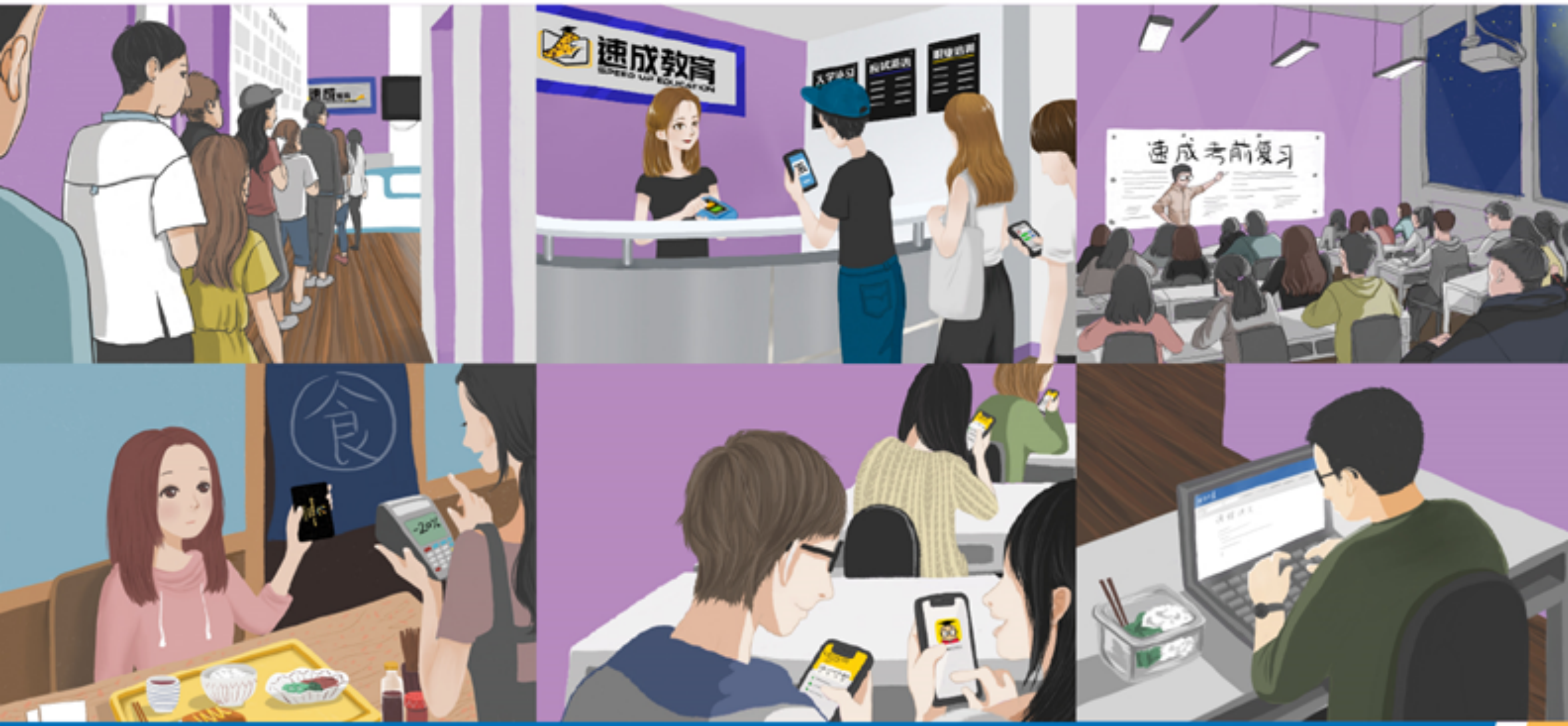
速成教育
SPEED UP EDUCATION



CSCA08

Introduction to Computer Science I

导师： **VC**

UTSC Week 02 Class | 2025/1/16

Function Design Recipe

Steps:

1. Example: If not specify, write 2 examples.
Must demonstrate different functionalities (different cases)

2. Function Header & type contract: function name and **parameter name** MUST be meaningful
Include **parameters type** and function **return type**

3. Precondition: Additional rules that inputs must should follow.
可以沒有

4. Description: Describe what the function is doing
First word is IMPORTANT, to specify the Main Purpose
`Return...` `Print...` `Modify...`
Mention every parameter name.

