L-4
Python Strings

ENGG1330
Computer Programming I

SCHOOL OF
COMPUTING &
DATA SCIENCE
The University of Hong Kong

L-3: What Your Survey Told Us

A String Can Describe a Webpage

What It Represents	Text Stored in a String	What We Can Do
Webpage (HTML)	`<h1>Hello, Ada!</h1>`	Build a page heading
Python program	`print('Hello, Ada!')`	Generate source code
App response (JSON)	`{"temperature": 24.5}`	Read weather data
DNA sequence	`ACGTGCAATG`	Find a pattern

A String Can Describe a Webpage

What It Represents	Text Stored in a String	What We Can Do
Webpage (HTML)	`<h1>Hello, Ada!</h1>`	Build a page heading
Python program	`print('Hello, Ada!')`	Generate source code
App response (JSON)	`{"temperature": 24.5}`	Read weather data
DNA sequence	`ACGTGCAATG`	Find a pattern

Strings carry structure, instructions, and data

A String Is a Sequence

PY main.py

```python
text = 'Python'
print(text)
print(len(text))
print(text[0])
print(len(''))
```

Download: lec_04_string_sequence.py

Strings Can Contain Unicode Text

PY main.py

```
1 text = 'café'
2 print(text)
3 print(len(text))
4 print(text[-1])
5 print('🍌')
```

Download: lec_04_unicode_text.py

Either Quote Style Can Delimit a String

PY main.py

```
1 print("Hello 'Python'")
2 print('Hello "Python"')
3 print('Hello \'Python\'')
```

Download: lec_04_quotation_marks.py

A Matching Quote Ends the String

main.py

```
1 print('Hello 'Python'')
```

Download: lec_04_quotation_error.py

Escape Sequences Represent Special Characters

Escape	Meaning
`\n`	New line
`\t`	Next tab stop
`\'`	Single quote
`\"`	Double quote
`\\`	Backslash

main.py

```
1 text = 'Hello\nWorld'
2 print(text)
3 print(repr(text))
```

Download: lec_04_escape_characters.py

`repr(text)` returns a string showing quotes and escape sequences

Newlines and Tabs Control the Layout

PY main.py

```
1 text = '*\t*\n**\t**\n***\t***'
2 print(text)
```

Download: lec_04_newlines_tabs.py

Triple Quotes Retain Line Breaks

PY main.py

```
1 text = """This
2 is
3 a
4 multiline string"""
5 print(text)
6 print(repr(text))
```

Download: lec_04_multiline_strings.py

Raw Strings Preserve Backslashes

PY main.py

```
1 print('C:\\some\name')
2 print(r'C:\some\name')
```

Download: lec_04_raw_strings.py

String Operations

ENGG1330
Computer Programming I

Concatenation Creates a New String

main.py

```
1 description = 'immutable'
2 message = 'Strings are ' + description + ' at all times'
3 print(message)
```

Download: lec_04_concatenation.py

Adjacent Literals Join Automatically

PY main.py

```
1 message = ('Vigorous writing is concise. '
2              'Every word should contribute.')
3 print(message)
4 first = 'Py'
5 second = 'thon'
6 print(first + second)
```

Download: lec_04_adjacent_literals.py

An Integer Repeats a String

PY main.py

```
1 print('Ha' * 3)
2 print(3 * 'un' + 'ium')
3 print(repr('Ha' * 0))
4 print(repr('Ha' * -2))
```

Download: lec_04_repetition.py

An Index Selects One Character

main.py

```
1 text = "You miss all shots you don't take"
2 for index in [0, -1, -10, 10]:
3     print(index, repr(text[index]))
```

Download: lec_04_indexing.py

An Index Selects One Character

`text[0]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e
-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

main.py

```
1 text = "You miss all shots you don't take"
2 for index in [0, -1, -10, 10]:
3     print(index, repr(text[index]))
```

Download: lec_04_indexing.py

An Index Selects One Character

`text[-1]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e
-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

main.py

```
1 text = "You miss all shots you don't take"
2 for index in [0, -1, -10, 10]:
3     print(index, repr(text[index]))
```

Download: lec_04_indexing.py

An Index Selects One Character

`text[-10]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e
-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

PY main.py

```
1 text = "You miss all shots you don't take"
2 for index in [0, -1, -10, 10]:
3     print(index, repr(text[index]))
```

Download: lec_04_indexing.py

An Index Selects One Character

`text[10]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e
-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

main.py

```
1 text = "You miss all shots you don't take"
2 for index in [0, -1, -10, 10]:
3     print(index, repr(text[index]))
```

