CSC108H Winter 2024 Worksheet 13 : String Methods

On the back of this page is a help sheet for type `str` similar to what you will be given on the midterm test. Using that sheet as a reference, answer the following questions.

1. Consider this code

   ```
   wish = 'Happy Birthday'
   ```

   Assuming the code above has been executed, circle the expression(s) that produce `'happy birthday'`.

   (a) `wish[0].lower() + wish[6].lower()`
   (b) `wish.swapcase()`
   (c) `wish[0].lower() + wish[1:6] + wish[6].lower() + wish[7:]`
   (d) `wish.lower()`

2. Consider this code

   ```
   robot = 'R2D2'
   ```

   Assuming the code above has been executed, circle the expression(s) that produce `True`.

   (a) `robot.isupper()`
   (b) `robot.isalpha()`
   (c) `robot.isalnum()`
   (d) `robot.isdigit()`

3. Consider this code

   ```
   lyrics = '''O Canada!
   Our home and native land!
   True patriot love in all of us command.'''
   ```

   Circle the expression that produces the index of the second exclamation mark.

   (a) `lyrics.find('!')`
   (b) `lyrics.find('!').find('!')`
   (c) `lyrics.find('!', lyrics.find('!'))`
   (d) `lyrics.find('!', lyrics.find('!') + 1)`

CSC108H Winter 2024 Worksheet 13 : String Methods

Short Python help descriptions:

```
str:
  x in s --> bool
    Produce True if and only if string x is in string s.
  str(x: object) -> str
    Convert an object into its string representation, if possible.
  S.count(sub: str[, start: int[, end: int]]) -> int
    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are interpreted
    as in slice notation.
  S.find(sub: str[, i: int]) -> int
    Return the lowest index in S (starting at S[i], if i is given) where the
    string sub is found or -1 if sub does not occur in S.
  S.index(sub: str) -> int
    Like find but raises an exception if sub does not occur in S.
  S.isalnum() -> bool
    Return True if and only if all characters in S are alphanumeric
    and there is at least one character in S.
  S.isalpha() -> bool
    Return True if and only if all characters in S are alphabetic
    and there is at least one character in S.
  S.isdigit() -> bool
    Return True if and only if all characters in S are digits
    and there is at least one character in S.
  S.islower() -> bool
    Return True if and only if all cased characters in S are lowercase
    and there is at least one cased character in S.
  S.isupper() -> bool
    Return True if and only if all cased characters in S are uppercase
    and there is at least one cased character in S.
  S.lower() -> str
    Return a copy of the string S converted to lowercase.
  S.lstrip([chars: str]) -> str
    Return a copy of the string S with leading whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  S.replace(old: str, new: str) -> str
    Return a copy of string S with all occurrences of the string old replaced
    with the string new.
  S.rstrip([chars: str]) -> str
    Return a copy of the string S with trailing whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  S.split([sep: str]) -> list of str
    Return a list of the words in S, using string sep as the separator and
    any whitespace string if sep is not specified.
  S.strip([chars: str]) -> str
    Return a copy of S with leading and trailing whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  S.swapcase() -> str
    Return a copy of S with uppercase characters converted to lowercase
    and vice versa.
  S.upper() -> str
    Return a copy of the string S converted to uppercase.
```

CSC108H Winter 2024 Worksheet 14 : For Loops Over Strings

For each function, add at least one example to the docstring and complete the function body.

1. `def count_uppercase(s: str) -> int:`
   `"""Return the number of uppercase letters in s.`

   ```
   >>> count_uppercase('abc')
       0
   >>> count_uppercase('ABc')
       2
   ```

   `"""`

   ```
   uppercase_count = 0
   for char in s:
       if char.isupper():
           uppercase_count += 1
   return uppercase_count
   ```

2. `def all_fluffy(s: str) -> bool:`
   `"""Return True if and only if every character in s is fluffy. Fluffy`
   `characters are those that appear in the word 'fluffy'.`

   ```
   >>> all_fluffy('fluffy')
       True
   >>> all_fluffy('nice')
       False
   ```

   `"""`

   ```
   for char in s:
       if char not in 'fluffy':
           return False
   return True
   ```

3. `def add_underscores(s: str) -> str:`
   `"""Return a string that contains the characters from s with an underscore added after`
   `every character except the last.`

   ```
   >>> add_underscores('hello')
       'h_e_l_l_o'
   >>> add_underscores('abc')
       'a_b_c'
   ```

   `"""`

   ```
   result = ''
   for char in s[:-1]:
       result += char + '_'
   result += s[-1]
   return result
   ```

CSC108H Winter 2024 Worksheet 15 : While Loops

1. In the boxes below, fill in the missing code that will make the function definition match its description.

```
def every_nth_character(s: str, n: int) -> str:
    """Return a string that contains every nth character from s, starting at index 0.

    Precondition: n > 0

    >>> every_nth_character('Computer Science', 3)
    'CpeSee'
    """

    result = ''
    i = 0  # The index of the next character to examine.

    while [ i < len(s)                ]:

        result = result + s[i]

        i = [ i + n                   ]

    return result
```

2. In the boxes below, fill in the missing code that will make the function definition match its description.

```
def find_letter_n_times(s: str, letter: str, n: int) -> str:
    """Return the smallest substring of s starting from index 0 that contains
    n occurrences of letter.

    Precondition: letter occurs at least n times in s

    >>> find_letter_n_times('Computer Science', 'e', 2)
    'Computer Scie'
    """

    i = 0  # The index of the next character to examine.
    count = 0  # The number of occurrences of letter in s[:i].

    while [ i < len(s) and count < n  ]:
        if [ s[i] == letter           ]:
            count = count + 1
        i = i + 1

    return [ s[:i]                    ]
```

CSC108H Winter 2024 Worksheet 15 : While Loops

3. In math, the Collatz conjecture states that starting from any positive integer, you will eventually reach the number 1 by repeatedly applying the following two rules:

   - if the number is even, divide it by 2 to get the next number in the sequence
   - if the number is odd, multiply by 3 and add 1 to get the next number in the sequence

   Repeatedly applying the rules generates a sequence of numbers. The Collatz step count is the number of applications of the rules required before the sequence reaches 1. For example, there are 8 Collatz steps in the Collatz sequence:
   `n = 6 -> n = 3 -> n = 10 -> n = 5 -> n = 16 -> n = 8 -> n = 4 -> n = 2 -> n = 1`

   Complete this function to count the Collatz steps for a particular number n.

```
def count_collatz_steps(n: int) -> int:
    """Return the number of steps it takes to reach 1 by applying the two rules
    of the Collatz conjecture beginning from the positive integer n.

    Precondition: n >= 1

    >>> count_collatz_steps(6)
    8
    """
     steps = 0
     while n != 1:
         if n % 2 == 0:
             n = n / 2
         else:
             n = n * 3 + 1
         steps += 1
     return steps
```

4. The function below has an incomplete header and docstring. Based on the code in the function body, fill in the missing parts: the Header (including the Type Contract), Description, and Examples.

```
def first_digit(s: str) -> int:
    """
    Finds the first digit in the string and returns the index of the first digit.

    Precondition: s must contain at least one digit

    >>> first_digit(a1)
    1

    >>> first_digit(nice 1)
    5


    """

    i = 0

    while i < len(s) and s[i] not in '0123456789':
        i = i + 1
    return i
```