5. Function Body: Code ☺

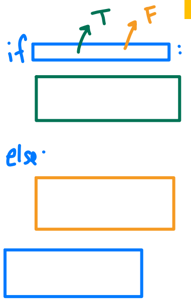
Example

docstring

```
def repeat_word(word: str, repeat_count: int) -> str:
    """Return a new string with word repeated repeat_count times.

    Precondition: repeat_count >= 0

    >>> repeat_word("Vincent ", 3)
    Vincent Vincent Vincent
    >>> repeat_word("I like programming", 0)
    ''
    """
    return word * repeat_count
```

```
def nucleobase_complement(nucleobase: str) -> str:
    """Returns the nucleobase complement of the given DNA nucleobase ('A', 'C',
    'T', 'G'). That is, 'A' and 'T' are complements of each other. 'C' and 'G'
    are complements of each other.

    Precondition:
        nucleobase: a string containing only the letters A, C, T, or G

    >>> nucleobase_complement('A')
    'T'
    >>> nucleobase_complement('G')
    'C'
    """
```

If Statement



```
num = 12
if num > 10:
    print("Large")✓
~~else:~~
    ~~print("Small")~~
print("Done")
```

Large
Done

```
num = 8
if num > 10: ✗
    ~~print("Large")~~
else: ✓
    print("Small")✓
print("Done")
```

Small
Done

```
num = 12
if num > 10:✓
    print("Large")✓

if num >= 5: ✓
    print("Medium") ✓
~~else~~:
    ~~print("Small")~~
print("Done")
```

Large
Medium
Done

else if

```
if (A):
    x
elif (B):
    y
elif (C):
    z
....

else:
    v
```

```
num = 12
if num > 10:
    print("Large")✓
~~elif num >= 5:~~
    ~~print("Medium")~~
~~else:~~
    ~~print("Small")~~
print("Done")
```

Large
Done

```
num = 8
if num > 10:✗
    print("Large")✗
elif num >=5: ✓
    print("Medium")✓
~~else~~:
    ~~print("Small")~~
print("Done")
```

Medium
Done

```
num = 12
if num >= 5:✓
    print("Medium")✓
elif num > 10✗
    ~~print("Large")~~
~~else:~~
    ~~print("Small")~~
print("Done")
```

Medium
Done

Nested If statements:

```
num = 12
if num > 10:
    if num % 2 == 0:
        print("Even Large")
    else:
        print("Odd Large")
elif nun >= 5:
    print("Medium")
else:
    print("Small")
print("Done")
```

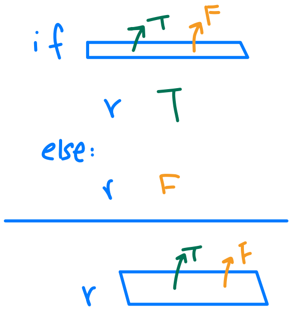
Simplify

```
if x > 1:
    if y > 2:
        if z > 3:
            print("Hello")
```

```
if x > 1 and y > 2 and z > 3:
    print("Hello")
```

```
☆ def foo(x):
    if x > 0:
        return True
    else:
        return False
```

```
def foo(x):
    return x > 0
```



Example

```
def nucleobase_complement(nucleobase: str) -> str:
    """Returns the nucleobase complement of the given DNA nucleobase ('A', 'C', 'T',
    'G'). That is, 'A' and 'T' are complements of each other. 'C' and 'G' are
    complements of each other.

    Precondition: nucleobase: a string containing only the letters A, C, T, or G

    >>> nucleobase_complement('A')
    'T'
    >>> nucleobase_complement('G')
    'C'
    """
    if nucleobase == 'A':
        complement = 'T'
    elif nucleobase == 'C':
        complement = 'G'
    elif nucleobase == 'T':
        complement = 'A'
    else:
        complement = 'C'
    return complement
```