Download: lec_04_indexing.py

A Loop Can Visit Characters Directly

main.py

```
1 text = 'dirk'
2 for index in range(len(text)):
3     if text[index] == 'i' or text[index] == 'u':
4         print('Found i or u')
5         break
6 for char in text:
7     if char == 'i' or char == 'u':
8         print('Found i or u')
9         break
```

Download: lec_04_character_loops.py

Robot Cheerleader

main.py

```
word = input('Word: ')
times = int(input('Enthusiasm (1-10): '))
for char in word:
    print('Give me a ' + char + '! ' + char)
print('What does that spell?')
print((word + '!!! ') * times)
```


Download: lec_04_robot_cheerleader.py

Membership Tests for a Substring

PY main.py

```
1 print('a' in 'program')
2 print('at' not in 'battle')
3 print('battle' in 'at')
```

Download: lec_04_membership.py

Membership Improves the Cheerleader

main.py

```
word = input('Uppercase word: ')
an_letters = 'AEFHILMNORSX'
for char in word:
    article = 'a'
    if char in an_letters:
        article = 'an'
    print('Give me ' + article + ' ' + char + '! ' + char)
print(word, '!!!')
```


Download: lec_04_robot_articles.py

A Slice Stops Before Its Stop Index

main.py

```
1 text = "You miss all shots you don't take"
2 print(text[13:17])
3 print(text[4:])
4 print(text[:3])
```

Download: lec_04_slicing.py

A Slice Stops Before Its Stop Index

`text[13:17]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e

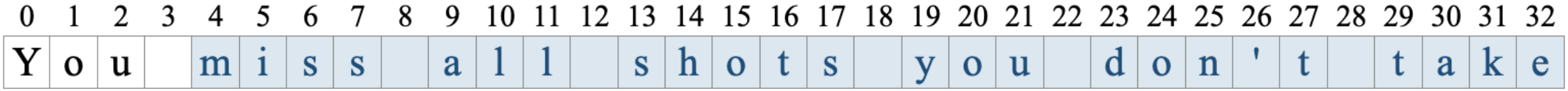
PY main.py

```
1 text = "You miss all shots you don't take"
2 print(text[13:17])
3 print(text[4:])
4 print(text[:3])
```

Download: lec_04_slicing.py

A Slice Stops Before Its Stop Index

`text[4:]`



main.py

```
1 text = "You miss all shots you don't take"
2 print(text[13:17])
3 print(text[4:])
4 print(text[:3])
```

Download: lec_04_slicing.py

A Slice Stops Before Its Stop Index

`text[:3]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e

main.py

```
1 text = "You miss all shots you don't take"
2 print(text[13:17])
3 print(text[4:])
4 print(text[:3])
```

Download: lec_04_slicing.py

Negative Indices Count from the End

PY main.py

```
1 text = "You miss all shots you don't take"
2 print(text[-4:])
3 print(text[-10:-5])
4 print(text[-10:])
```

Download: lec_04_negative_slices.py

Negative Indices Count from the End

`text[-4:]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e
-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

main.py

```
1 text = "You miss all shots you don't take"
2 print(text[-4:])
3 print(text[-10:-5])
4 print(text[-10:])
```

Download: lec_04_negative_slices.py

Negative Indices Count from the End

`text[-10:-5]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e
-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

main.py

```
1 text = "You miss all shots you don't take"
2 print(text[-4:])
3 print(text[-10:-5])
4 print(text[-10:])
```

Download: lec_04_negative_slices.py

Negative Indices Count from the End

`text[-10:]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e
-33	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

main.py

```
1 text = "You miss all shots you don't take"
2 print(text[-4:])
3 print(text[-10:-5])
4 print(text[-10:])
```

Download: lec_04_negative_slices.py

A Step Selects Spaced Characters

PY main.py

```
1 text = "You miss all shots you don't take"
2 print(text[1:10:1], text[1:10:2], sep=" | ")
3 print(text[1:10:3])
4 print(text[::3])
```

Download: lec_04_slice_step.py

A Step Selects Spaced Characters

`text[1:10:1]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e

main.py

```
1 text = "You miss all shots you don't take"
2 print(text[1:10:1], text[1:10:2], sep=" | ")
3 print(text[1:10:3])
4 print(text[::3])
```

Download: lec_04_slice_step.py

A Step Selects Spaced Characters

`text[1:10:2]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e

main.py

```
1 text = "You miss all shots you don't take"
2 print(text[1:10:1], text[1:10:2], sep=" | ")
3 print(text[1:10:3])
4 print(text[::3])
```

