	`\u0027`
Double Quote	`\"`	`\u0022`

Printing without the Newline

```
print(item, end = 'anyendingstring')

print("AAA", end = ' ')
print("BBB", end = '')
print("CCC", end = '***')
print("DDD", end = '***')
```

The str Function

The str function can be used to convert a number into a string. For example,

```
>>> s = str(4.4) # Convert a float to string
>>> s
'4.4'
>>> s = str(3) # Convert an integer to string
>>> s
'3'
>>>
```

The String Concatenation Operator

You can use the + operator add two numbers. The + operator can also be used to concatenate (combine) two strings. Here are some examples:

```
>>> message = "Welcome " + "to " + "Python"
>>> message
'Weclome to Python'
>>> chapterNo = 2
>>> s = "Chapter " + str(chapterNo)
>>> s
'Chapter 2'
>>>
```

Reading Strings from the Console

To read a string from the console, use the input function. For example, the following code reads three strings from the keyboard:

```
s1 = input("Enter a string: ")
s2 = input("Enter a string: ")
s3 = input("Enter a string: ")
print("s1 is " + s1)
print("s2 is " + s2)
print("s3 is " + s3)
```

Case Study: Minimum Number of Coins

This program lets the user enter the amount in decimal representing dollars and cents and output a report listing the monetary equivalent in single dollars, quarters, dimes, nickels, and pennies. Your program should report maximum number of dollars, then the maximum number of quarters, and so on, in this order.

ComputeChange

Introduction to Objects and Methods

In Python, all data—including numbers and strings—are actually objects.

An object is an entity. Each object has an id and a type. Objects of the same kind have the same type. You can use the **id** function and **type** function to get these information for an object.

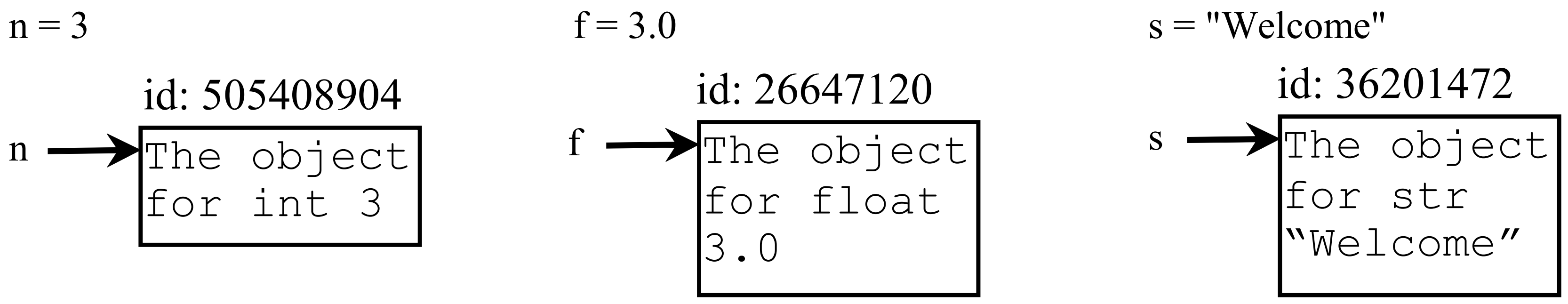
Object Types and Ids

The **id** and **type** functions are rarely used in programming, but they are good pedagogical tools for understanding objects.

```
>>> n = 3  # n is an integer
>>> id(n)
505408904
>>> type(n)
<class ’int’>
>>> f = 4.0  # f is a float
>>> id(f)
26647120
>>> type(f)
<class ’float’>
```

```
>>> s = "Welcome" # s is a string
>>> id(s)
36201472
>>> type(s)
<class ’str’>
```

OOP and str Objects



The **id** and **type** functions are rarely used in programming, but they are good pedagogical tools for understanding objects.

Object vs. Object reference Variable

For n = 4, we say n is an integer variable that holds value 4. Strictly speaking, n is a variable that references an int object for value 4. For simplicity, it is fine to say n is an int variable with value 4.