If Statement Practices

考试原题 1:

```
def practice_time(age: int, beginner: bool) -> int:
    """Return the practice time (in minute) for a player of the given age
     who may or may not be a beginner player, according to the practice
     times in the following table:

     age of player               practice time
     -------------               -----------------------------------
     under 6 years               30 minutes
     6 to 8 years, inclusive     50 minutes
     over 8 years                75 minutes

     Add 15 minutes to the practice time for a player who is a beginner.

     Precondition: age >= 0

     >>> practice_time(8, True)
     65
     >>> practice_time(9, False)
     75
     """
     if age < 6:
        time = 30
     elif ~~6 <=~~ age <= 8:
        time = 50
     else:
        time = 75


     if beginner ~~== True~~:
        time = time + 15


     return time
```

```
if a == False:
   …

if not a:
   ...
```

```
time += 15
```

```
+=
-=
*=
/=
```

考试原题 2

```
def latte_order(shots: int, milk: str, flavour: str) -> str:
    """Return the latte order with the given espresso shots, milk, and flavour,
    formatted as follows:
    <shots> shot <flavour> latte with <milk> milk

    There should exactly one space between each word in the order, and no
    extra leading or trailing spaces.

    If milk is 'A', replace it with 'almond', and if milk is 'S', replace it
    with 'soy'.

    Precondition: milk is one of 'A', 'S', '2%', '1%'

A   >>> latte_order(2, 'A', 'vanilla')                '2 shot van-. latte with '
    '2 shot vanilla latte with almond milk'
    >>> latte_order(1, '2%', 'pumpkin spice')
    '1 shot pumpkin spice latte with 2% milk'
B   >>> latte_order(3, 'S', '')                       '3 shot _ latte with '
    '3 shot latte with soy milk'
    """
                       ''
     str(shots) + ' shot ' + flavour + ' latte with ' + …. ✗


    order = str(shots) + ' shot '

    if flavour != '':
       order += flavour + ' '

    order += 'latte with '


    if milk == 'A':
       order += 'almond'
    elif milk == 'S':
       order += 'soy'
    else:
       order += milk

    order += ' milk'

    return order
```

考试原题 3:

```
    def large_odd_num(num: int) -> bool:
        """Return True iff the given num is an odd integer greater than 100.
        Precondition: num >= 0
        >>> large_odd_num(50)
        False
        >>> large_odd_num(101)
        True
        """
        if num % 2 == 1 and num > 100:
            return True
        else:
            return False


        return num % 2 == 1 and num > 100
```

Simplifying If statements

Simplify the following functions so that there is **NO any if statements**.

考试原题 4:

```
def example1(x: int, y: int, z: str) -> bool:
    if x >= y: ✓ ✗
        if str(x) in z: ✓
            return True
        elif str(y) not in z:
            return False
        else:
            return False
    else:
        if str(x) in z: ✗
            return False
        elif str(y) not in z: ✓
            return True
        else:
            return False


def example1(x: int, y: int, z: str) -> bool:

    return   (x >= y and str(x) in z) or
             (x < y and str(x) not in z and str(y) not in z)
```

考试原题 5:

```
def example2(n: int) -> bool:
    if n % 2 == 0:
        if n % 3 == 1:
            return True
        else:
            return False
    elif n <= 4:
        if n < 0:
            return True
        else:
            return False
    else:
        if n % 3 == 1:
            return False
        else:
            return True
```

```
def example2(n: int) -> bool:
     a = n % 2 == 0 and n % 3 == 1
     b = n % 2 != 0 and n < 0
     c = n % 2 != 0 and n > 4 and n % 3 != 1

     return a or b or c
```