Download: lec_04_slice_step.py

A Step Selects Spaced Characters

`text[1:10:3]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e

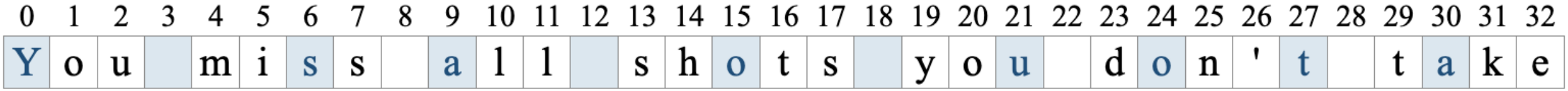
main.py

```
1 text = "You miss all shots you don't take"
2 print(text[1:10:1], text[1:10:2], sep=" | ")
3 print(text[1:10:3])
4 print(text[::3])
```

Download: lec_04_slice_step.py

A Step Selects Spaced Characters

`text[::3]`



main.py

```
1 text = "You miss all shots you don't take"
2 print(text[1:10:1], text[1:10:2], sep=" | ")
3 print(text[1:10:3])
4 print(text[::3])
```

Download: lec_04_slice_step.py

A Negative Step Moves Backwards

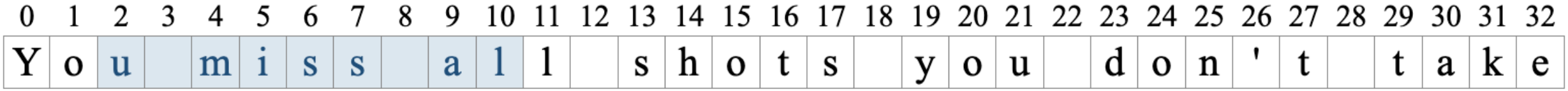
main.py

```
1 text = "You miss all shots you don't take"
2 print(text[10:1:-1])
3 print(text[10:1:-2])
4 print(text[::-1])
```

Download: lec_04_reverse_slices.py

A Negative Step Moves Backwards

`text[10:1:-1]`



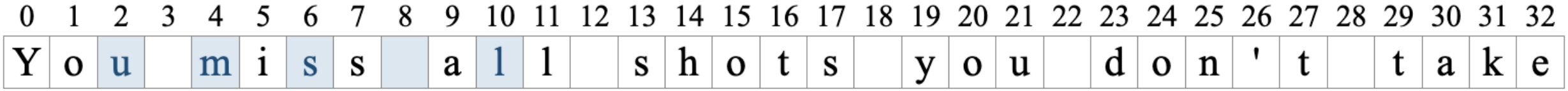
main.py

```
1 text = "You miss all shots you don't take"
2 print(text[10:1:-1])
3 print(text[10:1:-2])
4 print(text[::-1])
```

Download: lec_04_reverse_slices.py

A Negative Step Moves Backwards

`text[10:1:-2]`



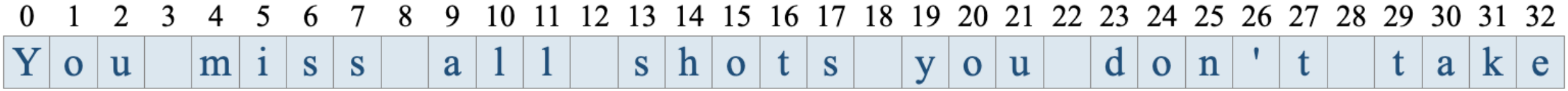
main.py

```
1 text = "You miss all shots you don't take"
2 print(text[10:1:-1])
3 print(text[10:1:-2])
4 print(text[::-1])
```

Download: lec_04_reverse_slices.py

A Negative Step Moves Backwards

`text[::-1]`



main.py

```
1 text = "You miss all shots you don't take"
2 print(text[10:1:-1])
3 print(text[10:1:-2])
4 print(text[::-1])
```

Download: lec_04_reverse_slices.py

Slicing Can Reach Beyond the String

PY main.py

```
1 text = 'Python'
2 print(text[:100])
3 print(repr(text[100:]))
```

Download: lec_04_slice_boundaries.py

An Index Must Select an Existing Character

main.py

```
1 text = 'Python'
2 print(text[100])
```

Download: lec_04_index_boundary_error.py

Strings Compare at the First Difference

main.py

```
1 print('Apple' == 'Apple')
2 print('Apple' != 'Apple')
3 print('Apple' < 'Apple')
4 print('Apple' <= 'Apple')
5 print('cat' < 'car')
6 print('HKU' < 'CUHK')
```

