Do not turn this page until you have received the signal to start.

(Please fill out the identification section above, write your name on the back of the test, and read the instructions below.)

Good Luck!

This midterm is double-sided, and consists of 6 questions and a list of function/method descriptions. When you receive the signal to start, please make sure that your copy is complete.

- Comments are not required except where indicated, although they may help us mark your answers.
- No error checking is required: assume all user input and all argument values are valid.
- If you use any space for rough work, indicate clearly what you want marked.
- Do not remove any pages from the exam booklet.
- You may use a pencil; however, work written in pencil will not be considered for remarking.

TOTAL: _____/23
Question 1.  [6 marks]

Beside each code fragment in the table below, write what is printed when the code fragment is executed. If the code would cause an error, write ERROR and give a brief explanation.

<table>
<thead>
<tr>
<th>Code</th>
<th>Output or Cause of Error</th>
</tr>
</thead>
</table>
| `s = 'two'
v = int(s)
print(v)` | ERROR: Cannot convert 'two' to int |
| `x = '1' + 2
print(x)` | ERROR: Cannot concatenate string and integer |
| `list1 = [1, 2, 3, 4]
element = list1[1:] [1]
print(element)` | 3 |
| `L = ['a', 'b']
L = L.append('c')
print(L)` | ['a', 'b', 'c'] |
| `for value in range(7, 2, -2):
    print(value)` | ERROR: Range step cannot be negative |
| `result = 'banana'.find('an', 2)
print(result)` | 3 |
**Question 2.**  [2 marks]

Complete the docstring examples with arguments that will cause the function calls to return the values shown.

```python
def midterm_function(s1: str, s2: str) -> bool:
    """
    Precondition: len(s1) >= 1 and len(s2) >= 1
    >>> midterm_function( , )
    True
    >>> midterm_function( , )
    False
    """

    return s1[0].isdigit() and s2[-1].isalpha()
```

**Question 3.**  [3 marks]

Step 1 of the Function Design Recipe (docstring examples) has been completed for the function `repeat_first_letter`. Complete steps 2 and 3 of the Function Design Recipe: Fill in the function header (including the type contract) and write a good description.

Do not write the function body. Do not include preconditions.

```python
def repeat_first_letter
    """

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

    # DO NOT WRITE THE BODY OF THIS FUNCTION
```
Question 4. [4 marks]

Complete the following function according to its docstring.

```python
def practice_time(age: int, beginner: bool) -> int:
    """Return the practice time (in minutes) 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 for a beginner player
    ----------------------|-----------------------------------------
    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 not a beginner.

    Precondition: age >= 0

    >>> practice_time(8, True)
    50
    >>> practice_time(9, True)
    75
    >>> practice_time(7, False)
    65
    """
```

from web solution
Question 5.  [5 marks]

Complete the following function according to its docstring.

```python
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
    """
```

```python
>>> has_letter_cases('abcDEF')
True
>>> has_letter_cases('abc123')
False
>>> has_letter_cases('ABCXYZ')
False
```
Question 6. [3 marks]

Fill in the box with the while loop condition required for the function to work as described in its docstring.

def find_first_fluffy(msg: str) -> int:
    """Return the index of the first fluffy character (f, l, u, y) in msg, or the length of msg if it does not contain any fluffy characters. Fluffy characters are those that appear in the word 'fluffy'.\n    \n    >>> find_first_fluffy('firefly')
    0
    >>> find_first_fluffy('cats')
    4
    >>> find_first_fluffy('Fully')
    1
    """

i = 0
while
    msg[i] not in 'fluffy' and i < len(msg)
    i = i + 1
return i

stop

looking

swap

order

this order will cause error for 'cats'

stop when i == len(msg)

↓
i < len(msg)

↓
i < (len(msg) and msg[i] not in 'fluffy')
[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]
Short Python function/method descriptions:

__builtins__:

`int(x: object) -> int`
- Convert x to an integer, if possible. A floating point argument will be truncated towards zero.

`len(x: object) -> int`
- Return the length of list, tuple, or string x.

`print(values: object) -> None`
- Prints the values.

`range([start: int], stop: int, [step: int]) -> list-like-object of int`
- Return the integers starting with start and ending with stop - 1 with step specifying the amount to increment (or decrement). If start is not specified, the sequence starts at 0. If step is not specified, the values are incremented by 1.

`str(x: object) -> str`
- Return an object converted to its string representation, if possible.

str:

`x in s -> bool`
- Produce True if and only if string x is in string s.

`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.isalpha() -> bool`
- Return True if and only if all characters in S are alphabetic and there is at least one character 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.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.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.upper() -> str`
- Return a copy of the string S converted to uppercase.

list:

`x in L -> bool`
- Produce True if and only if object x is in list L.

`L.append(item: object) -> None`
- Append item to end of list L.

`L.extend(items: iterable) -> None`
- Extend list L by appending elements from items. Strings and lists are iterables whose elements are characters and list items respectively.