考试原题 6:

```
def example3(c1: int, c2: int, c3: int) -> bool:
    if c1 == c2:
        return False
    elif c1 > c2:
        if c3 <= c2:
            return False
        else:
            return True
    else:
        if c2 < c3:
            return True
        else:
            return False
```

```
def example3(c1: int, c2: int, c3: int) -> bool:
```

Art of If Statement

TOPAF 脱了裤子放屁

```
def example1():
    if x < 10:
        return True
    else:
        return False
```

```
return x < 10
```

```
def example2():
    if x < 10:
        return False
    else:
        return True
```

```
return not x < 10

return x >= 10
```

```
def example3():
    if x < 10:
        return y > 10
    else:
        return False
```

```
return x < 10 and y > 10
```

```
def example4():
    if x < 10:
        return True
    elif y > 10:
        return True
    else:
        return False
```

```
return x < 10 or y > 10
```

```
def example5():
    if x == True:
        return 0
    else:
        return
```

用数学的方式表达以下代码当 x 为多少的时候结果为 2

```
if x < 20:
    print(1)
✓elif x > 15 and x < 30:
    print(2)
else:
    print(3)
```

$[20, 30)$

$20 <= x < 30$

index

String

```
           0 1 2 3 4 5 6 7 8 9 10  ''
>>> s = "SpeedUp A08"
         -11-10         -4-3-2-1
>>> len(s)
11
>>> s[0]
'S'
>>> s[2]                              >>> s[-3]
'e'                                   'A'
>>> s[10] or s[len(s)-1] or s[-1]
'8'
 8

>>> s[11]
IndexError: index of out range
```

```
>>> "csc" + "a08"
'csca08'
>>> "csca08 " * 2
'csca08 csca08 '
>>> "apple" > "appears"
True
>>> 'cs' in 'csca08'
True
>>> "Vincent"[0] = "B"x              immutable
TypeError
>>> str(3) + "2"
'32'
```

String Slicing 永遠不會Error

s[start:end:step] 从 start 开始，往右，到 end 結束（不包括），每次增加 step.

s[start:end:-step] 从 start 开始，往左，到 end 結束（不包括），每次减少 step.

s[start:end] 默认每次+1 (step = 1)

s[:end] 默认从 0 开始

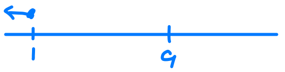
s[:] 从头到尾 same as s[0:len(s):1]

s[::-1] 从尾到头 反过来

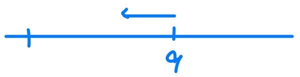
```
          0 1 2 3 4 5 6 7 8 9 10
>>> s = "abcdefghijk"
        -11-10-9-8-7-6-5-4-3-2-1

>>> s[1:9:3]  1 4 7
'beh'
>>> s[1:9:1]
'bcdefghi'
>>> s[1:9]
'bcdefghi'
>>> s[:9]
'abcdefghi'
>>> s[:]        done
'abcdefghijk'
>>> s[1:-4]
'bcdefg'
>>> s[-9:4]0
'cd'
```

```
>>> s[1:99]
'bcdefghijk'
>>> s[99:1]
''
```

```
>>> s[1:9:-1]
''
```



```
>>> s[9:1:-1]
'jihgfedc'
```



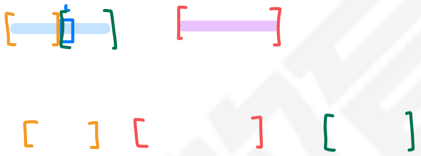
```
>>> s[::-1]
'kjihgfedcba'
```

Slicing & Cloning Practice

考试原题 7:

```
def insert(s1: str, s2: str, i: int) -> str:
    """Return a new string that is a copy of <s1> with <s2> inserted
    at index <i>.

    >>> insert("123","abc",2)
    '12abc3'
    """
    return s1[:i] + s2 + s1[i:]
```

```
>>> insert("Vincent", "csca08", 4)
'Vinccsca08ent'
 01234
```