Download: lec_04_string_comparison.py

String Functions and Methods

ENGG1330
Computer Programming I

Finding the Operation You Need

Question	Reference
A function such as len or ord	Built-in Functions
An operation such as upper or split	String Methods

A Function Returns a Result to Its Caller

main.py

```python
def is_palindrome(text):
    return text == text[::-1]

print(is_palindrome('test'))
print(is_palindrome('madam'))
print(is_palindrome('Anna'))
```

Download: lec_04_palindrome.py

A Function Returns a Result to Its Caller

Argument: `'madam'` Parameter: `text` Return value: `True`

main.py

```python
def is_palindrome(text):
    return text == text[::-1]

print(is_palindrome('test'))
print(is_palindrome('madam'))
print(is_palindrome('Anna'))
```

Download: lec_04_palindrome.py

A Function Returns a Result to Its Caller

`'Anna'` Case-sensitive comparison `'annA'`

main.py

```python
def is_palindrome(text):
    return text == text[::-1]

print(is_palindrome('test'))
print(is_palindrome('madam'))
print(is_palindrome('Anna'))
```

Download: lec_04_palindrome.py

Returning a Value and Printing Output

main.py

```
1  def double(number):
2      return number * 2
3
4  def show_double(number):
5      print(number * 2)
6
7  print(double(5) + 1)
8  result = show_double(5)
9  print(result)
```

Download: lec_04_return_versus_print.py

A Loop Can Check the Same Palindrome

PY main.py

```
1  def is_palindrome(text):
2      index = 0
3      for char in reversed(text):
4          if char != text[index]:
5              return False
6          index += 1
7      return True
8
9  print(is_palindrome('madam'))
10 print(is_palindrome('Anna'))
```

Download: lec_04_palindrome_loop.py

ord Gives a Character's Code Point

PY main.py

```
1 print(ord('a'), ord('A'))
2 print(ord('1'), ord('2'))
3 print(ord('é'))
```

Download: lec_04_code_points.py

chr Converts a Code Point to Text

PY main.py

```
1 print(chr(65))
2 print(chr(ord('é')))
3 for number in range(65, 91):
4     print(chr(number), end=' ')
5 print()
```

Download: lec_04_characters_from_codes.py

String Comparison Is Case Sensitive

PY main.py

```
1 print('HKU' == "HKU")
2 print('HKU' == 'HkU')
3 print(ord('K'), ord('k'))
4 print('HKU' < 'HkU')
```

Download: lec_04_case_comparison.py

Useful String Methods

Method	Purpose
`capitalize()`	Capitalise the first character; lowercase the rest
`upper(), lower()`	Change case
`split()`	Split into a list of strings
`find()`	Find a substring, or return −1
`count()`	Count non-overlapping occurrences
`isalnum(), isalpha()`	Test alphanumeric or alphabetic text
`isdecimal()`	Test for decimal digit characters
`strip()`	Remove leading and trailing whitespace
`replace()`	Return text with replacements

Case Methods Return a New String

PY main.py

```
1 name = 'DiRk'
2 print(name.capitalize())
3 print(name.upper())
4 print(name.lower())
5 print(name)
```

Download: lec_04_case_methods.py

Searching and Splitting Text

PY main.py

```
1 text = 'red blue red'
2 print(text.split())
3 print(text.find('blue'))
4 print(text.find('green'))
5 print(text.count('red'))
```

Download: lec_04_search_and_split.py

Check the Start or End of a String

PY main.py

```
1 filename = 'report.csv'
2 print(filename.endswith('.csv'))
3 address = 'https://hku.hk'
4 print(address.startswith('https://'))
5 print('report.csv.bak'.endswith('.csv'))
```

Download: lec_04_prefix_suffix.py

Character Tests Return Booleans

main.py

```
print('HKU1330'.isalnum())
print('HKU'.isalpha())
print('1330'.isdecimal())
print('-3'.isdecimal())
print('3.5'.isdecimal())
print(''.isalpha())
```

Download: lec_04_character_tests.py

Removing Whitespace and Replacing Text

PY main.py

```
1 text = '  Hello HKU  '
2 print(repr(text.strip()))
3 print(repr(text.replace('HKU', 'Python')))
4 print(repr(text))
```

