



Python

Programming

Variables & Operations

Dr. Chun-Hsiang Chan
Department of Geography
National Taiwan Normal University



Outlines

- Variables
- Data Types
- Strings
- Booleans
- Operators
- Assignments





Variables

- As a programming language, Python also have some striction during variable declaration.
- In addition, Python is a special programming language that all variables just like chameleon, which I will mention later.
- First of all, the rule of variable names: try the followings,

```
avar = "hello"  
a_var = "hello"  
_var = "hello"  
aVar = "hello"  
var2 = "hello"
```

```
1avar = "hello"  
1a_var = "hello"  
a var = "hello"  
a-Var = "hello"  
a@var2 = "hello"
```



Variables

- Variable name style with multi-word combinations

- **Camel case**

```
aVarEx = 3
```

- **Pascal case**

```
AVarEx = 3
```

- **Snake case**

```
a_var_ex = 3
```

- **Delete a variable**

```
del a_var_ex
```



Variables

- Declare a variable without casting

```
a = 3  
b = 3.7
```

- Declare a variable with casting

```
a = str(3)  
b = float(3.7)
```

- Get data type information

```
print(type(a))  
print(type(b))
```



Variables

- Assign **multiple variables** at one time.

```
a, b, c = 3, 3.5, "master"  
print(a)  
print(b)  
print(c)  
abc = [3, 3.5, "master"]  
a, b, c = abc  
print(a)  
print(b)  
print(c)
```



Variables

- Print the **multiple variables** with **concatenation** or **formation**.

```
a, b, c = "I", "am", "master"
```

```
print(a, b, c)
```

```
print(a + b + c)
```

```
abc = "I am master"
```

```
print(abc)
```

```
d = 10
```

```
e = 20
```

```
print(d+e)
```

```
# add ending symbol
```

```
print(abc, end="@")
```

```
print(abc, end="!")
```

```
# formatting the numbers
```

```
m = 123.456789
```

```
print("{0:.2f}".format(m))
```

```
print("{0:.3f}".format(m))
```

```
print(round(m, 2))
```



Data Types

- In Python, there are several data types: text, numeric, sequence, mapping, set, boolean, binary, and none types.

text type

str

numeric type

int, float, double, complex

sequence type

list, tuple, range

mapping type

dict

set type

set, frozenset

boolean type

bool

binary type

byte, bytearray, memoryview

none type

NoneType



Data Types

A = "master"

A = 20

A = 20.567

A = 2j

A = ["m1", "m2", "m3"]

A = ("m1", "m2", "m3")

A = range(10)

A = {"name": "mike", "wt": 65}

A = {"m1", "m2", "m3"}

str

int

float

complex

list

tuple

range

dict

set



Data Types and Numbers

```
A = frozenset({"m1", "m2", "m3"})
```

frozenset

```
A = True
```

bool

```
A = b"m1"
```

bytes

```
A = bytearray(10)
```

bytearray

```
A = memoryview(bytes(20))
```

memoryview

```
A = None
```

NoneType

• Numbers: Special case

```
x = 3e10 # what is the data type of x? test and run
```

```
y = 3E10 # what is the data type of y? test and run
```



Strings

- String is the most common data type in Python, and we may use different ways for declaring a string.

- **Multiple line string**

```
a = "once upon a time, there was a kingdom ..."  
b = "once upon a time, \nthere was a kingdom ..."  
c = """once upon a time, there was a kingdom  
with a large territory"""
```

- **Indexing a string**

```
a[1], a[:10], a[2:8], a[10:], a[-10:-1], a[-8:]
```



Strings

- In some case, we want to change the format of all strings in one time, for example ...

```
a = "once upon a time, there was a kingdom ..."  
print(a.upper()) # returns the string in upper case  
print(a.lower()) # returns the string in upper case  
print(a.strip()) # returns the string without space from the  
beginning and the end  
print(a.replace("o", "X")) # replaces the specific words  
print(a.split(",")) # split the string by comma
```



Strings

- Speaking of splitting a string, we could concatenate strings together or format a string.

```
a, b, c = "I", "am", "master"  
print(a, b, c)  
print(a + b + c)  
print(a + " " + b + " " + c)  
age = 18  
txt = "Hey, I'm Mike and {} year-old"  
print(txt.format(age))
```



Strings

- Speaking of splitting a string, we could concatenate strings together or format a string.

```
age = 3
height = 567
weight = 49.95
txtOrder = "My sister's height and weight are {2} and {0} ,
respectively, while she is {1} year-old."
print(txtOrder.format(weight, age, height))
```



Strings

- How to type in some special characters?

