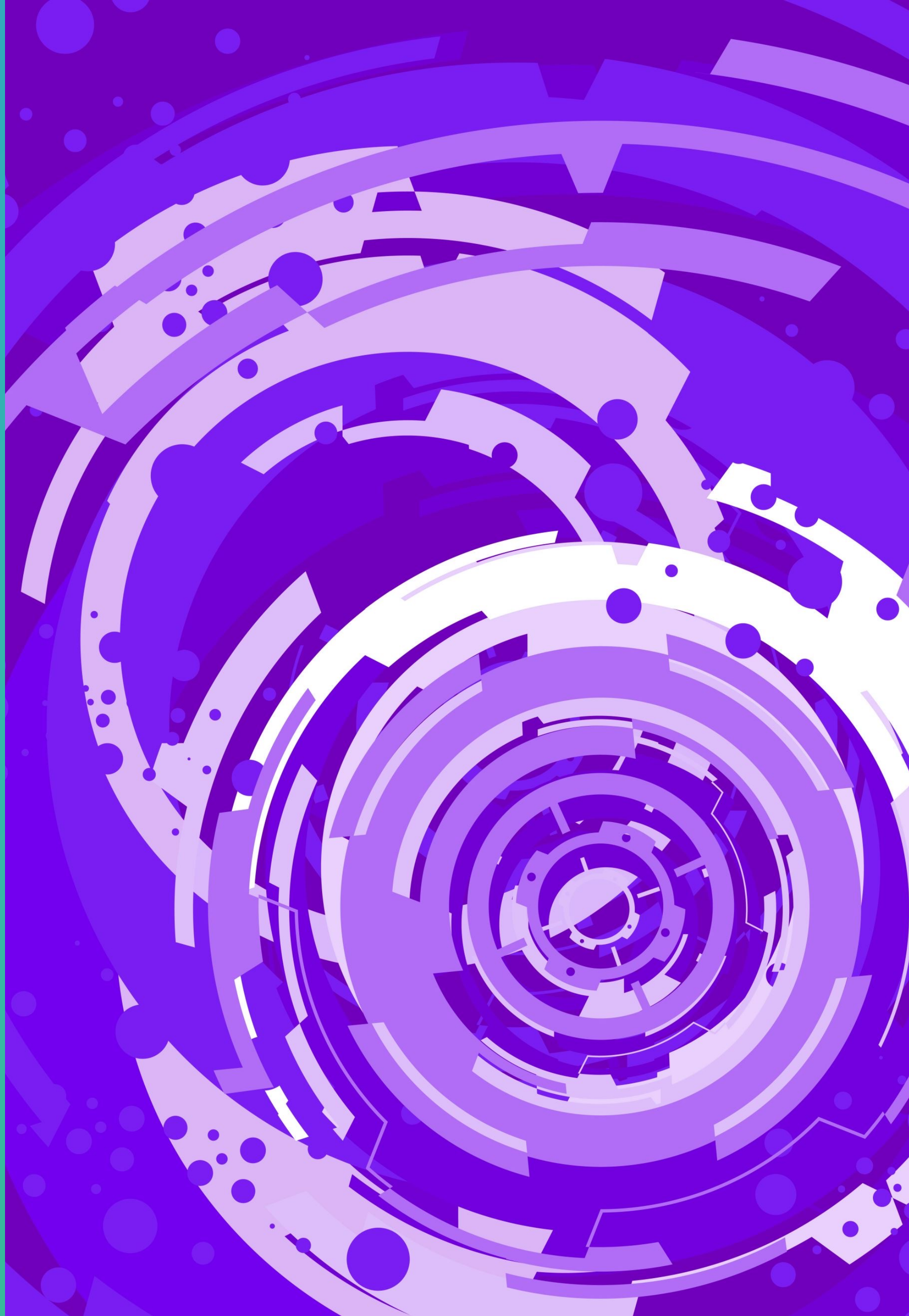
URBAN GEOGRAPHIC INFORMATION SYSTEM

Python Basic I – Variables

Chun-Hsiang Chan

Department of Geography,

National Taiwan Normal University



CHUN-HSIANG CHAN (2023)

Outline

- Variables
- Data Types
- Numbers
- Strings
- Booleans
- Operators
- Lists
- Tuples
- Sets
- Dictionaries

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"                1avar = "hello"
a_var = "hello"               1a_var = "hello"
_var = "hello"                a var = "hello"
aVar = "hello"                a-Var = "hello"
var2 = "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="!")

# formating 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"`	str
`A = 20`	int
`A = 20.567`	float
`A = 2j`	complex
`A = ["m1", "m2",  "m3"]`	list
`A = ("m1", "m2",  "m3")`	tuple
`A = range(10)`	range
`A = {"name": "mike", "wt":65}`	dict
`A = {"m1", "m2",  "m3"}`	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.

