

Chapter 3

Mathematical Functions, Strings, and Formatting

Objectives

- ▶ To solve mathematics problems by using the functions in the **math** module
- ▶ To represent and process strings and characters
- ▶ To encode characters using ASCII and Unicode
- ▶ To use **ord** to obtain a numerical code for a character
- ▶ To use **chr** to obtain a numerical code to a character
- ▶ To represent special characters using the escape sequence
- ▶ To invoke the **print** function with the end argument
- ▶ To convert numbers to a string using the **str** function
- ▶ To use the **+** operator to concatenate strings
- ▶ To format numbers and strings using the **format** function

Built-in Math Function

```
>>> max(2, 3, 4) # Returns a maximum number  
4
```

```
>>> min(2, 3, 4) # Returns a minimum number  
2
```

```
>>> round(3.51) # Rounds to its nearest integer  
4
```

```
>>> round(3.4) # Rounds to its nearest integer  
3
```

```
>>> abs(-3) # Returns the absolute value  
3
```

```
>>> pow(2, 3) # Same as 2 ** 3  
8
```

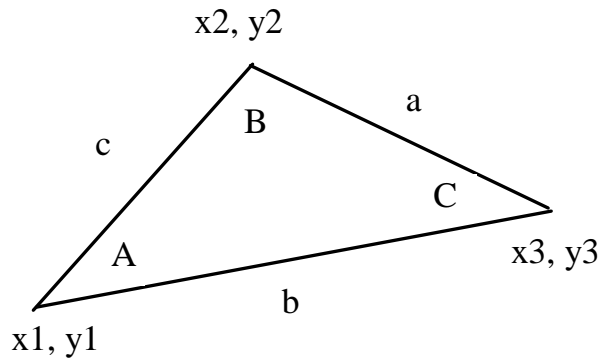
The math Functions

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

MathFunctions

Problem: Compute Angles

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



$$A = \arccos((a * a - b * b - c * c) / (-2 * b * c))$$
$$B = \arccos((b * b - a * a - c * c) / (-2 * a * c))$$
$$C = \arccos((c * c - b * b - a * a) / (-2 * a * b))$$

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'  
numChar = '4'  
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.

Unicode and ASCII Code

Python characters use *Unicode*, a 16-bit encoding scheme Python supports Unicode. Unicode is an encoding scheme for representing international characters. ASCII is a small subset of Unicode.

```
# NOTE:  this is a modified version of DisplayUnicode

import turtle
turtle.write("\u6B22 \u8FCE \u03b1 \u3b2 \u2740",font=("",40,""))
turtle.done()

# try adding the letter 'A', which is ASCII 65
# but you must use hexadecimal for turtle/Python/Unicode
```

DisplayUnicode

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

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
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
>>> num = 98
>>> chr(num)
>>> 'b'
```

Escape Sequences for Special Characters

<i>Description</i>	<i>Escape Sequence</i>	<i>Unicode</i>
Tab	\t	\u0009
Linefeed	\n	\u000A
Backslash	\\	\u005C
Single Quote	\'	\u0027
Double Quote	\"	\u0022

Printing without the Newline

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

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

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

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

Formatting Numbers and Strings

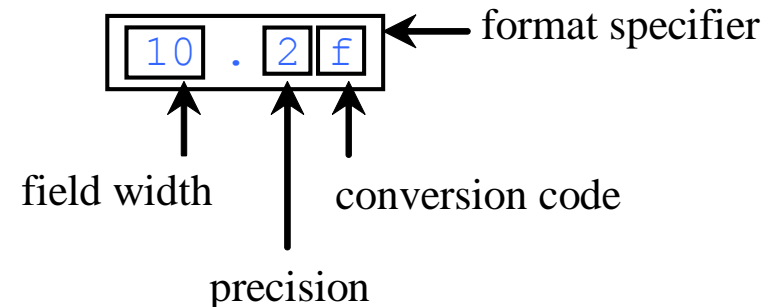
Often it is desirable to display numbers in certain format. For example, the following code computes the interest, given the amount and the annual interest rate.

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'))
```



← 10 →

□□□□57.47

12345678.92

□□□□57.40

□□□□57.00

Justifying Format

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

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

The diagram shows a horizontal line with vertical end caps. Above the line, a double-headed arrow spans a width of 10 units. Below the line, the number 57.47 is shown. To the left of the number, there are five empty square boxes, representing the padding used for left justification. The total width of the boxes plus the number is 10 units.

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'))
```

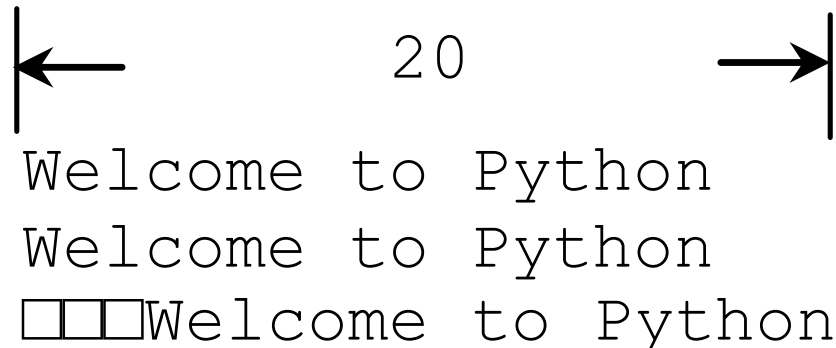
The diagram illustrates the effect of the '10' width specifier in the format string. It shows the number 59832 followed by five empty boxes, representing a total width of 10 characters. Above this, a double-headed arrow spans the width of the boxes and is labeled '10'. Below the boxes, the number 59832 is shown again, followed by the hexadecimal representation 'e9b8'.

← 10 →
□□□□□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'))
```



← 20 →

Welcome to Python

Welcome to Python

Welcome to Python