Download: lec_04_strip_replace.py

Immutability and F-Strings

ENGG1330
Computer Programming I

A String's Characters Cannot Be Reassigned

main.py

```
1 text = 'change me'
2 text[0] = 'a'
```

Download: lec_04_immutable_assignment.py

A String's Characters Cannot Be Deleted

PY main.py

```
1 text = 'delete me'
2 del text[1]
```

Download: lec_04_immutable_deletion.py

Slicing Can Omit a Character

main.py

```
1 text = "You miss all shots you don't take"
2 changed = text[:29] + text[30:]
3 print(changed)
4 print(text)
```

Download: lec_04_remove_character.py

Slicing Can Omit a Character

Index 29: t

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e

PY main.py

```
1 text = "You miss all shots you don't take"
2 changed = text[:29] + text[30:]
3 print(changed)
4 print(text)
```

Download: lec_04_remove_character.py

Slicing Can Replace a Character

PY main.py

```
1 text = "You miss all shots you don't take"
2 changed = text[:29] + 'f' + text[30:]
3 print(changed)
4 print(text)
```

Download: lec_04_replace_character.py

Slicing Can Replace a Character

Index 29: t

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Y	o	u		m	i	s	s		a	l	l		s	h	o	t	s		y	o	u		d	o	n	'	t		t	a	k	e

main.py

```
1 text = "You miss all shots you don't take"
2 changed = text[:29] + 'f' + text[30:]
3 print(changed)
4 print(text)
```

Download: lec_04_replace_character.py

A List Can Be Edited and Joined

PY main.py

```
1 text = 'Test'
2 letters = list(text)
3 letters[1] = 'E'
4 changed = ''.join(letters)
5 print(letters)
6 print(changed)
7 print(text)
```

Download: lec_04_list_join.py

Split and Join Clean Up Extra Spaces

PY main.py

```
1 name = '  Ada   Lovelace  '
2 words = name.split()
3 cleaned = ' '.join(words)
4 print(words)
5 print(repr(cleaned))
```

Download: lec_04_clean_whitespace.py

F-Strings Insert Expression Values

main.py

```
1 name = 'Dirk'
2 student = {'name': 'Dirk', 'age': 21}
3 print(f'My name is {name}')
4 print(f"{student['name']} is {student['age']} years old")
```

Download: lec_04_f_strings.py

Alignment Uses a Minimum Field Width

PY main.py

```
1 name = 'Dirk'
2 print(f'|{name:<20}|')
3 print(f'|{name:^20}|')
4 print(f'|{name:>20}|')
5 print(f'|{name:0>20}|')
```

Download: lec_04_alignment.py

Zero Padding Keeps Numbers Aligned

PY main.py

```
1 for number in [5, 9, 10, 15]:
2     print(f'{number:03}')
```

Download: lec_04_zero_padding.py

Precision Controls Decimal Places

PY main.py

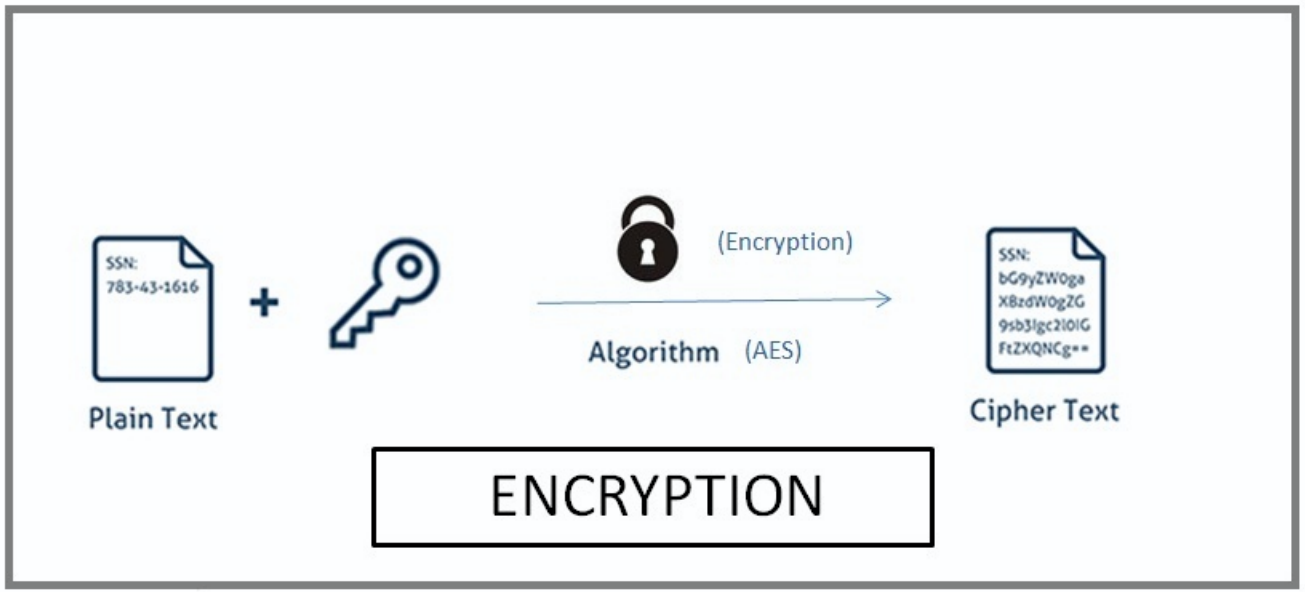
```
1 pi = 3.14159265
2 print(f'Pi is {pi:.4f}')
3 print(pi)
```