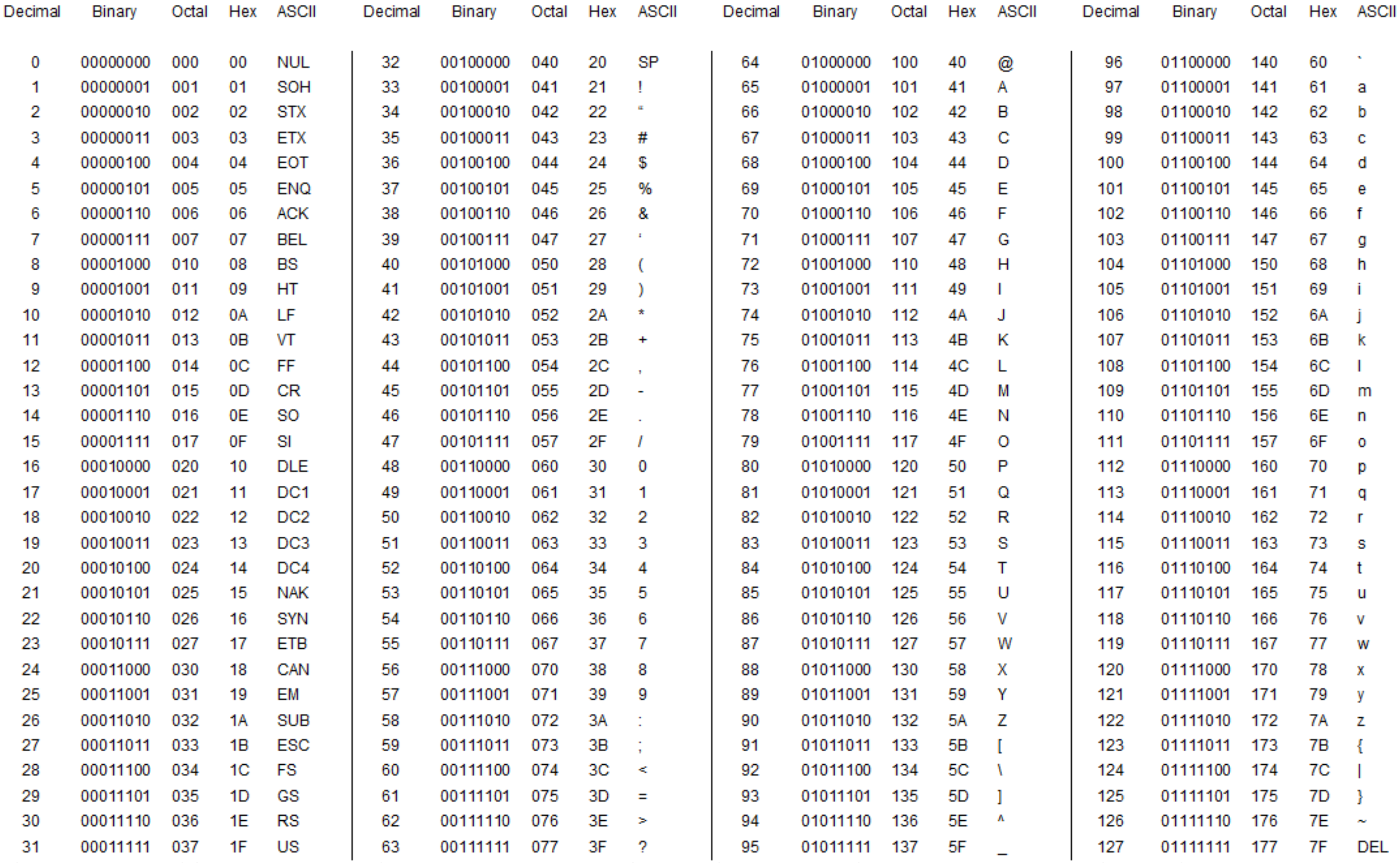
```
age = 3
height = 567
weight = 49.95
txtOrder = "My sister’s height and weight are {2} and {0} , respectively, while she is {1} year-old."
print(txtOrder.format(weight, age, height))
```

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

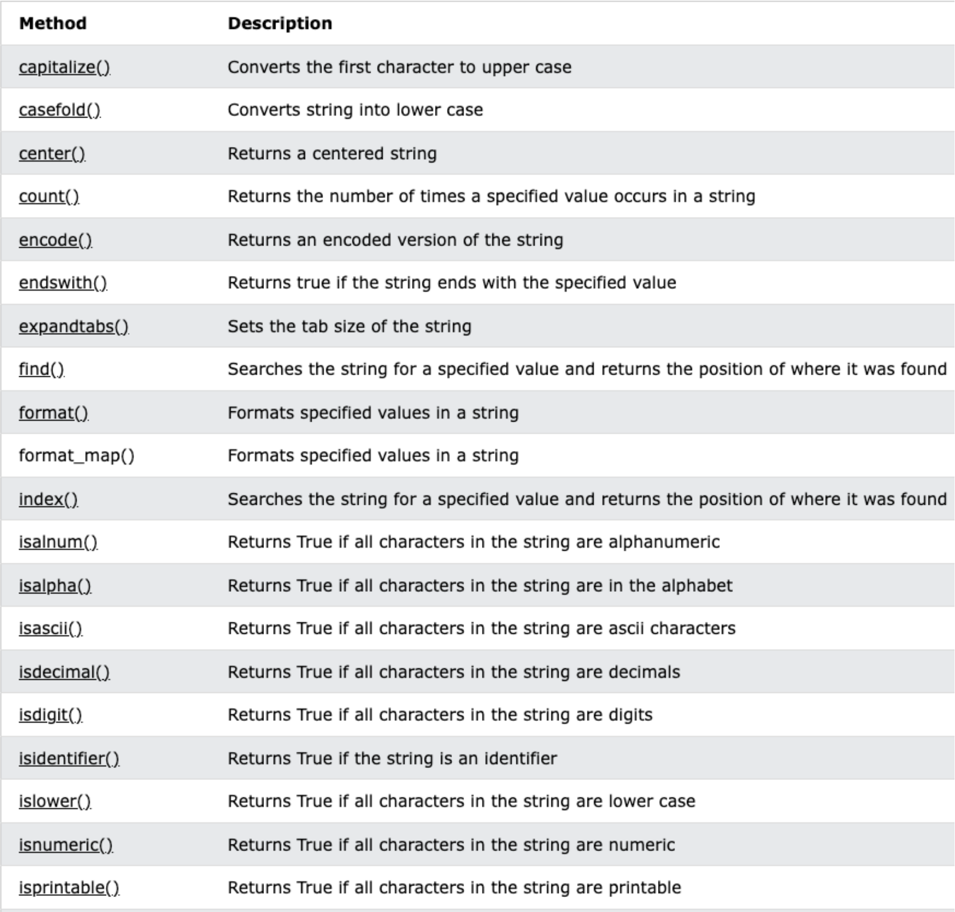


Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	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