`\'` Single quote

`\\` Backslash

`\n` New line

`\t` Tab

`\b` Backspace

`\ooo` Octal value

`\xhh` Hex value



Strings – Octal and Hex Value

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	\$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

```
# octal
a = "\110\145\154\154\157"
print(a)
# print NTNU with octal and hex
# ...
```

Source: https://www.reddit.com/r/coolguides/comments/e2pp5r/decimal_binary_octal_hex_ascii_conversion_chart/?rdt=59206

Chun-Hsiang Chan (2026)

Strings – Methods



Method	Description
<code>capitalize()</code>	Converts the first character to upper case
<code>casefold()</code>	Converts string into lower case
<code>center()</code>	Returns a centered string
<code>count()</code>	Returns the number of times a specified value occurs in a string
<code>encode()</code>	Returns an encoded version of the string
<code>endswith()</code>	Returns true if the string ends with the specified value
<code>expandtabs()</code>	Sets the tab size of the string
<code>find()</code>	Searches the string for a specified value and returns the position of where it was found
<code>format()</code>	Formats specified values in a string
<code>format_map()</code>	Formats specified values in a string
<code>index()</code>	Searches the string for a specified value and returns the position of where it was found
<code>isalnum()</code>	Returns True if all characters in the string are alphanumeric
<code>isalpha()</code>	Returns True if all characters in the string are in the alphabet
<code>isascii()</code>	Returns True if all characters in the string are ascii characters
<code>isdecimal()</code>	Returns True if all characters in the string are decimals
<code>isdigit()</code>	Returns True if all characters in the string are digits
<code>isidentifier()</code>	Returns True if the string is an identifier
<code>islower()</code>	Returns True if all characters in the string are lower case
<code>isnumeric()</code>	Returns True if all characters in the string are numeric
<code>isprintable()</code>	Returns True if all characters in the string are printable

Source: https://www.w3schools.com/python/python_strings_methods.asp

<code>isspace()</code>	Returns True if all characters in the string are whitespaces
<code>istitle()</code>	Returns True if the string follows the rules of a title
<code>isupper()</code>	Returns True if all characters in the string are upper case
<code>join()</code>	Joins the elements of an iterable to the end of the string
<code>ljust()</code>	Returns a left justified version of the string
<code>lower()</code>	Converts a string into lower case
<code>lstrip()</code>	Returns a left trim version of the string
<code>maketrans()</code>	Returns a translation table to be used in translations
<code>partition()</code>	Returns a tuple where the string is parted into three parts
<code>replace()</code>	Returns a string where a specified value is replaced with a specified value
<code>rfind()</code>	Searches the string for a specified value and returns the last position of where it was found
<code>rindex()</code>	Searches the string for a specified value and returns the last position of where it was found
<code>rjust()</code>	Returns a right justified version of the string
<code>rpartition()</code>	Returns a tuple where the string is parted into three parts
<code>rsplit()</code>	Splits the string at the specified separator, and returns a list
<code>rstrip()</code>	Returns a right trim version of the string
<code>split()</code>	Splits the string at the specified separator, and returns a list
<code>splitlines()</code>	Splits the string at line breaks and returns a list
<code>startswith()</code>	Returns true if the string starts with the specified value
<code>strip()</code>	Returns a trimmed version of the string
<code>swapcase()</code>	Swaps cases, lower case becomes upper case and vice versa
<code>title()</code>	Converts the first character of each word to upper case
<code>translate()</code>	Returns a translated string
<code>upper()</code>	Converts a string into upper case
<code>zfill()</code>	Fills the string with a specified number of 0 values at the beginning



Booleans

- In Python, there are two boolean values: **True** and **False**.

```
# basic
print(1>2)
print(1>=2)
print(2==2)
print(1<2)
# try some specials
print(bool(12))
print(bool("am"))
```



Operators

- For sure, you may do some mathematic calculation.

Operator	Name	Example
+	addition	$x + y$
-	subtraction	$x - y$
*	multiplication	$x * y$
/	division	x / y
%	modulus	$x \% y$
**	exponentiation	$x ** y$
//	floor division	$x // y$

Source: https://www.w3schools.com/python/python_operators.asp

Operators

- There are some fantastic operators.

Operator	Example	Same as
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
=	x = 3	x = x 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3



Assignment #01

- In Taiwan, our government adopts the Richter Scale to quantify a seismic event's magnitude. However, news reporters prefer to conduct the number of TNT bombs equivalent to seismic energy. Here are the energy equations.

$$\log_{10} E = 4.4 + 1.5 \times M_L$$

$$1 \text{ ton of TNT} = 4.184 \times 10^9 J$$

- In addition, the equivalence Hiroshima-bomb is 12.5 *kT* TNT.
- Design a function that the user may give a M_L value, and it will return the number of Hiroshima-bombs.



Assignment #01

Format

```
def earthquake2bomb(ML):  
    # annotation  
    ...  
    ...  
    ...  
    return num_bomb
```

- You need to notice the precision issue.
- The return value should be an integer.
- You may use a function called “round” to obtain an integer number.

The End

Thank you for your attention!

Email: chchan@ntnu.edu.tw

Website: <https://toodou.github.io/>

