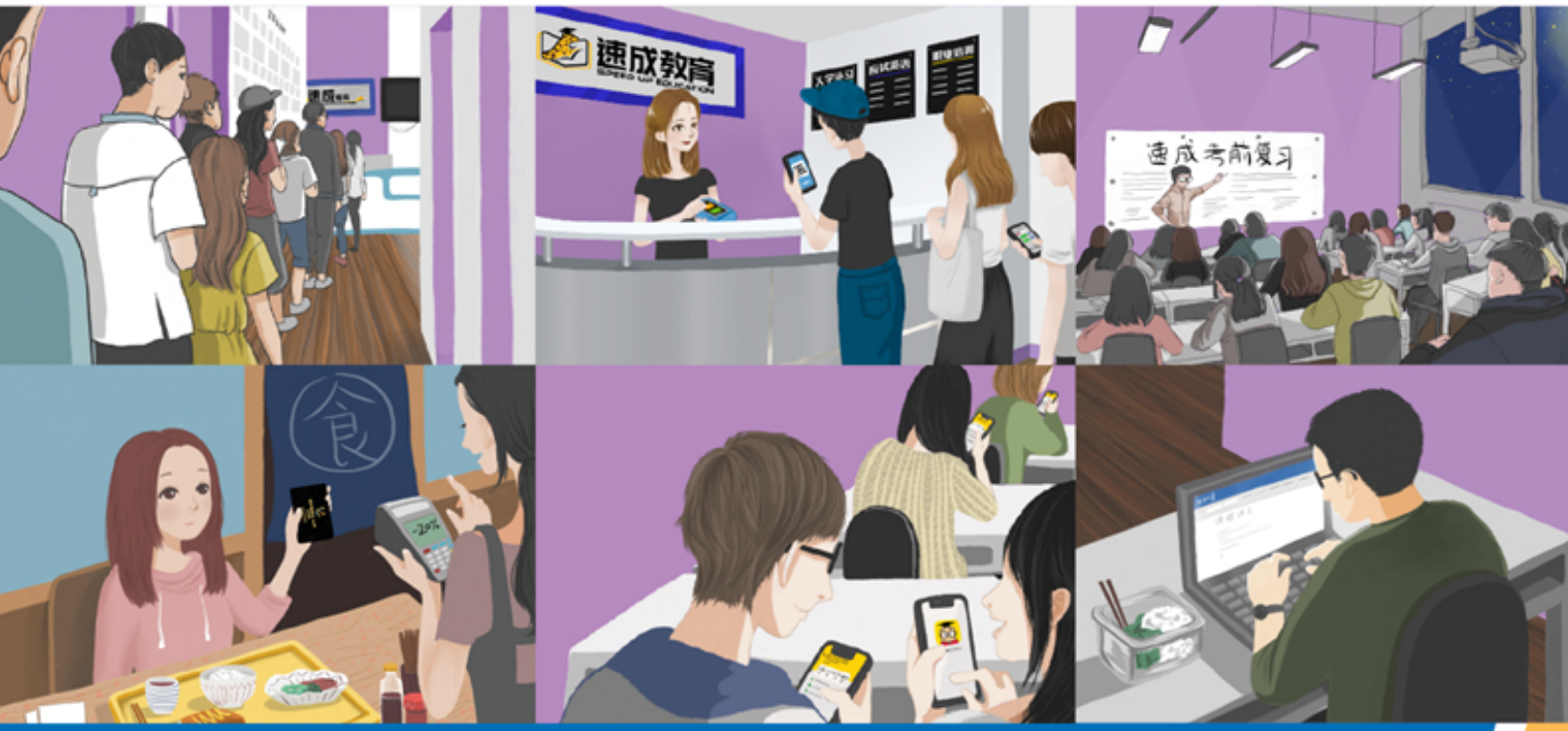
速成教育
SPEED UP EDUCATION



CSCA08

Introduction to Computer Science I

导师： **Vincent**

UTSC Week 05 | 2025/2/6

Disclaimer

This study handout is provided by Speed Up Education and its affiliated tutors. The study handout seeks to support your study process and should be used as a complement, not substitute to the university course material, lecture notes, problem, sets, past tests and other available resources.

The study handout is distributed for free to the students participating in Speed Up Education classes and is not for sale or other commercial uses whatsoever. We kindly ask you to refrain from copying or selling in part or in whole any information provided in the handout.

Thanks for choosing Speed Up Education. We wish you the best of luck in your exam.