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

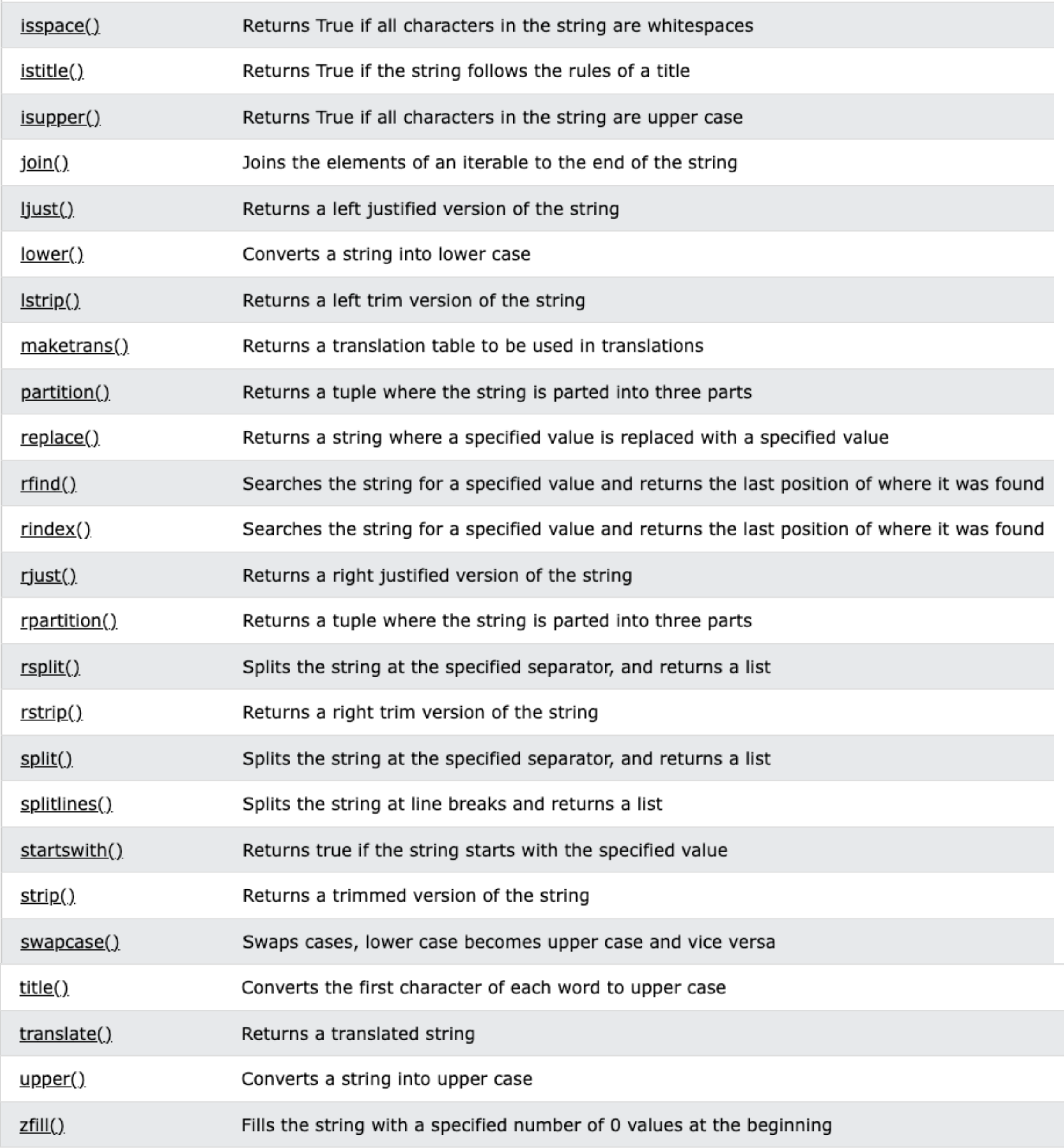
```
# octal
a = "\110\145\154\154\157"
print(a)
# print NTNU with octal and hex
# …
```