Download: lec_04_precision.py

Encryption and Sorting

ENGG1330
Computer Programming I

Encryption Uses a Key



Plain text and a key produce cipher text

A Caesar Cipher Shifts the Alphabet

Key = 11

Plain	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k

Each letter maps to the letter below it

A Caesar Cipher Shifts the Alphabet

Key = 11

Plain	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k

Each letter maps to the letter below it

```
h e l l o  →  s p w w z
```

The Inputs Define the Encryption Task

main.py

```
1 plain_text = 'hello'
2 key = 11
3 print(plain_text)
4 print(key)
```

Download: lec_04_caesar_inputs.py

`plain_text`

The text to transform

`key`

How far to shift each letter

Use lowercase English letters and a key from 0 to 25

Two Slices Rotate the Alphabet

main.py

```
1 plain_text = 'hello'
2 key = 11
3 alphabet = 'abcdefghijklmnopqrstuvwxyz'
4 shifted = alphabet[key:] + alphabet[:key]
5 print(alphabet)
6 print(shifted)
```

```
alphabet[11:]
lmnopqrstuvwxyz

alphabet[:11]
abcdefghijk
```

Join the suffix and prefix

Download: lec_04_caesar_alphabet.py

A Loop Builds the Cipher Text

main.py

```
1 plain_text = 'hello'
2 key = 11
3 alphabet = 'abcdefghijklmnopqrstuvwxyz'
4 shifted = alphabet[key:] + alphabet[:key]
5 cipher_text = ''
6 for char in plain_text:
7     index = ord(char) - ord('a')
8     cipher_text += shifted[index]
9 print(cipher_text)
```

Download: lec_04_caesar_encrypt.py

A Loop Builds the Cipher Text

main.py

```
1 plain_text = 'hello'
2 key = 11
3 alphabet = 'abcdefghijklmnopqrstuvwxyz'
4 shifted = alphabet[key:] + alphabet[:key]
5 cipher_text = ''
6 for char in plain_text:
7     index = ord(char) - ord('a')
8     cipher_text += shifted[index]
9 print(cipher_text)
```

Variable	Value
char	'h'
index	7
shifted[index]	's'
cipher_text	's'

Download: lec_04_caesar_encrypt.py

A Loop Builds the Cipher Text

main.py

```
1 plain_text = 'hello'
2 key = 11
3 alphabet = 'abcdefghijklmnopqrstuvwxyz'
4 shifted = alphabet[key:] + alphabet[:key]
5 cipher_text = ''
6 for char in plain_text:
7     index = ord(char) - ord('a')
8     cipher_text += shifted[index]
9 print(cipher_text)
```

Variable	Value
char	'e'
index	4
shifted[index]	'p'
cipher_text	'sp'

Download: lec_04_caesar_encrypt.py

A Loop Builds the Cipher Text

main.py

```
1 plain_text = 'hello'
2 key = 11
3 alphabet = 'abcdefghijklmnopqrstuvwxyz'
4 shifted = alphabet[key:] + alphabet[:key]
5 cipher_text = ''
6 for char in plain_text:
7     index = ord(char) - ord('a')
8     cipher_text += shifted[index]
9 print(cipher_text)
```

Variable	Value
char	'l'
index	11
shifted[index]	'w'
cipher_text	'spw'

Download: lec_04_caesar_encrypt.py

A Loop Builds the Cipher Text

main.py

```
1 plain_text = 'hello'
2 key = 11
3 alphabet = 'abcdefghijklmnopqrstuvwxyz'
4 shifted = alphabet[key:] + alphabet[:key]
5 cipher_text = ''
6 for char in plain_text:
7     index = ord(char) - ord('a')
8     cipher_text += shifted[index]
9 print(cipher_text)
```