Methods

You can perform operations on an object. The operations are defined using functions. The functions for the objects are called *methods* in Python. Methods can only be invoked from a specific object. For example, the string type has the methods such as lower() and upper(), which returns a new string in lowercase and uppercase. Here are the examples to invoke these methods:

str Object Methods

```
>>> s = "Welcome"
>>> s1 = s.lower() # Invoke the lower method
>>> s1
'welcome'
>>> s2 = s.upper() # Invoke the upper method
>>> s2
'WELCOME'
```

Striping beginning and ending Whitespace Characters

Another useful string method is strip(), which can be used to strip the whitespace characters from the both ends of a string.

```
>>> s = "\t Welcome \n"
>>> s1 = s.strip() # Invoke the strip method
>>> s1
'Welcome'
```

Testing Strings

Method	Description
isalnum()	Returns True if all characters in this string are alphanumeric and there is at least one character.
isalpha()	Returns True if all characters in this string are alphabetic and there is at least one character.
isdigit()	Returns True if this string contains only number characters.
isidentifier()	Returns True if this string is a Python identifier.
islower()	Returns True if all characters in this string are lowercase letters and there is at least one character.
isupper()	Returns True if all characters in this string are uppercase letters and there is at least one character.
isspace()	Returns True if this string contains only whitespace characters.

Searching for Substrings

Method	Description
endswith(s1)	Returns True if the string ends with the substring s1.
startswith(s1)	Returns True if the string starts with the substring s1.
find(s1)	Returns the lowest index where s1 starts in this string, or -1 if s1 is not found in this string.
rfind(s1)	Returns the highest index where s1 starts in this string, or -1 if s1 is not found in this string.
count(subtring)	Returns the number of non-overlapping occurrences of this substring.

Converting Strings

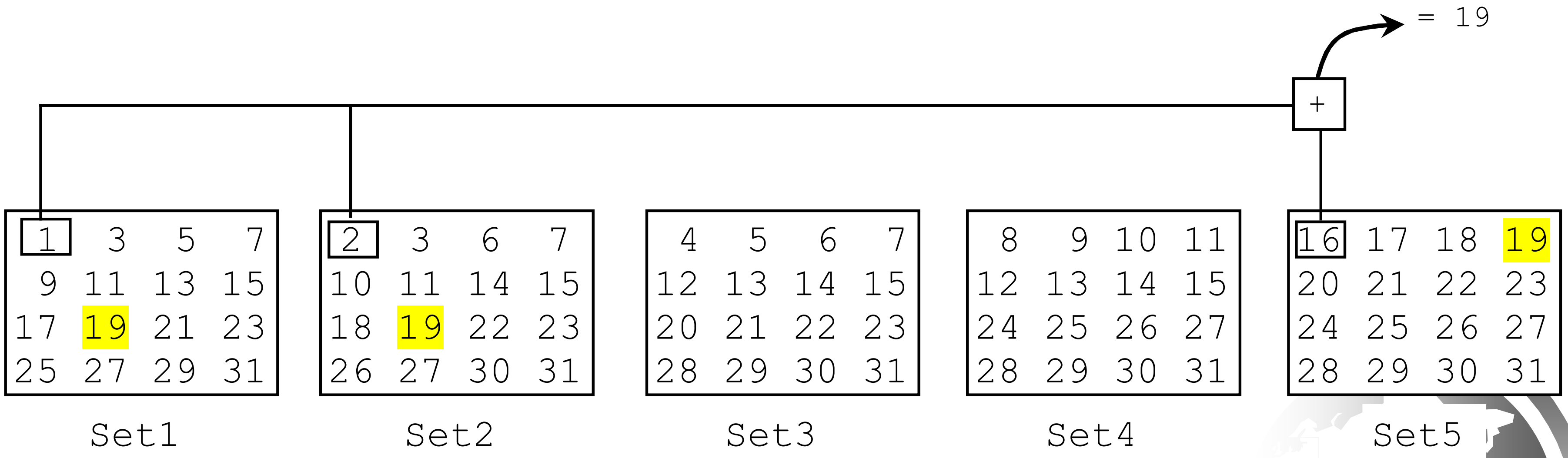
Method	Description
capitalize()	Returns a copy of this string with only the first character capitalized.
lower()	Returns a copy of this string with all letters converted to lowercase.
upper()	Returns a copy of this string with all letters converted to uppercase.
title()	Returns a copy of this string with the first letter capitalized in each word.
swapcase()	Returns a copy of this string in which lowercase letters are converted to upper and uppercase to lowercase.
replace(old, new)	Returns a new string that replaces all the occurrence of the old string with a n string.
replace(old, new, n)	Returns a new string that replaces up to n number of the occurrence of the old with a new string.

Striping Whitespace Characters

Method	Description
lstrip()	Returns a string with the leading whitespace characters removed.
rstrip()	Returns a string with the trailing whitespace characters removed.
strip()	Returns a string with the starting and trailing whitespace characters removed.

Problem: Guessing Birthday

The program can guess your birth date. Run to see how it works.