Strings – Methods



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

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



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

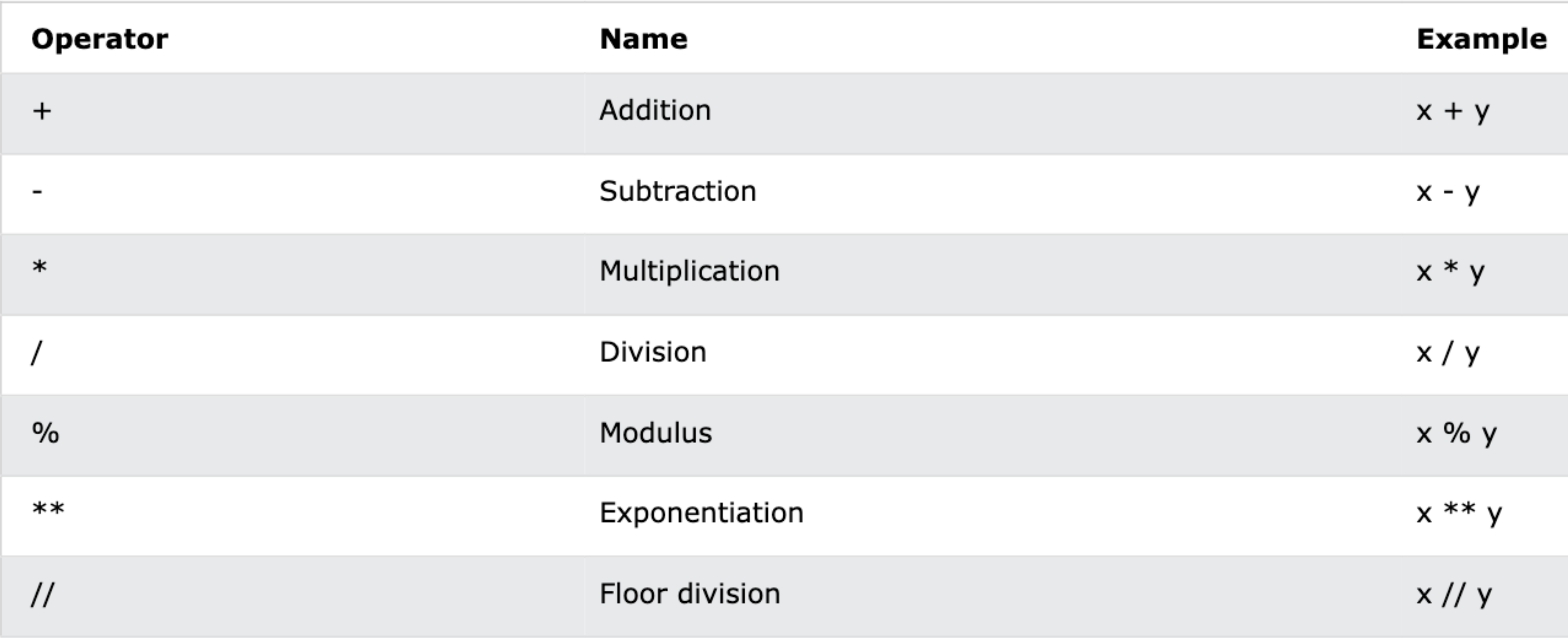
Booleans

- 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.

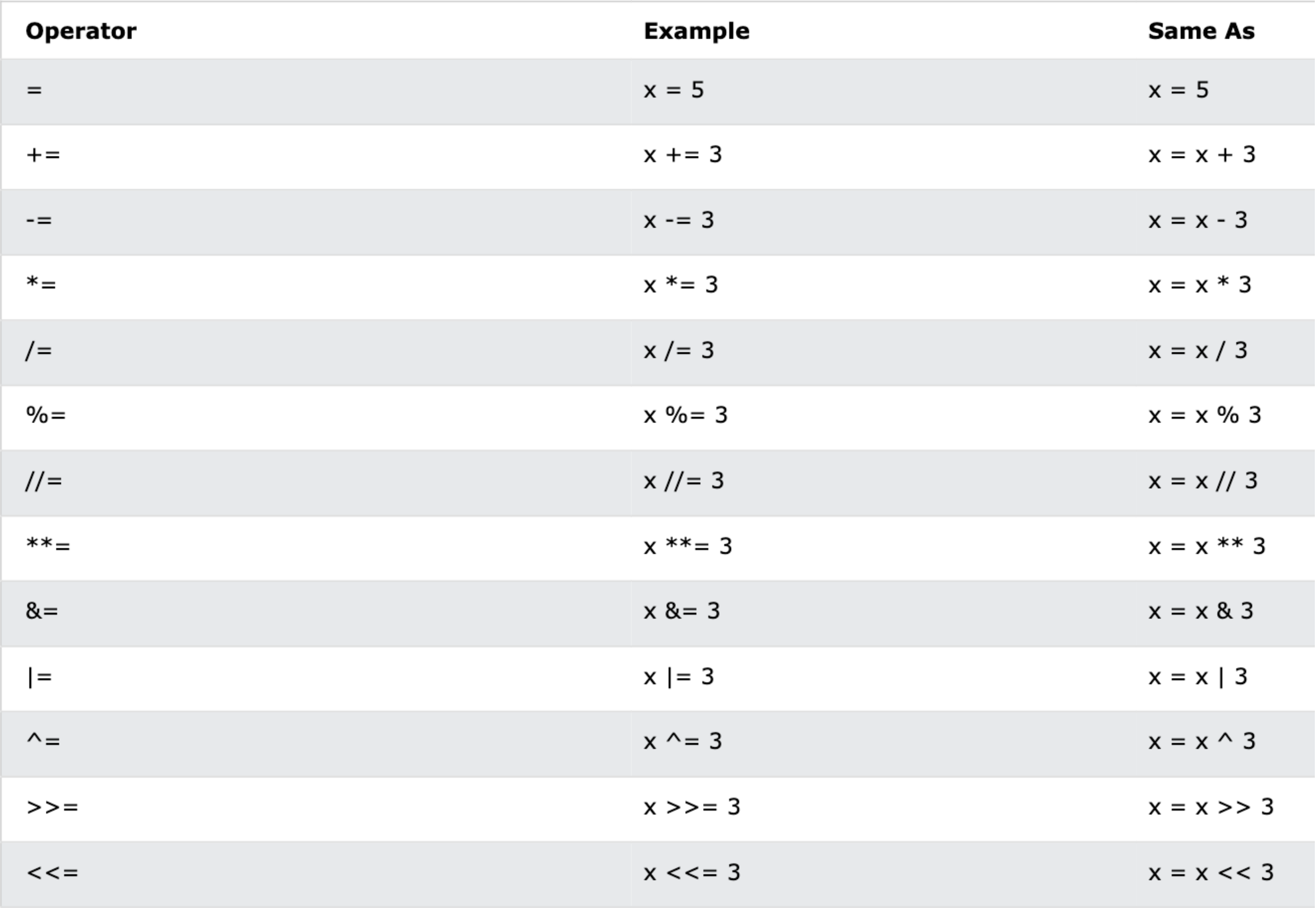


Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

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

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

Before starting to know, …

- There are **four collection** data types in the Python programming:
- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- **Dictionary** is a collection which is ordered** and changeable. No duplicate members.