Variable	Value
char	'l'
index	11
shifted[index]	'w'
cipher_text	'spww'

Download: lec_04_caesar_encrypt.py

A Loop Builds the Cipher Text

PY main.py

```
1 plain_text = 'hello'
2 key = 11
3 alphabet = 'abcdefghijklmnopqrstuvwxyz'
4 shifted = alphabet[key:] + alphabet[:key]
5 cipher_text = ''
6 for char in plain_text:
7     index = ord(char) - ord('a')
8     cipher_text += shifted[index]
9 print(cipher_text)
```

Variable	Value
char	'o'
index	14
shifted[index]	'z'
cipher_text	'spwwz'

Download: lec_04_caesar_encrypt.py

Trying All 26 Keys

main.py

```
1  alphabet = 'abcdefghijklmnopqrstuvwxyz'
2  cipher_text = 'spwwz'
3  for key in range(26):
4      shifted = alphabet[-key:] + alphabet[:-key]
5      plain_text = ''
6      for char in cipher_text:
7          plain_text += shifted[ord(char) - ord('a')]
8      print(f'{key:02}:{plain_text}', end='  ')
```


Download: lec_04_caesar_attack.py

A Loop Can Collect Names

PY main.py

```
1  names = []
2  while True:
3      new_name = input('Name (blank ends): ')
4      if new_name == '':
5          break
6      names.append(new_name)
7      print(names)
```

