MATHEMATICAL, FUNCTIONS, STRINGS, AND OBJECTS

INTERNATIONAL COLLEGE, KMITL

PRESSESIONAL COURSE

OVERALL

- Common Python Functions
- Strings and characters
- Introduction to Objects and Methods
- Formatting Numbers and Strings
- Drawing Various Shapes
- Drawing with Colors and Fonts

COMMON PYTHON FUNCTIONS

SIMPLE PYTHON BUILT-IN FUNCTIONS

TABLE 3.1 Simple 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`

MATHEMATICAL FUNCTIONS

TABLE 3.2 Mathematical Functions

Function	*Description*	*Example*
`fabs(x)`	Returns the absolute value for `x` as a float.	`fabs(-2)` is `2.0`
`ceil(x)`	Rounds `x` up to its nearest integer and returns that integer.	`ceil(2.1)` is `3` `ceil(-2.1)` is `-2`
`floor(x)`	Rounds `x` down to its nearest integer and returns that 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.	`log(100, 10)` is `2.0`
`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`
`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`

** Required **
Math module

MATHEMATICAL FUNCTIONS :: EXAMPLE

```python
import math # import math module to use the math functions

# Test algebraic functions
print("exp(1.0) =", math.exp(1))
print("log(2.78) =", math.log(math.e))
print("log10(10, 10) =", math.log(10, 10))
print("sqrt(4.0) =", math.sqrt(4.0))

# Test trigonometric functions
print("sin(PI / 2) =", math.sin(math.pi / 2))
print("cos(PI / 2) =", math.cos(math.pi / 2))
print("tan(PI / 2) =", math.tan(math.pi / 2))
print("degrees(1.57) =", math.degrees(1.57))
print("radians(90) =", math.radians(90))
```

```
exp(1.0) = 2.71828182846
log(2.78) = 1.0
log10(10, 10) = 1.0
sqrt(4.0) = 2.0
sin(PI / 2) = 1.0
cos(PI / 2) = 6.12323399574e-17
tan(PI / 2) = 1.63312393532e+16
degrees(1.57) = 89.9543738355
radians(90) = 1.57079632679
```

STRINGS AND CHARACTERS

STRINGS AND CHARACTERS

- **String** is a sequence of a characters (include text and numbers)
- String values must be enclosed in matching **single quotes** (‘) or **double quotes** (“)
- Python doesn’t have char- datatype.
- Single character string = character

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

ASCII CODE

The ASCII code

American Standard Code for Information Interchange

www.theasciicode.com.ar

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ò	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

** Lowercase = Uppercase + 32 **

UNICODE

Microsoft Windows Codepage : 1252 (Latin I)

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC		‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	Š 0160	‹ 2039	Œ 0152		Ž 017D	
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	˜ 02DC	™ 2122	š 0161	› 203A	œ 0153		ž 017E	Ÿ 0178
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

A Unicode starts with ‘\u’, followed by four hexadecimal digits that run from ‘\u0000’ to ‘\uFFFF’

LISTING 3.3 DisplayUnicode.py

```
import turtle

turtle.write("\u6B22\u8FCE \u03b1 \u03b2 \u03b3")

turtle.done()
```

THE ORD() AND CHR() FUNCTIONS

```
>>> ch = 'a'
>>> ord(ch)
97
>>> chr(98)
'b'
>>> ord('A')
65
>>>
```

```
>>> ord('a') - ord('A')
32
>>> ord('d') - ord('D')
32
>>> offset = ord('a') - ord('A')
>>> lowercaseLetter = 'h'
>>> uppercaseLetter = chr(ord(lowercaseLetter) - offset)
>>> uppercaseLetter
'H'
>>>
```

ESCAPE SEQUENCES

TABLE 3.3 Python Escape Sequences

Character Escape Sequence	*Name*	*Numeric Value*
`\b`	Backspace	8
`\t`	Tab	9
`\n`	Linefeed	10
`\f`	Formfeed	12
`\r`	Carriage Return	13
`\\`	Backslash	92
`\'`	Single Quote	39
`\"`	Double Quote	34

```
>>> print("He said, \"John's program is easy to read\"")
He said, "John's program is easy to read"
```