https://www.w3schools.com/python/python_tuples.asp

Lists – Fundamentals

- List is the most powerful data type in Python, which I think at least. Because you may add or insert any data type into the list whereever you like. Usually, we can use the list as an array.

```
A = [1.2, 3.14, 100]
print(A)
print(type(A))
print(len(A))
B = [(1.2, 3.14, 100)]
print(B)
```

Lists – Indexing

- After knowing the list, there is onething that you have to know…

```
abc = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(abc[1])
print(abc[-1])
print(abc[1:])
print(abc[-5:])
print(abc[-3:-1])
```

Lists – Change

- After knowing the list, there is onething that you have to know…

```
abc = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10]
abc[1:4] = [2, 2, 2]
print(abc)

abc[5:] = [2, 2, 2]
print(abc)

abc.insert(3, 999)
print(abc)
```

Lists – Add

- After knowing the list, there is onething that you have to know…

```
# continue using the previous list for the following practice
abc.append(9999)
print(abc)

abc.extend([3333])
print(abc)

abc.extend((3333, 5555, 6666))
print(abc)
```

Lists – Remove

- After knowing the list, there is onething that you have to know…

```
# continue using the previous list for the following practice
abc. remove(9999)
print(abc)
abc.remove(10)
print(abc)
abc.pop(1)
print(abc)
```

```
abc. pop()
print(abc)

del abc[10]
print(abc)

abc.clear()
print(abc)
```

Lists – Sort

- Usually, you may want to re-order your dataset in some order.

```
# given two types of lists for list sorting
num = [3, 24, 13, 41, -50, 26, -17, 18, 99, 140, 1110, 190]
mystr = ['doctor','part','unique','college','taiwan','apple']
num.sort()
mystr.sort()
print(num)
print(mystr)
mystr.sort(reverse = True) # plz try ➔ mystr.reverse()
print(mystr)
```

Lists – Copy

- In data analysis, you may copy your list twice or more for different scenarios. **Notice:** you cannot just use b_list = a_list because b_list will only be a *reference* to a_list, and all changes you made on/in a_list will automatically also be made in b_list.

```
# make an experiment to prove it!
a_list = [1,2,3,4,5]
b_list = a_list
b_list[2] = 999
print(a_list) # what is the answer?
```

Lists – Copy

- So, how to copy a list?

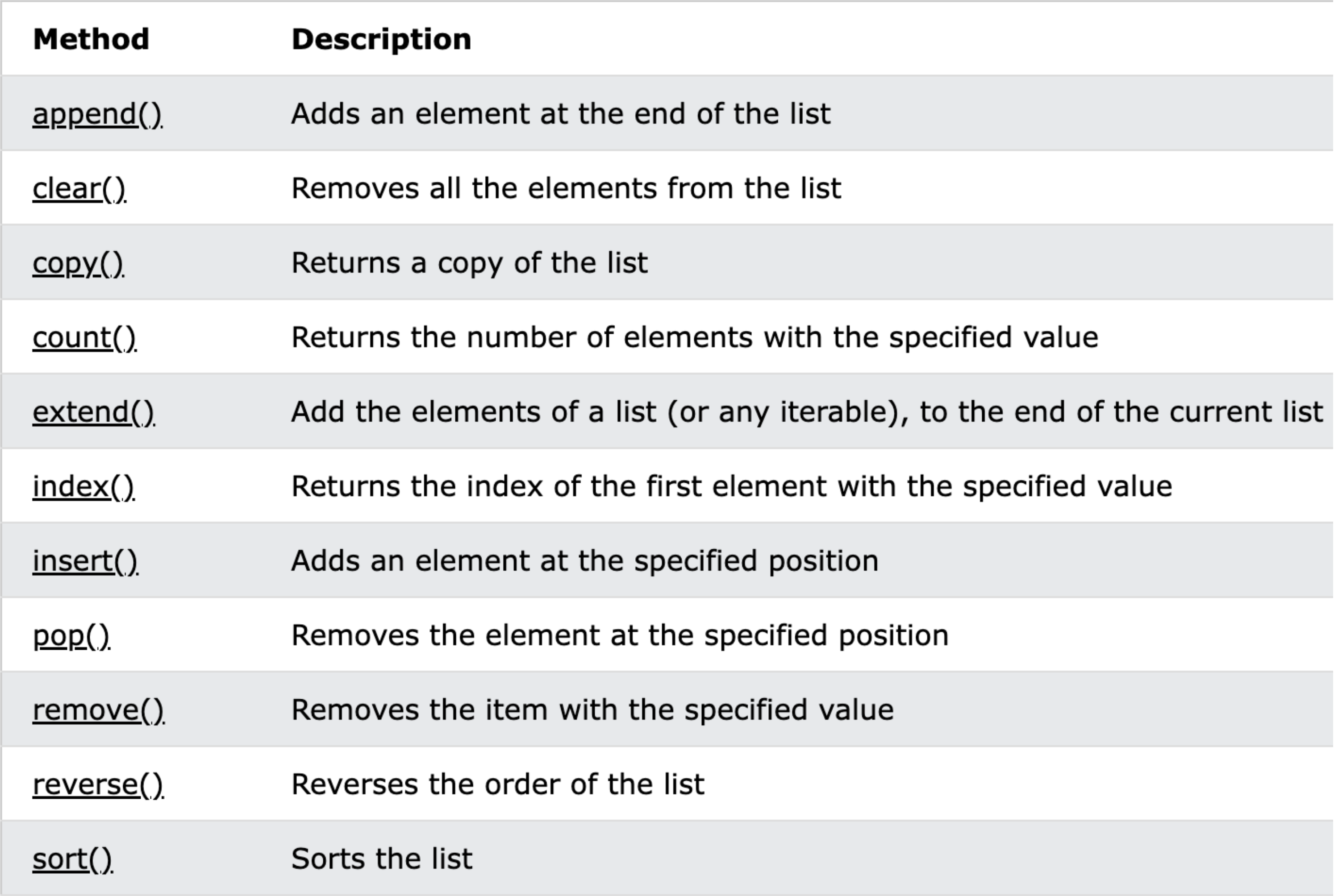
```
# directly use the function of "copy"
a_list = [1,2,3,4,5]
b_list = a_list.copy()
b_list[2] = 999
print(a_list, b_list) # what is the answer?
# another method
b_list = list(a_list)
b_list[2] = 999
print(a_list, b_list) # what is the answer?
```