Download: lec_04_collect_names.py

Each Insertion Keeps the List Sorted

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

Download: lec_04_sorted_insertion.py

Find the first name that is greater than the new name

Insert immediately before it

If none is greater, insert at the end

The First Name Enters an Empty List

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

Download: lec_04_sorted_insertion.py

The First Name Enters an Empty List

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKU'`

pos = `0`

Empty list

Begin with an empty list

Download: lec_04_sorted_insertion.py

The First Name Enters an Empty List

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKU'`

pos = `0`

Empty list

$0 < 0$ is False

Download: lec_04_sorted_insertion.py

The First Name Enters an Empty List

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKU'`

pos = `0`

Empty list

Skip the while body

Download: lec_04_sorted_insertion.py

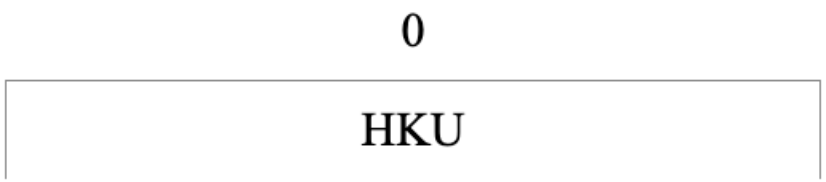
The First Name Enters an Empty List

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKU'`

pos = `0`



Insert at position 0

Download: lec_04_sorted_insertion.py

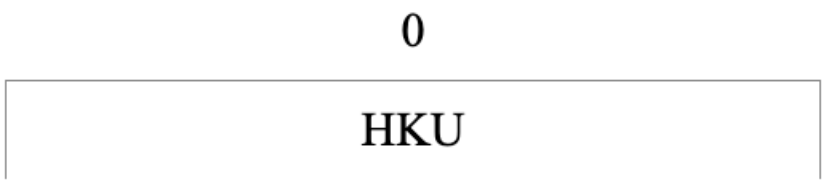
The First Name Enters an Empty List

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKU'`

pos = `0`



Print the updated list

Download: lec_04_sorted_insertion.py

A Smaller Name Goes Before HKU

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

Download: lec_04_sorted_insertion.py

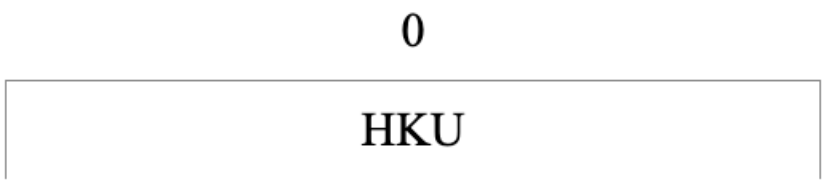
A Smaller Name Goes Before HKU

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'CUHK'`

pos = `0`



Begin the next iteration

Download: lec_04_sorted_insertion.py

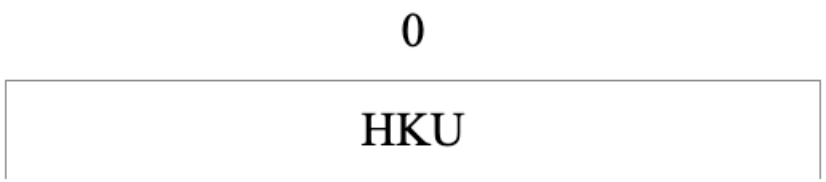
A Smaller Name Goes Before HKU

PY main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'CUHK'`

pos = `0`



0 < 1 is True

Download: lec_04_sorted_insertion.py

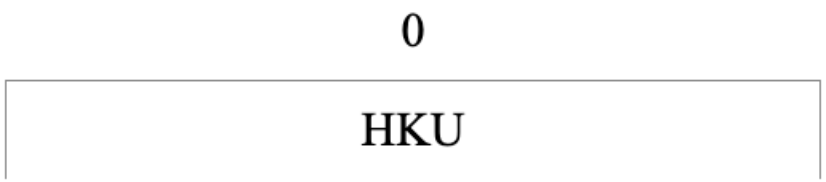
A Smaller Name Goes Before HKU

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'CUHK'`

pos = `0`



'HKU' > 'CUHK' is True

Download: lec_04_sorted_insertion.py

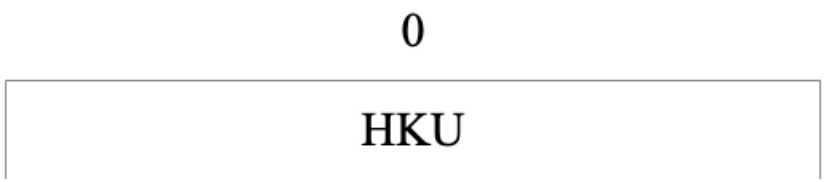
A Smaller Name Goes Before HKU

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'CUHK'`

pos = `0`



break leaves the while loop

Download: lec_04_sorted_insertion.py

A Smaller Name Goes Before HKU

main.py

```python
names = []
for new_name in ['HKU', 'CUHK', 'HKUST']:
    pos = 0
    while pos < len(names):
        if names[pos] > new_name:
            break
        pos += 1
    names.insert(pos, new_name)
    print(names)
```

Download: lec_04_sorted_insertion.py

New name: `'CUHK'`

pos = `0`

Insert before HKU

A Smaller Name Goes Before HKU

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'CUHK'`

pos = `0`

Print the updated list

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = `0`

0	1
CUHK	HKU

Reset pos for the new name

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = `0`

0	1
CUHK	HKU

0 < 2 is True

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = `0`

0	1
CUHK	HKU

'CUHK' > 'HKUST' is False

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1  names = []
2  for new_name in ['HKU', 'CUHK', 'HKUST']:
3      pos = 0
4      while pos < len(names):
5          if names[pos] > new_name:
6              break
7          pos += 1
8      names.insert(pos, new_name)
9      print(names)
```

New name: `'HKUST'`

pos = 1

0	1
CUHK	HKU

Advance to the next position

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = 1

0	1
CUHK	HKU

1 < 2 is True

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = 1

0	1
CUHK	HKU

'HKU' > 'HKUST' is False

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = 2

0	1
CUHK	HKU

Advance again

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = 2

0	1
CUHK	HKU

2 < 2 is False

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1  names = []
2  for new_name in ['HKU', 'CUHK', 'HKUST']:
3      pos = 0
4      while pos < len(names):
5          if names[pos] > new_name:
6              break
7          pos += 1
8      names.insert(pos, new_name)
9      print(names)
```

New name: `'HKUST'`

pos = 2

0	1	2
CUHK	HKU	HKUST

Insert at the end

Download: lec_04_sorted_insertion.py

A Larger Name Goes at the End

main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     pos = 0
4     while pos < len(names):
5         if names[pos] > new_name:
6             break
7         pos += 1
8     names.insert(pos, new_name)
9     print(names)
```

New name: `'HKUST'`

pos = 2

0	1	2
CUHK	HKU	HKUST

Print the updated list

Download: lec_04_sorted_insertion.py

sorted Returns a New Sorted List

PY main.py

```
1 names = []
2 for new_name in ['HKU', 'CUHK', 'HKUST']:
3     names.append(new_name)
4 ordered = sorted(names)
5 print(ordered)
6 print(names)
```

Download: lec_04_builtin_sorted.py

Working with Strings

Select	Index, slice, or iterate
Check	Compare, search, or test characters
Construct	Concatenate, join, or replace
Present	Format values with f-strings

A string operation can return a new value
without changing the original string

L-4 Survey

Complete the full survey during the lecture

Coding Challenge

Work independently on the L-4 challenges in Ed