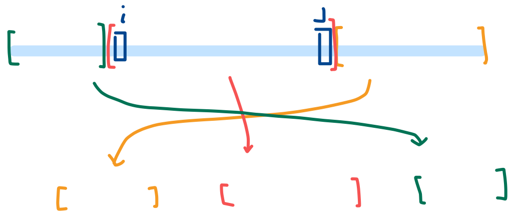


考试原题 8:

```
def cut_string(s: str, i: int) -> str:
    """Return a copy of <s> but with the character before and after
    index <i> swapped.

    >>> cut_string("ABCDEFGX1234", 7)
    '1234XABCDEFG'
    """
```

考试原题 9:

```
def triple_cut(s: str, i: int, j: int) -> str:
    """Return a copy of <s> but with characters before index <i> and swapped
    with the characters after index <j>.

    Precondition: 0 <= i < j < len(s)

    >>> triple_cut('ABCDEFGHIJKL', 2, 6)
    'HIJKLCDEFGAB'
    """
```



```
return s[j+1:] + s[i:j+1] + s[:i]
```

考试原题 10:

Consider this set of function descriptions:

```
(A) Return the character at index n of s
      Precondition: 0 <= n < len(s)

(B) Return the characters of s with two copies of the character at index n.
      Precondition: 0 <= n < len(s)

(C) Return the first n characters of s. If the length of s is less than n, return
      Precondition: n > 0

(D) Return the characters of s with the character at index n and the character at
    index len(s) – n – 1 swapped.
      Precondition: 1 <= n < len(s) // 2

(E) Return the last n characters of s
      Precondition: 0 <= n < len(s)

(F) Return the characters of s with the character at index n removed.
      Precondition: 0 <= n < len(s)

(G) Return every nth character of s, starting from the left and moving to the right.
      Precondition: 1 <= n < len(s)

(H) Return the characters of s with all occurences of str(n) removed.
```

Fill in blank with the letter from the list above the corresponds with the best description of the function.

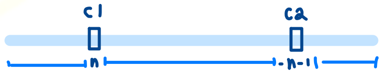
```
def func1(s: str, n: int) -> str:
    return s[::n]
```

Best description of func1:_______

```
def func2(s: str, n: int) -> str:
    value = len(s) - n
    return s[value:]
```

Best description of func2:_______

```
def func3(s: str, n: int) -> str:
    c1 = s[n]
    c2 = s[-n - 1]
    return s[:n] + c2 + s[n + 1:-n - 1] + c1 + s[-n:]
```



Best description of func3: D

```
def func4(s: str, n: int) -> str:
    return s[:n+1] + s[n:]
```

Best description of func4: B

```
def func5(s: str, n: int) -> str:
    return s[:n] + s[n+1:]
```

Best description of func5:_______

Useful String Methods

```
      0 1 2 3 4 5 6 7  9
s = " speedup csca08! "
                 11

>>> s.isalpha()         全是字母，无论大小写          'abc'.isalpha()
False
>>> s.isdigit()         全是 0 - 9                   '123'.isdigit()
False
>>> s.islower()         至少一个字母，并且全部小写     'vc'.islower()      'Vc'.islower()
~~False~~ T
>>> s.isupper()         至少一个字母，并且全部大写
~~True~~ F
>>> s.isalnum()         只有数字和字母 isalpha() + isdigit()
False


>>> s.lower()           返回一个『新』的 string，全部小写
' speedup csca08! '
>>> s.upper()           返回一个『新』的 string，全部大写
' SPEEDUP CSCA08! '

>>> s.find("csc")       返回从『左』数第一个存在的 index
9                                                        9
>>> s.find("z")         若不存在 返回-1         >>> s.find(“p”, 3, ~~6~~)
-1                                       -1
                                         7
>>> s.rfind("c")        返回从『右』数第一个存在的 index
11
                              >>> “Vincent nigga”.replace(“n”, “VC”, 2)
>>> s.replace("c", "@")       ‘ViVCceVCt nigga’
' SpeedUp @s@a08! '
>>> s.count("p")        出現次数 'Ayna'.count('a') -> 1
2

>>> s.startswith(" Spe")
~~True~~ F
>>> s.endswith("csca08! ")
True

>>> ~~'Vincent'.split('n')~~
['Vi', 'ce', 't']
               5
```