Lists – Join

- The last part is "join" – combining two or more list together.

```
# let mystr join into num
num = [3, 24, 13, 41, -50, 26, -17]
mystr = ['doctor','part','unique']
ns1 = num + mystr
print(ns1)
num.append(mystr)
print(num)
num.extend(mystr)
print(num)
```

Lists – Methods



Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

https://www.w3schools.com/python/python_lists_methods.asp

Tuples – Fundamentals

- Tuple is a very special data type in Python.
- To be honest, using tuple should consider twice because it is equipped the following characteristics:
    1) Ordered
    2) Unchangeable
    3) Allow duplicates

```
# make an experiment to prove it!
mytuple = (3, 24, 13, 41, -50, 26, -17, -50, 26, -17)
print(mytuple, mytuple[1])
mytuple[1] = 999 # can it work?
print(len(mytuple))
```

Tuples – Multi-type Tuples

- Some functionality in Tuple is just the same as that in List.

```
# different data types of tuples
tuple1 = ("apple", "banana", "cherry", "melon")
tuple2 = (1, 5, 7, 9, 3, 9, 3)
tuple3 = (True, False, False, True, False, True)
print(tuple1)
print(tuple2)
print(tuple3)
# multi-datatype tuples
tuple4 = ("abc", 56, 314, True, True, False, 40, "male")
```

Tuples – Indexing

- Indexing tuples …

```
# different data types of tuples
tuple1 = (1, 5, 7, 9, 3, 9, 3)
print(tuple1[2:])
print(tuple1[2:5])
print(tuple1[-2])
```

Tuples – Update

```
# add an element into the tuple
tuple1 = (1, 5, 7, 9, 3, 9, 3)
tuple2 = list(tuple1)
tuple2.append(1000)
print(tuple(tuple2))
# why do we need to
# transform into list at first?
X = ("apple", )
tuple1 += X
print(tuple1)
```

```
# remove an element from the
# tuple
Y = list(tuple1)
Y.remove("apple")
Y = tuple(Y)
```

Tuples – Unpack

- Due to the unchangeable nature of tuple, unpacking a tuple is very important.

```
# assign each tuple element for
# one variable
year1 = (1, 5, 7)
(joy, may, jon) = year1
print(joy)
print(may)
print(jon)
# we can also use asterisk (*) for unpacking
# the tuple; here, you need to observe the
# results of two examples and explain …
```

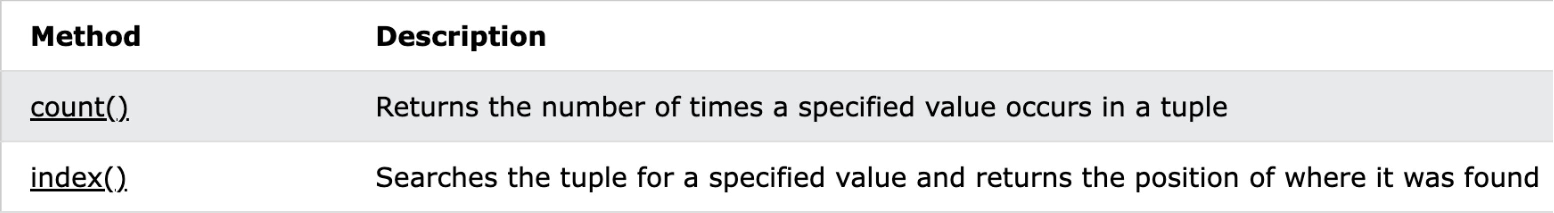
```
# example 1
year2 = (1, 5, 7, 9, 3, 9, 3)
(joy, may, *jon) = year2
print(joy)
print(may)
print(jon)
# example 2
(joy, *may, jon) = year2
print(joy)
print(may)
print(jon)
```

Tuples – Join Two or More Tuples & Methods

- As other data types, tuple also offers a cability of joint.

```
# join tuples – by using addition
year1 = (1, 5, 7)
year2 = (12, 52, 72)
print(year1 + year2)
# join tuples – by using multiplication
year3 = year1*2
print(year3)
```

- Tuple methods



Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Sets – Fundamentals

- A set is a collection which is *unordered*, *unchangeable**, and *unindexed*.
- **Set Items:** are unordered, unchangeable, and do not allow duplicate values.
- **Unordered:** means that the items in a set do not have a defined order. Set items can appear in a different order every time you use them and cannot be referred to by index or key.
- **Unchangeable:** Set items are unchangeable, meaning that we cannot change the items after the set has been created.

https://www.w3schools.com/python/python_sets.asp

Sets – Duplicated Values

- Due to the nature of set in Python, all elements in a set should be unique. Let’s do an experiment.

```
# duplicated problem in a set
subject = {'math', 'english', 'sociology', 'math', 'physics'}
print(subject)
# True or 1 and False or 0
txtset = {3.5, 1, 0, 'math', False, True}
print(txtset)
# what do you observe in the second example?
print(len(txtset))
```

Sets – Add

- We can add an element, a set, or a list into a set.

```
# add an element into the set by using addition
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject.add('russian')
print(subject)
# add a set into the set by using update
subject2 = {'chinese', 'korean'}
subject.update(subject2)
print(subject)
# add a list into the set by using update
subject2 = ['chinese', 'korean']
subject.update(subject2)
print(subject)
```