While Loops

Assume we want to print 'I hate csca08' ten times, we can repeat the code ten times:

```
print('I hate csca08')
print('I hate csca08')
print('I hate csca08')
...
...
print('I hate csca08')
```

10 次

A better solution will be using **while loop**

```
x = 0
while x < 10:
    print('I hate csca08')
    x += 1
```

While loop is often used when you **don't know** how many times you want to iterate.

```
n = input('Please enter a number: ')

while not n.isdigit():
    print('Invalid input...')
    n = input('Please enter a number: ')

n = int(n)
# do something with n ...
```

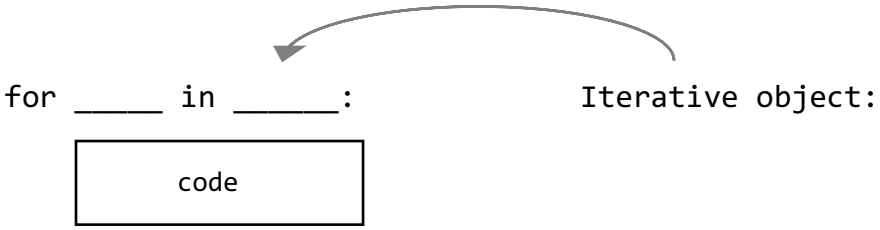
We can also iterate over a collection and access **each item** **one-at-a-time**.

```
word = 'csca08'
i = 0
while i < len(word):
    print('The character at index ' + str(i) + ' is: ' + word[i])
    x += 1
```

```
The character at index 0 is: c
The character at index 1 is: s
The character at index 2 is: c
The character at index 3 is: a
The character at index 4 is: 0
The character at index 5 is: 8
```

For Loops

Syntax



Example1

```
s = "CSCA08"
for ch in s:
    print("hello " + ch)
    print("bye")
print("Done")
```

Output:

```
hello C
bye
hello S
bye
hello C
bye
hello A
bye
hello 0
bye
hello 8
bye
Done
```

Example2

```
s = "MATA31"
for ch in s:
    if ch in "aeiouAEIOU":
        print("Vowel")
    elif ch.isdigit():
        print("Number")
    else:
        print("Letter")
print("Done")
```

Output:

```
Letter
Vowel
Letter
Vowel
Number
Number
Done
```

Range

range(start, end, step) *Creates a* ***collection*** *of* ***integers*** *(int)*

```
# same as slicing

>>> list(range(1, 9, 3))
[1, 4, 7]
```

Examples

```
r = range(1, 9, 3)
for x in r:
    print(x)
```

```
>>> 1
    4
    7
```

```
for x in range(1, 9, 3):
    print(x)
```

```
>>> 1
    4
    7
```

```
for x in range(1, 9, 1):
    print(x)
```

```
>>> 1
    2
    3
    4
    5
    6
    7
    8
```

```
for x in range(1, 9):
    print(x)
```

```
>>> 1
    2
    3
    4
    5
    6
    7
    8
```

```
for x in range(9):
    print(x)
```

```
>>> 0
    1
    2
    3
    4
    5
    6
    7
    8
```

```
for x in range(8, 1, -1):
    print(x)
```

```
>>> 8
    7
    6
    5
    4
    3
    2
```

The following codes are Equivalent

```
s = 'csca08'

for ch in s:
    print(ch)
```

```
s = 'csca08'

for i in range(0, len(s), 1):
    print(s[i])
```

```
s = 'csca08'

i = 0
while i < len(s):
    print(s[i])
    i += 1
```

題型 1: find it or not ?

```
def no_vowels(text: str) -> bool:
    '''Return True if and only if text does not contain any vowels.
    Vowels are a, e, i, o, and u.
    >>> no_vowels('xyz')
    True
    >>> no_vowels('Vincent')
    False
    '''

    # 試著用 for loop




    # 試著用 while loop
```

```
def contains_number(s: str) -> bool:
    """Return True iff s contains a numeric character.
    >>> contains_lowercase_vowels("csca08")
    True
    >>> contains_lowercase_vowels("vincent")
    False
    """
```

```
def has_letter_cases(s: str) -> bool:
    """Return True if and only if s contains at least one lowercase letter and at
    least one uppercase letter.

    >>> has_letter_cases('abcDEF')
    True
    >>> has_letter_cases('abc123')
    False
    >>> has_letter_cases('ABCXYZ')
    False
    """
```

```
def is_valid_phone_number(s: str) -> bool:
    """Return True iff s is a valid phone number.
    For our purposes, a phone number is a string of 10 or more characters, and
    contains only digits, spaces, and dashes ('-').

    >>> is_valid_phone_number('416 123-4567')
    True
    >>> is_valid_phone_number('416 EAT-CAKE')
    False
    >>> is_valid_phone_number('44 1908 640404')
    True
    >>> is_valid_phone_number('416 978')
    False
    """
```

```
def has_3_consecutive_digits(s: str) -> bool:
    """Return True if and only if s has 3 digits that appear consecutively
    (i.e., next to each other) in s, and False otherwise.

    >>> has_3_consecutive_digits('abc135xyz')
    True
    >>> has_3_consecutive_digits('a1b2c3')
    False
    """
```

```
def is_palindrome(word: str) -> bool:
    """Return True if and only if word is a palindrome
    i.e. if word reads the same backward and forward.

    >>> is_palindrome('racecar')
    True
    >>> is_palindrome('letter')
    False
    """
```

```
def pair_back_and_front(word: str) -> str:
    """Return a string that consists of every pair of characters taken from the
    back and front of word, i.e. the new string would consist of the last and
    first characters of word, followed by the second last and second characters
    of word, and so on, until the midpoint of word is reached.

    >>> pair_back_and_front('computer')
    'rceotmup'
    >>> pair_back_and_front('first')
    'tfsir'
    """
```