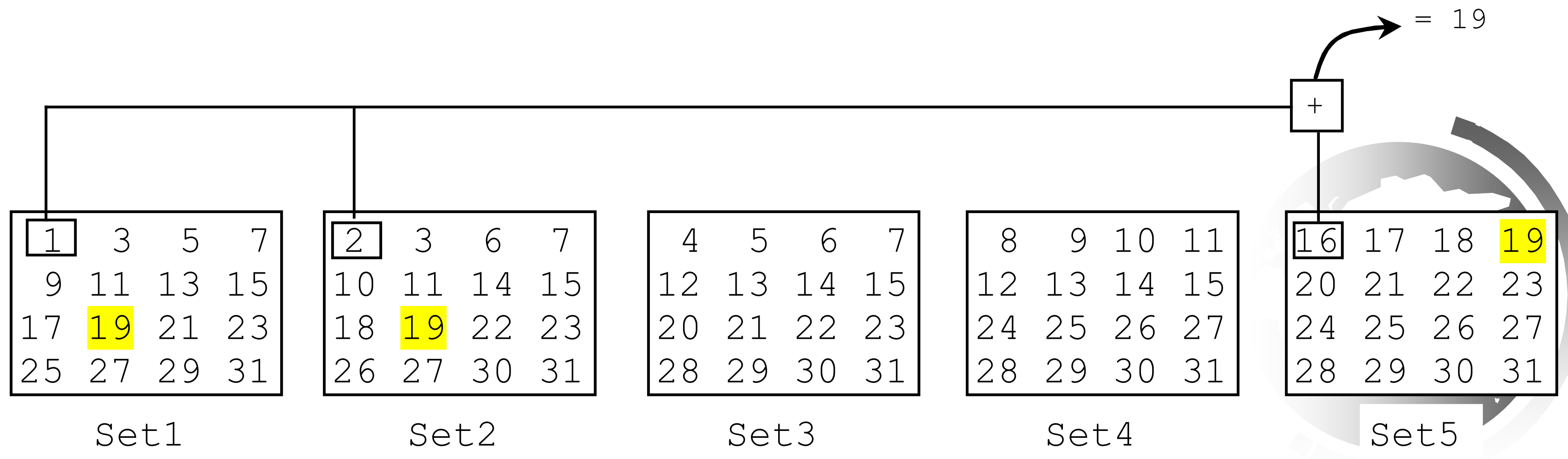
GuessBirthday

Mathematics Basis for the Game

19 is 10011 in binary. 7 is 111 in binary. 23 is 11101 in binary

```
                                        10000
   10000          00110                  1000
      10             10                   100
+      1       +      1            +        1
--------       --------            ----------
   10011          00111                 11101


     19              7                    23
```



Set1

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

Set2

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

Set3

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

Set4

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

Set5

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

Formatting Numbers and Strings

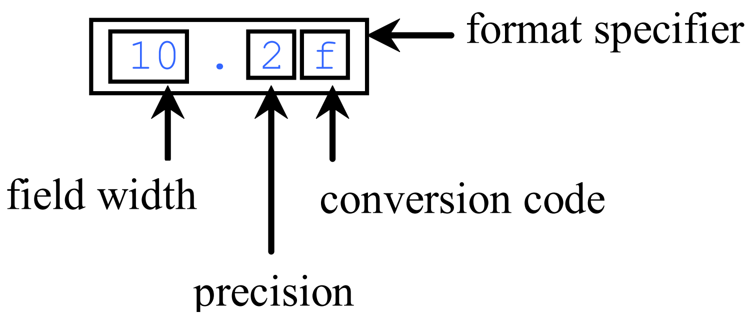
Often it is desirable to display numbers in certain format. For example, the following code computes the interest, given the amount and the annual interest rate.

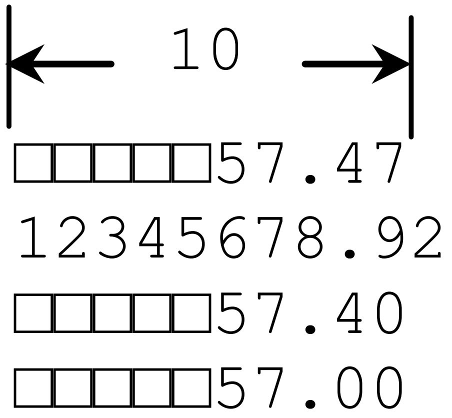
The format function formats a number or a string and returns a string.

format(item, format-specifier)

Formatting Floating-Point Numbers

```
print(format(57.467657, '10.2f'))
print(format(12345678.923, '10.2f'))
print(format(57.4, '10.2f'))
print(format(57, '10.2f'))
```