Sets – Remove

- If you want to remove an element from the set, then …

```
# remove an element from the set by using remove
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject.remove('russian')
print(subject)
# remove an element from the set by using discard
subject.discard('math')
print(subject)
# delete all elements from the set
subject.clear()
print(subject)
```

Sets – Join1

- Combine two or more sets together, you may …

```
# join an element from the set by using union
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject2 = {'chinese', 'korean'}
subject.union(subject2)
print(subject)
# join an element from the set by using update
subject.update(subject2)
print(subject)
```

Sets – Join2 (Keep ONLY the Duplicates)

- Combine two or more sets together, you may …

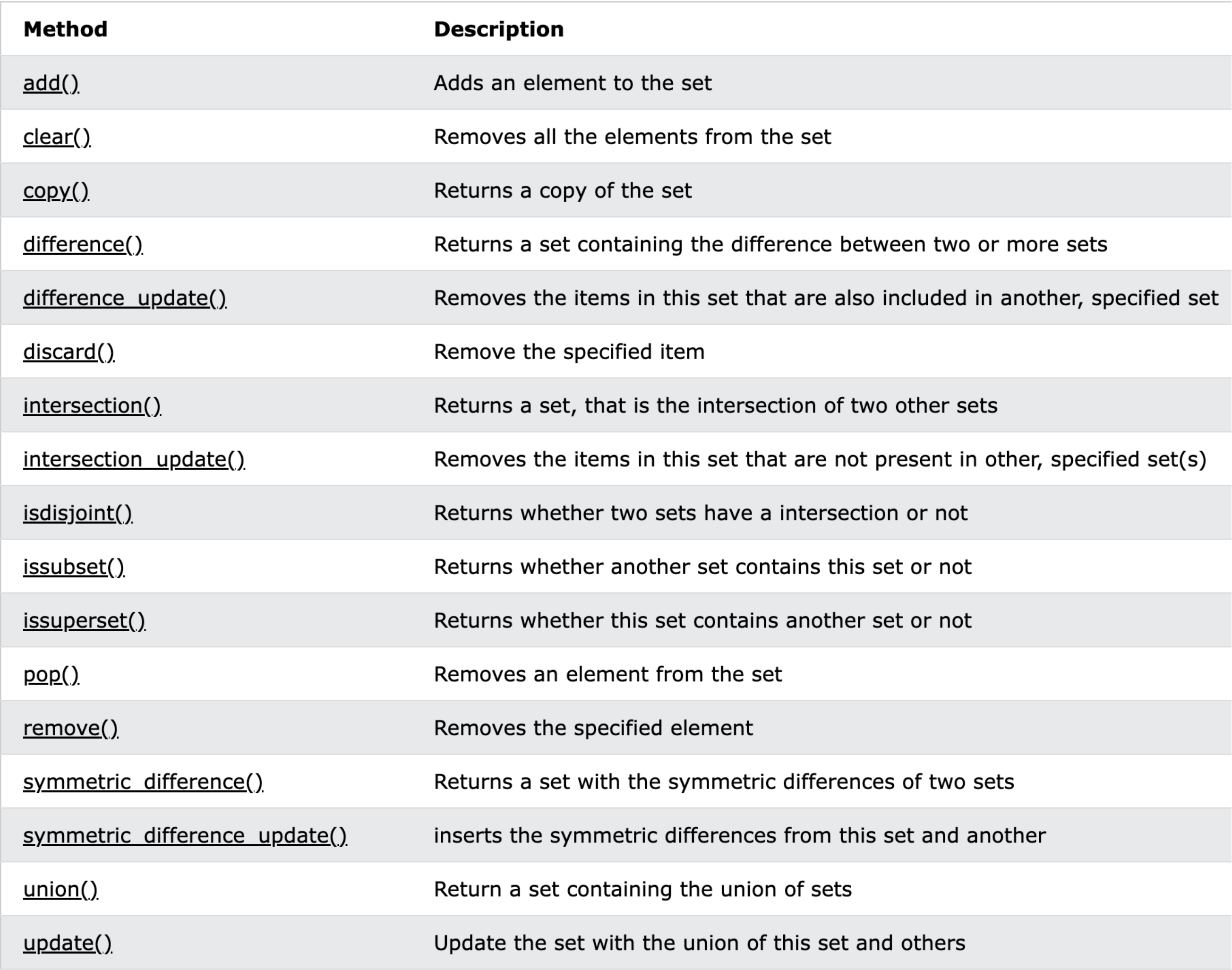
```
# union - keep only the items that are present in both sets
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject2 = {'sociology', 'math', 'chinese', 'korean'}
subject.intersection_update(subject2)
print(subject)
# merging two sets by using intersection
subject.intersection(subject2)
print(subject)
```

Sets – Join3 (But NOT the Duplicates)

- Combine two or more sets together, you may …

```
# union - keep only the items that are present in both sets
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject2 = {'sociology', 'math', 'chinese', 'korean'}
# keep only the elements that are NOT present in both sets
subject. symmetric_difference_update(subject2)
print(subject)
# contains only the elements that are NOT present in both sets
subject. symmetric_difference(subject2)
print(subject)
# try the following test
x = {1, True}
print(subject. symmetric_difference(x))
```

Set Methods



Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Dictionaries – Fundamentals

- Dictionary is also a powerful data type in Python; especially, one of the most common package, Pandas (or GeoPandas), has a useful class - dataframe, developed on the basis of dict.

```
# declare a dict
airport = {'air_name': 'TPE', 'Pax': 100}
print(airport)
print(airport['air_name'])
# duplicates are not allowed in dicts
airport2 = {'air_name': 'TPE', 'Pax': 100, 'Pax': 200}
print(airport2, len(airport)) # what does the length mean here?
```