題型 2: Accumulate

```
def count_vowels(s: str) -> int:
    """Return the number of vowels in s.

    >>> count_vowels("Vincent")
    2
    >>> count_vowels("I like programming")
    6
    """
```

```
def count_alphanumeric(s: str) -> float:
    """Return the percentage of characters in s that are alphanumeric
    (letters or digits)
    The percentage should be between 0.0 and 1.0

    Precondition: len(s) >= 1

    >>> count_aplhanumeric("CSCA08")
    1.0
    >>> count_aplhanumeric("I love A08")
    0.8
    >>> count_aplhanumeric("!!!!")
    0.0
    """
```

```
def has_even_num_of_char(s: str, ch: str) -> bool:
    """Return True iff s contains an even number of the character ch.

    Precondition: s contains at least one occurrence of ch

    >>> has_even_num_of_char('hello', 'l')
    True
    >>> has_even_num_of_char('hello', 'e')
    False
    """
```

```
def get_percent(s: str, ch: str) -> float:
    """Return the percentage of characters in s that are equal to ch.
    The percentage should be between 0.0 and 1.0.

    Precondition: len(s) >= 1 and len(ch) == 1

    >>> get_percent('hello', 'l')
    0.4
    >>> get_percent('Testing 123', 'z')
    0.0
    >>> get_percent('Valentines', 'a')
    0.1
    """
```

```
def num_upper_digits_same(s: str) -> bool:
    """Return True iff s contains the same number of uppercase letters as digits.

    >>> num_upper_digits_same("CS 08")
    True
    >>> num_upper_digits_same("COMPUTER SCIENCE A08")
    False
    >>> num_upper_digits_same("apple")
    True
    """
```

```
def remove_vowel(s: str) -> str:
    """Return a copy of s but with all the vowel removed.

    >>> remove_vowel("Vincent")
    'Vncnt'
    >>> remove_vowel("I like Programming")
    ' lk Prgrmmng'
    """
```

```
def extract_digit(s: str) -> str:
    """Return a new string that contains only the digit character from s

    >>> extract_digit("csca08 4.0")
    '0840'
    """
```

```
def digitize_phrase(s: str) -> str:
    """Return a new string that consists of '0's and '1's
    For every even digit in s, add a '0' to the new string
    For every odd digit in s, add a '1' to the new string

    >>> digit_phrase("csca08 4.0 123")
    '0000101'
    '''
```

```
def remove_even_digit(n: int) -> int:
    """Return a new integer but with all the even number digit removed
    Precondition: n contains at least one odd digit

    >>> remove_even_digit(5201314)
    5131
    """
```

```
def shuffle(s: str) -> str:
    """Return a new string that contains all lowercase letters from s followed
    by all digits from s. The characters in the new string must appear in the
    same order as in s.

    Precondition: s contains only lowercase letters and/or digits

    >>> shuffle('416toronto')
    'toronto416'
    >>> shuffle('416yyz780yeg')
    'yyzyeg416780'
    """
```

```
def find_max(text: str) -> int:
    """return the max digit in the given text.

    Precondition:
        text must contain at least one digit character
.
    >>> find_max('my number is 647-719-1023')
    9
    >>> find_max('csca08 4.0')
    8
    """
```

```
def clean_text(text: str) -> str:
    """Return text with all leading non-alphabetic characters and all
    Trailing non-alphanumeric characters removed.

    >>> clean_text('  ? 42csca08:    !')
    'csca08'
    >>> clean_text('123 abc 5?67?    ')
    'abc 5?67'
    """
```

題型 3: Parallel Input

```
def match(s1: str, s2: str, s2: str) -> int:
    """Return the number of position in which s1, d2, and s3 match.

    Precondition: len(s1) == len(s2) == len(s3)

    >>> match('cat', 'bat', hat')
    2
    >>> match('cat, 'dog', 'log')
    0
    """
```

```
def stretch_string(s: str, stretch_factors: str) -> str:
    """Return a string consisting of the characters in s in the same order
    as in s, repeated the number of times indicated by the item at the
    corresponding position of stretch_factors.

    Precondition: len(s) == len(stretch_factors) and the items of
                  stretch_factors are non-negative

    >>> stretch_string('Hello', '20311')
    'HHllllo'
    >>> stretch_string('echo', '0015')
    'hooooo'
    """
```

```
def select_characters(s1: str, s2: str, selection: str) -> str:
    '''Return a new string where each character is a character from either s1
    or s2 chosen based on selection. selection is made up of 1s and 2s and
    indicates whether the character at the corresponding position of the new
    string should be from that position in s1 or that position in s2.

    Precondition: len(s1) == len(s2) == len(selection)

    >>> select_characters('coat', 'hard', '1221')
    'cart'
    >>> select_characters('pizza', 'hints', '11222')
    'pints'
    '''
```