PRINTING WITHOUT THE NEWLINE

For example, the following code

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

displays

```
AAA BBBCCC***DDD***
```

PRINTING WITHOUT THE NEWLINE

Syntax:

```
print(item1, item2, ..., end = "anyendingstring")
```

For example,

```
radius = 3
print("The area is", radius * radius * math.pi, end = ' ')
print("and the perimeter is", 2 * radius)
```

displays

```
The area is 28.26 and the perimeter is 6
```

THE STR() FUNCTION

```
>>> s = str(3.4) # Convert a float to string
>>> s
'3.4'
>>> s = str(3) # Convert an integer to string
>>> s
'3'
>>>
```

THE STRING CONCATENATION OPERATOR

- You can use the + operator to concatenate two strings.

```
>>> message = "Welcome " + "to " + "Python"
>>> message
'Welcome to Python'
>>> chapterNo = 3
>>> s = "Chapter " + str(chapterNo)
>>> s
'Chapter 3'
>>>
```

THE STRING CONCATENATION OPERATOR

- The augmented assignment += operator can also be used

```
>>> message = "Welcome to Python"
>>> message
'Welcome to Python'
>>> message += " and Python is fun"
>>> message
'Welcome to Python and Python is fun'
>>>
```

READING STRINGS FROM THE CONSOLE

```
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)
```

```
Enter a string: Welcome ↵Enter
Enter a string: to ↵Enter
Enter a string: Python ↵Enter
s1 is Welcome
s2 is to
s3 is Python
```

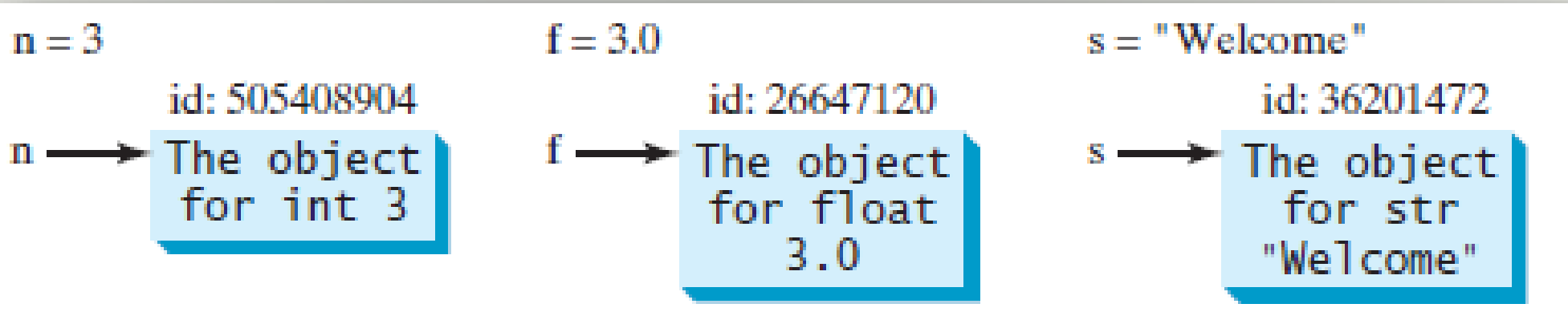
OBJECTS AND METHODS

METHODS: ID() AND TYPE()

```
>>> n = 3  # n is an integer
>>> id(n)
505408904
>>> type(n)
<class 'int'>
>>> f = 3.0  # f is a float
>>> id(f)
26647120
>>> type(f)
<class 'float'>
>>> s = "Welcome" # s is a string
>>> id(s)
36201472
>>> type(s)
<class 'str'>
>>>
```

METHODS: ID() AND TYPE()

The relationship between the variables and objects:



METHODS: LOWER(), UPPER(), STRIP()

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

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

```
s = input("Enter a string").strip()
```