Dictionaries – Index

- After declaration, again, we need to know how get the data.

```
airport = {'air_name': 'TPE', 'Pax': 100}
# get info of one attribute
print(airport['air_name'])
print(airport.get('air_name'))
# get all keys, values, and items
print(airport.keys(), '\n',airport.values() , '\n',airport.items()))
# add a key
airport['year'] = 1981
print(airport.keys())
```

Dictionaries – Change

- If you want to change or update the values in the dict, then …

```
airport = {'air_name': 'TPE', 'Pax': 100}
# change value in a dict by using direct indexing
airport['air_name'] = 'LHR'
# test if it changed
print(airport['air_name'])
# change value in a dict by using update
airport.update({'air_name' : 'KHH'})
# test if it changed
print(airport['air_name'])
```

Dictionaries – Add

- If you want to add new items into a dict, then …

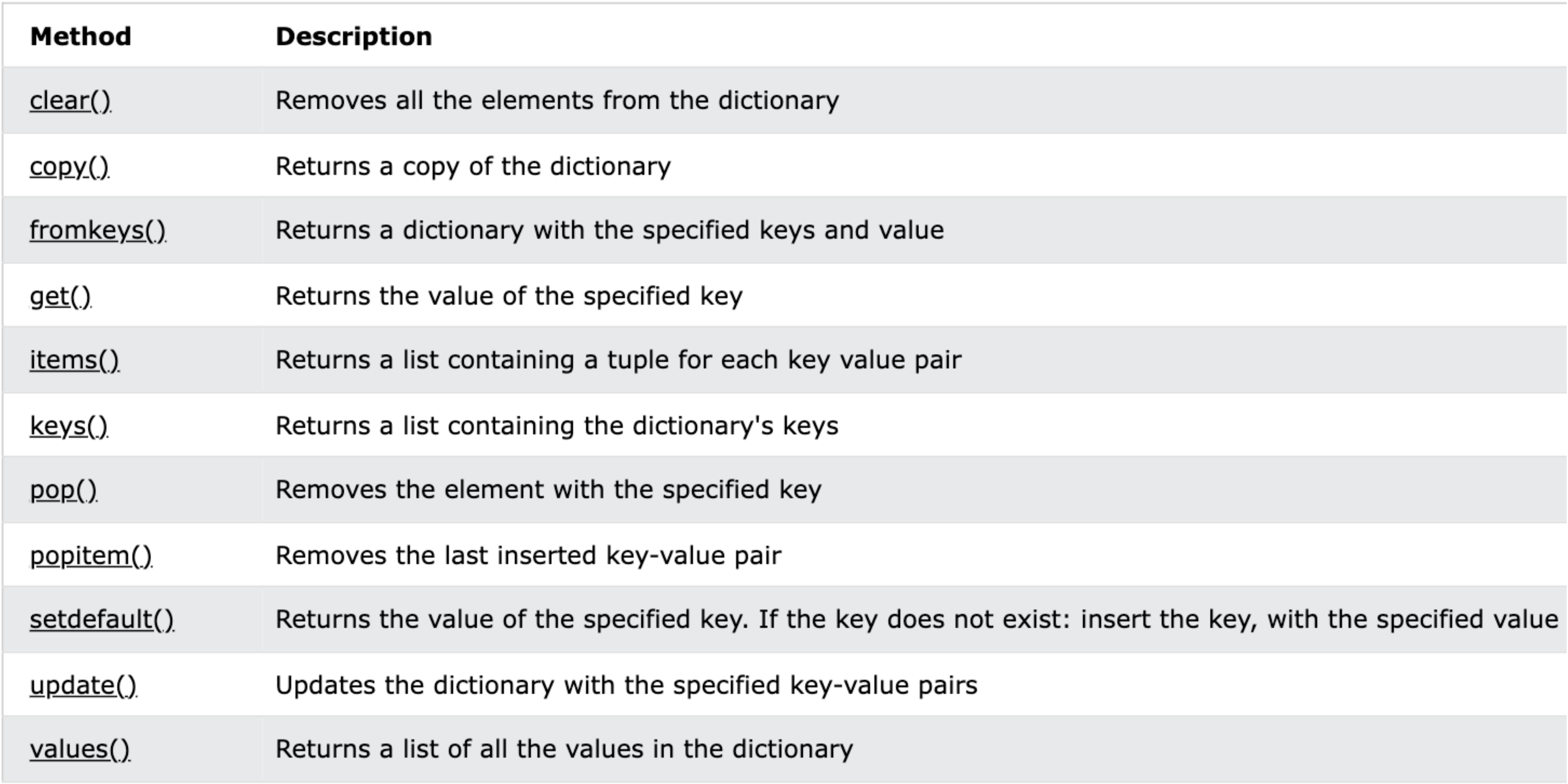
```
airport = {'air_name': 'TPE', 'Pax': 100}
# add value in a dict by using direct indexing
airport['year'] = 1981
# test if it added
print(airport)
# added value in a dict by using update
airport.update({'year' : 1981})
# test if it added
print(airport)
```

Dictionaries – Remove

- If you want to remove new items into a dict, then …

```
airport = {'air_name': 'TPE', 'Pax': 100, 'year': 1981}
# remove value in a dict with a key
airport.pop('year')
print(airport)
# remove value in a dict by using popitem
airport.popitem()
print(airport)
# remove value in a dict by using del (notice: re-declare dict again)
del airport['Pax']
print(airport) # you may try airport.clear()
```

Dictionaries – Methods



Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

CHUN-HSIANG CHAN (2023)

Question Time

- **Assignment:**
- **Download today's lab practice and upload to moodle.**
- **Thx**

The End

Thank you for your attention!

Email: chchan@ntnu.edu.tw
Web: toodou.github.io

SCAN ME

SCAN ME