5 Week

Practice using String methods

```
def nucleobase_complement(nucleobase: str) -> str:
    """Returns the nucleobase complement of the given DNA nucleobase ('A', 'C', 'T',
    'G'). That is, 'A' and 'T' are complements of each other. 'C' and 'G' are
    complements of each other.

    Precondition:
        nucleobase: a string containing only the letters A, C, T, or G

    >>> nucleobase_complement('A')
    'T'
    >>> nucleobase_complement('G')
    'C'
    """
           0 1 2 3
    n = "ACTG"
    c = "TGAC"
    return c[n.find(nucleobase)]
                        'T'

          c[      2      ]
```

考试原题 11:

```
def mystery_function(s1: str, s2: str) -> bool:
    """Return True iff s1 starts with a number and s2 ends with a letter.
       False, otherwise.


     Precondition:
        len(s1) >= 1
        len(s2) >= 1
```

```
s = “CSCA08”

s = s.replace(“A”, “B”, 1)

print(s)


>>> CSCB08
```

```
    >>> mystery_function( “123”        ,    “abc”       )
    True
    >>> mystery_function(  “Vincent”  ,      “csca08” )
    False
    """
    return s1[0].isdigit() and s2[-1].isalpha()
```

考试原题 12:

```
def is_yes(msg: str) -> bool:
    """Return True iff msg begins with the letter y, followed by any vowel (a, e, i,
    o, u), or if msg begins with the string 'ok'.
     Your function should ignore the case of alphabetic characters.
     For example, treat 'a' and 'A' as they were the same.

     Precondition: len(msg) >= 2

     >>> is_yes("YUP!!!")
     True
     >>> is_yes("cat eyes")
     False
     >>> is_yes("OK then")
     True
     """
     msg = msg.lower()

     return (msg[0] == ‘y’ and msg[1] in ‘aeiou’) or msg.startswith(‘ok’)
                                                      msg[:2] == ‘ok’
                                                     (msg[0] == ‘o’ and msg[1] == ‘k’)
```

考试原题 13:

```
def repeat_first_letter(s: str, n: int) -> str:
    """Return a copy of s with the first character repeated n times.


    Precondition:
       n >= 0
```

```
if (A):    ✗
   return ...
elif (B):
   return ...
elif (C):
   return ...
else:
   return ...
```

```
if (A):    ✓
   return ...
if (B):
   return ...
if (C):
   return ...
return ...
```

```
if (A):
   ...
   res = ...
elif (B):
   ...
   res = ...
else:
   res = ...

return res
```

```
    >>> repeat_first_letter("abc", 2)
    'aabc'
    >>> repeat_first_letter("", 5)
    ''
    >>> repeat_first_letter("CS", 4)
    'CCCCS'
    """
    if s == ‘’:
       return ‘’
    return s[0] * n + s[1:]
```

```
if len(s) == 0:
   return s
```

考试原题 12:

```
def is_yes(msg: str) -> bool:
    """Return True iff msg begins with the letter y, followed by any vowel (a, e, i,
    o, u), or if msg begins with the string 'ok'.
     Your function should ignore the case of alphabetic characters.
     For example, treat 'a' and 'A' as they were the same.

     Precondition: len(msg) >= 2

     >>> is_yes("YUP!!!")
     True
     >>> is_yes("cat eyes")
     False
     >>> is_yes("OK then")
     True
     """
```

考试原题 13:

```
def repeat_first_letter(
    """






    >>> repeat_first_letter("abc", 2)
    'aabc'
    >>> repeat_first_letter("", 5)
    ''
    >>> repeat_first_letter("CS", 4)
    'CCCCS'
    """
```