FORMATTING NUMBERS AND STRINGS

FREQUENTLY USED SPECIFIERS

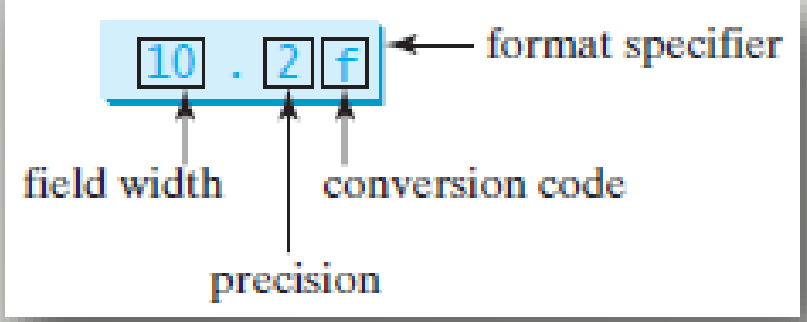
TABLE 3.4 Frequently Used Specifiers

Specifier	*Format*
`"10.2f"`	Format the float item with width 10 and precision 2.
`"10.2e"`	Format the float item in scientific notation with width 10 and precision 2.
`"5d"`	Format the integer item in decimal with width 5.
`"5x"`	Format the integer item in hexadecimal with width 5.
`"5o"`	Format the integer item in octal with width 5.
`"5b"`	Format the integer item in binary with width 5.
`"10.2%"`	Format the number in decimal.
`"50s"`	Format the string item with width 50.
`"<10.2f"`	Left-justify the formatted item.
`">10.2f"`	Right-justify the formatted item.

FORMATTING

Syntax:

format (item, format-specifier)



Item is a number or a string

Format-specifier is a string that specifics how the item is formatted

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"))
```

displays

```
|← 10 →|
□□□□□ 57.47
 123456782.92
□□□□□ 57.40
□□□□□ 57.00
```

A square box (□) denotes a blank space.
The decimal point is counted as one space

```
print(format(57.467657, "10.2f"))
print(format(57.467657, ".2f"))
```

displays

```
|← 10 →|
□□□□□ 57.47
57.47
```

FORMATTING :: SCIENTIFIC NOTATION

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

displays

```
|←— 10 —→|
□□ 5.75e+01
□□ 3.39e-03
□□ 5.74e+01
□□ 5.70e+01
```

The + and – signs are counted as places in the width limit

FORMATTING :: PERCENTAGE

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

displays

```
←— 10 —→|
□□□□ 53.46%
□□□□□ 0.34%
□□□ 740.00%
□□ 5700.00%
```

The Format 10.2% causes the number to be multiplied by 100 and displayed % sign.

The % sign counted as one space

FORMATTING :: JUSTIFYING FORMAT

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

displays

```
|←— 10 —→|
□□□□□ 57.47
57.47
```

FORMATTING :: INTEGERS

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

displays

```
|← 10 →|
□□□□□ 59832
59832
□□□□□ e9b8
e9b8
```

d = decimal
x = hexadecimal
o = octal
b = binary

FORMATTING :: STRINGS

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

displays

```
|←———— 20 ————→|
Welcome to Python
Welcome to Python
□□□ Welcome to Python
Welcome to Python and Java
```

If the string is longer than the specified width,
The width is automatically increase to fit the string

DRAWING VARIOUS SHAPES

TURTLE :: PEN DRAWING STATE METHODS

Method	*Description*
`turtle.pendown()`	Pulls the pen down—drawing when moving.
`turtle.penup()`	Pulls the pen up—no drawing when moving.
`turtle.pensize(width)`	Sets the line thickness to the specified width.

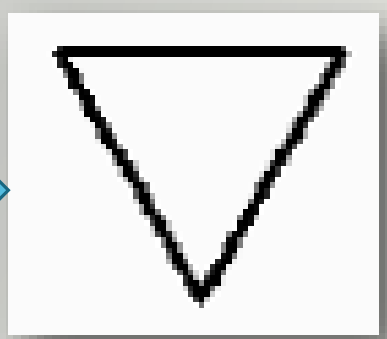
TURTLE :: MOTION METHODS

Method	*Description*
`turtle.forward(d)`	Moves the turtle forward by the specified distance in the direction the turtle is headed.
`turtle.backward(d)`	Moves the turtle backward by the specified distance in the opposite direction the turtle is headed. The turtle's direction is not changed.
`turtle.right(angle)`	Turns the turtle right by the specified angle.
`turtle.left(angle)`	Turns the turtle left by the specified angle.
`turtle.goto(x, y)`	Moves the turtle to an absolute position.
`turtle.setx(x)`	Moves the turtle's x-coordinate to the specified position.
`turtle.sety(y)`	Moves the turtle's y-coordinate to the specified position.
`turtle.setheading(angle)`	Sets the orientation of the turtle to a specified angle. 0-East, 90-North, 180-West, 270-South.
`turtle.home()`	Moves the turtle to the origin (0, 0) and east direction.
`turtle.circle(r, ext, step)`	Draws a circle with the specified radius, extent, and step.
`turtle.dot(diameter, color)`	Draws a circle with the specified diameter and color.
`turtle.undo()`	Undo (repeatedly) the last turtle action(s).
`turtle.speed(s)`	Sets the turtle's speed to an integer between 1 and 10, with 10 being the fastest.

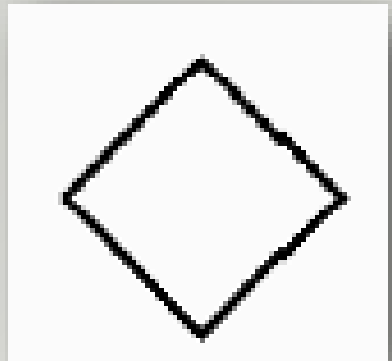
TURTLE :: CIRCLE

