CSC108H Winter 2024 Worksheet 09 : if statements

Complete the following functions.

1.
```
def earlier_name(name1: str, name2: str) -> str:
    """Return the name, name1 or name2, that comes first alphabetically.

    >>> earlier_name('Jen', 'Paul')
    'Jen'
    >>> earlier_name('Colin', 'Colin')
    'Colin'
    """
    if name1 < name2:
        return name1
    else:
        return name2
```

2.
```
def ticket_price(age: int) -> float:
    """Return the ticket price for a person who is age years old.
    Seniors 65 and over pay 4.75, kids 12 and under pay 4.25 and
    everyone else pays 7.50.

    Precondition: age > 0

    >>> ticket_price(7)
    4.25
    >>> ticket_price(21)
    7.5
    >>> ticket_price(101)
    4.75
    """
    if age <= 12
        return 4.25
    elif age >= 65:
        return 4.75
    else:
        return 7.5
```

3.
```
def format_name(first: str, last: str) -> str:
    """Return the first and last names as a single string, in the form:
    last, first
    Mononymous persons (those with no last name) should have their name
    returned without a comma.

    >>> format_name('Cherilyn', 'Sarkisian')
    'Sarkisian, Cherilyn'
    >>> format_name('Cher', '')
    'Cher'
    """
    if not last:
        return first
    else:
        return last + ', ' + first
```

CSC108H Winter 2024 Worksheet 10 : Functions and Booleans

Complete the following functions.

1.
```
def same_abs(num1: float, num2: float) -> bool:
    """Return True if and only if num1 and num2 have the same absolute value.

    >>> same_abs(-3.2, 3.2)
    True
    >>> same_abs(3.0, 3.5)
    False
    """
    return abs(num1) == abs(num2)
```

2.
```
def different_types(obj1: object, obj2: object) -> bool:
    """Return True if and only if obj1 and obj2 are of different types.

    >>> different_types(3, '3')
    True
    >>> different_types(108.0, 3.14)
    False
    """
    return type(obj1) != type(obj2)
```

3. *An extra exercise to try at home.*

```
def is_right_triangle(side1: int, side2: int, hypotenuse: int) -> bool:
    """Return whether a triangle with sides of length side1, side2 and
    hypotenuse is a right triangle.

    >>> is_right_triangle(3, 4, 5)
    True
    >>> is_right_triangle(2, 2, 4)
    False
    """
    return side1 ** 2 + side2 ** 2 == hypotenuse ** 2
```

CSC108H Winter 2024 Worksheet 11 : No if required

Each of the following functions is correctly implemented, but is more complex than it needs to be. Each function body can be replaced with a single return statement. You can use comparison operators `<`, `>`, `<=`, and so on, as well as boolean operators `and`, `or`, and `not`.

1.
```
def can_vote(age: int) -> bool:
    """Return True if and only if age is legal voting age of at least 18 years.

    >>> can_vote(16)
    False
    >>> can_vote(21)
    True
    """

    if age < 18:
        return False
    else:
        return True
```

Complete the new single-line function body in the box below:

```
return  age >= 18
```

2.
```
def is_teenager(age: int) -> bool:
    """Return True if and only if age is a teenager between 13 and 18 inclusive.

    >>> is_teenager(4)
    False
    >>> is_teenager(16)
    True
    >>> is_teenager(19)
    False
    """

    if age < 13:
        return False
    else:
        if age > 18:
            return False
        else:
            return True
```

For which age range will this function return `True`**?** `age in [13, 18]`

Complete the new single-line function body in the box below:

```
return  13 <= age <= 18
```

CSC108H Winter 2024 Worksheet 12 : String Operations

1. Consider this code:

   ```
   phrase = 'Laughing Out Loud'
   ```

   Assuming the code above has been executed, complete the indices in the expression below that will produce the string `'LOL'`. Use at least one negative index in your answer.

   ```
   phrase[ :1 ] + phrase[9:10] + phrase[-4:-3]
   ```

2. Consider this code:

   ```
   phrase = 'big orange cat'
   slice1 = phrase[:3]
   slice2 = phrase[-4:]
   slice3 = phrase[3:8]
   ```

   Assuming the code above has been executed, complete the table with the values that each variable refers to.

Variable	Value
`phrase`	`‘big orange cat’`
`slice1`	`‘big’`
`slice2`	`‘ cat’`
`slice3`	`‘ oran’`

3. Consider this code:

   ```
   lyrics = 'abc easy as 123'
   ```

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

   (a) `'easy' in lyrics` (b) `str(len('mj')) in lyrics`

   [✓ (c) `'cab' in lyrics`] (d) `'' in lyrics`

4. Consider this code:

   ```
   s = 'Jacqueline'
   ```

   You know that the slicing operation `s[1:4]` will produce the string `'acq'`. The slicing operation has an optional third parameter that determines the *stride* (or distance between characters) in the slice. For example, the slicing operation `s[::2]` will produce the string `'Jculn'`, which has every other character in `'Jacqueline'`, starting from the first character in the string, and up to the end of the string. Use a negative stride to work backwards through a string.

   (a) Write an expression that uses slicing on `s` to produce the string `'aqeie'`.

   `s[1::2]`

   (b) Write an expression that uses slicing on `s` to produce the string `'enileuqcaJ'`.

   `s[::-1]`

   (c) Write an expression that uses slicing on `s` to produce the string `'eieqa'`.

   `s[::-2]`