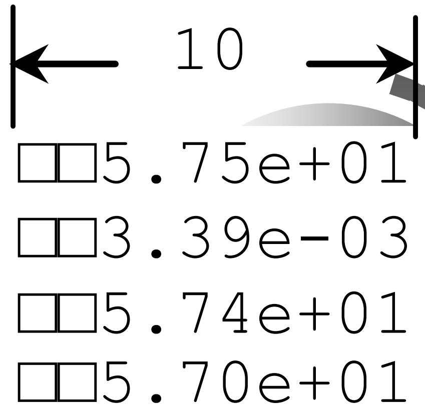


```
□□□□□57.47
12345678.92
□□□□□57.40
□□□□□57.00
```

Formatting in Scientific Notation

If you change the conversion code from <u>f</u> to <u>e</u>, the number will be formatted in scientific notation. For example,

```
print(format(57.467657, '10.2e'))
print(format(0.0033923, '10.2e'))
print(format(57.4, '10.2e'))
print(format(57, '10.2e'))
```

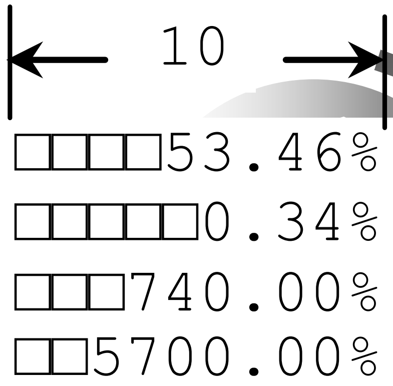


```
□□5.75e+01
□□3.39e-03
□□5.74e+01
□□5.70e+01
```

Formatting as a Percentage

You can use the conversion code % to format numbers as a percentage. For example,

```
print(format(0.53457, '10.2%'))
print(format(0.0033923, '10.2%'))
print(format(7.4, '10.2%'))
print(format(57, '10.2%'))
```

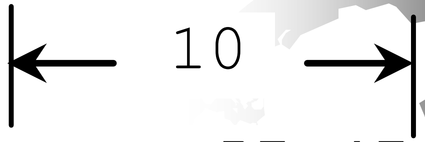


```
□□□□53.46%
□□□□□0.34%
□□□740.00%
□□5700.00%
```

Justifying Format

By default, the format is right justified. You can put the symbol < in the format specifier to specify that the item is a left justified in the resulting format within the specified width. For example,

```
print(format(57.467657, '10.2f'))
print(format(57.467657, '<10.2f'))
```

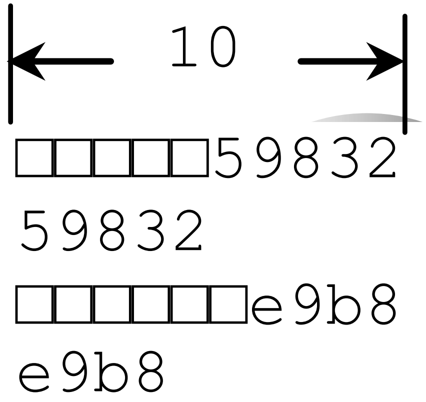


```
□□□□□57.47
57.47
```

Formatting Integers

You can use the conversion code d, x, o, and b to format an integer in decimal, hexadecimal, octal, or binary. You can specify a width for the conversion. For example,

```
print(format(59832, '10d'))
print(format(59832, '<10d'))
print(format(59832, '10x'))
print(format(59832, '<10x'))
```

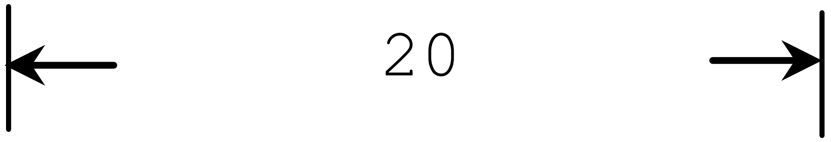


```
□□□□□59832
59832
□□□□□□e9b8
e9b8
```

Formatting Strings

You can use the conversion code s to format a string with a specified width. For example,

```
print(format("Welcome to Python", '20s'))
print(format("Welcome to Python", '<20s'))
print(format("Welcome to Python", '>20s'))
```



```
Welcome to Python
Welcome to Python
□□□Welcome to Python
```

FormatDemo

F-Strings

F-strings are new formatted strings since Python 3.6. An f-string is a string that begins with f or F. The string may contain expressions that are enclosed inside curly braces. The expressions are evaluated at runtime and formatted to strings. For example,

```
>>> weight = 140
>>> height = 73
>>> f"Weight is {weight} and height is {height}."
'Weight is 140 and height is 73.’
>>>
```

F-strings are more concise and more efficient than the format function. You can use f-strings to replace the format function. You can rewrite a format function call format(item, "format-specifier") using an f-string f"{item:format-specifier}".

FormatDemoUsingFString