```
import turtle
```

```
turtle.pensize(3) # Set pen thickness to 3 pixels
turtle.penup() # Pull the pen up
turtle.goto(-200, -50)
turtle.pendown() # Pull the pen down
turtle.circle(40, steps = 3) # Draw a triangle
```



```
turtle.penup()
turtle.goto(-100, -50)
turtle.pendown()
turtle.circle(40, steps = 4) # Draw a square
```

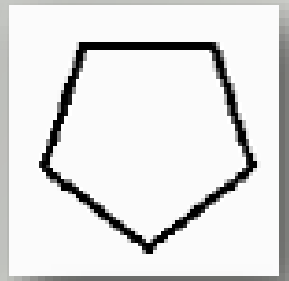


```
turtle.done()
```

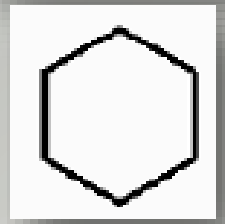
TURTLE :: CIRCLE

```
import turtle
```

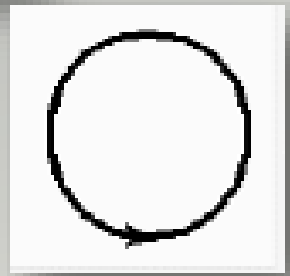
```
turtle.penup()
turtle.goto(0, -50)
turtle.pendown()
turtle.circle(40, steps = 5) # Draw a pentagon
```



```
turtle.penup()
turtle.goto(100, -50)
turtle.pendown()
turtle.circle(40, steps = 6) # Draw a hexagon
```

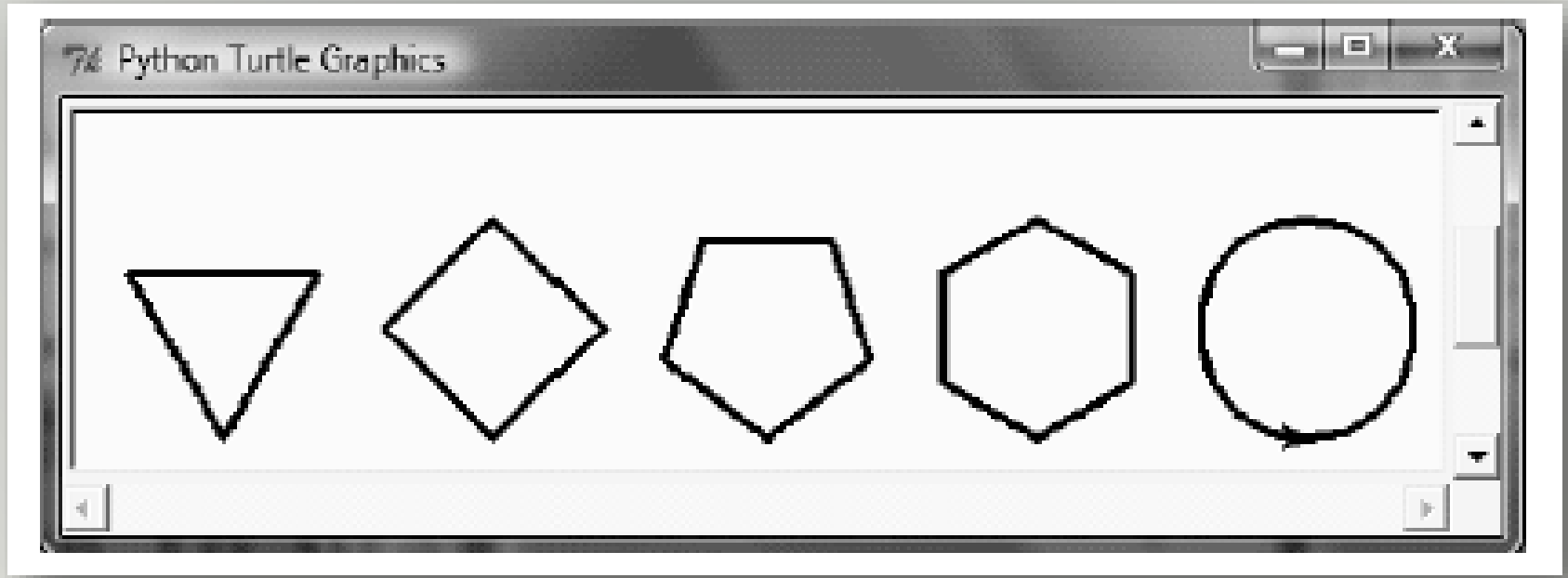


```
turtle.penup()
turtle.goto(200, -50)
turtle.pendown()
turtle.circle(40) # Draw a circle
```



```
turtle.done()
```

TURTLE :: CIRCLE



DRAWING WITH COLORS AND FONTS

TURTLE :: SHAPE

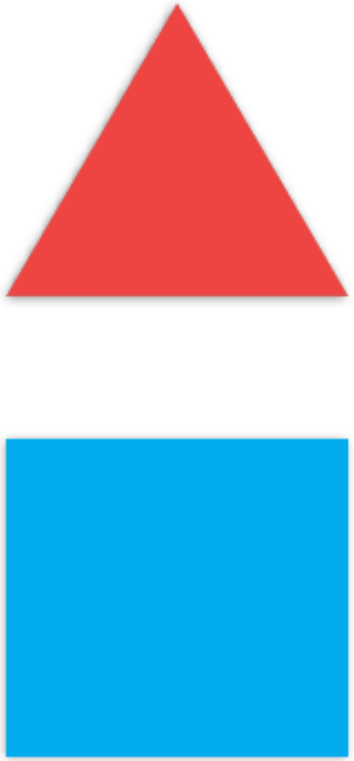
Method	*Description*
`turtle.color(c)`	Sets the pen color.
`turtle.fillcolor(c)`	Sets the pen fill color.
`turtle.begin_fill()`	Calls this method before filling a shape.
`turtle.end_fill()`	Fills the shapes drawn before the last call to `begin_fill`.
`turtle.filling()`	Returns the fill state: `True` if filling, `False` if not filling.
`turtle.clear()`	Clears the window. The state and the position of the turtle are not affected.
`turtle.reset()`	Clears the window and reset the state and position to the original default value.
`turtle.screensize(w, h)`	Sets the width and height of the canvas.
`turtle.hideturtle()`	Makes the turtle invisible.
`turtle.showturtle()`	Makes the turtle visible.
`turtle.isvisible()`	Returns `True` if the turtle is visible.
`turtle.write(s, font=("Arial", 8, "normal"))`	Writes the string `s` on the turtle position. Font is a triple consisting of fontname, fontsize, and fonttype.

TURTLE :: SHAPE

```
import turtle

turtle.pensize(3)
turtle.penup()
turtle.goto(-200, -50)
turtle.pendown()
turtle.begin_fill()
turtle.color("red")
turtle.circle(40, steps = 3)
turtle.end_fill()

turtle.penup()
turtle.goto(-100, -50)
turtle.pendown()
turtle.begin_fill()
turtle.color("blue")
turtle.circle(40, steps = 4)
turtle.end_fill()
```

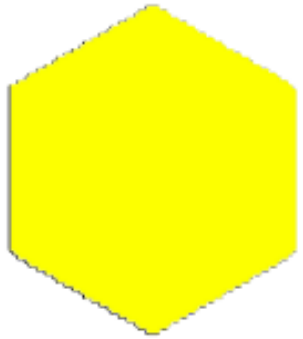


TURTLE :: SHAPE

```
import turtle

turtle.penup()
turtle.goto(0, -50)
turtle.pendown()
turtle.begin_fill()
turtle.color("green")
turtle.circle(40, steps = 5)
turtle.end_fill()

turtle.penup()
turtle.goto(100, -50)
turtle.pendown()
turtle.begin_fill()
turtle.color("yellow")
turtle.circle(40, steps = 6)
turtle.end_fill()
```



TURTLE :: SHAPE

```
import turtle
```

```
turtle.penup()
turtle.goto(200, -50)
turtle.pendown()
turtle.begin_fill()
turtle.color("purple")
turtle.circle(40)
turtle.end_fill()
turtle.color("green")
turtle.penup()
turtle.goto(-100, 50)
turtle.pendown()
turtle.write("Cool Colorful Shapes",
    font = ("Times", 18, "bold"))
turtle.hideturtle()

turtle.done()
```

TURTLE :: SHAPE