

PYTHON FUNCTIONS, EXPRESSIONS & CONTROL

COMPUTER SCIENCE MENTORS 61A

January 27 – January 31, 2024

1 Intro to Python

1. What Would Python Display?

```
>>> 3
```

```
>>> "cs61a"
```

```
>>> x = 3
>>> x
```

```
>>> x = print("cs61a")
cs61a
>>> x
```

```
>>> print(print(print("cs61a")))
```

```
>>> def f1(x):
...     return x + 1
>>> f1(3)
```

```
>>> f1(2) + f1(2 + 3)
```

```
>>> def f2(y):
...     return y / 0
>>> f2(4)
```

```

>>> def f3(x, y):
...     if x > y:
...         return x
...     elif x == y:
...         return x + y
...     else:
...         return y
>>> f3(1, 2)

>>> f3(5, 5)

>>> 1 or 2 or 3

>>> 1 or 0 or 3

>>> 4 and (2 or 1/0)

>>> 0 or (not 1 and 3)

>>> (2 or 1/0) and (False or (True and (0 or 1)))

```

2. For the following expressions, simplify the operands in the order of evaluation of the entire expression

Example: `add(3, mul(4, 5))`

Order of Evaluation: `add(3, mul(4, 5))` \rightarrow `add(3, 20)` \rightarrow 23

(a) `add(1, mul(2, 3))`

(b) `add(mul(2, 3), add(1, 4))`

(c) `max(mul(1, 2), add(5, 6), 3, mul(mul(3, 4), 1), 7)`

1. Write a function that returns `True` if a number is divisible by 4, 1 if a number is divisible by 7 and is not already divisible by 4, and returns `False` if neither condition is fulfilled.

```
def divisibility_check(num):
```

2. Implement `pow_of_two`, which prints all the positive integer powers of two less than or equal to `n` in ascending order. This function should return `None`.

*Follow up question: What would you change about your solution if the question asked to print all the powers of two **strictly less than** `n`?*

```
def pow_of_two(n):  
    """  
    >>> pow_of_two(6)  
    1  
    2  
    4  
    >>> result = pow_of_two(16)  
    1  
    2  
    4  
    8  
    16  
    >>> result is None  
    True  
    """
```

3. Write a function, `is_leap_year`, that returns `true` if a number is a leap year and `false` otherwise. A *leap year* is a year that is divisible by 4 but not by 100, except for years divisible by 400, which are leap years.

```
def is_leap_year(year):  
    """  
    Returns whether ``year`` is a leap year.  
    >>> is_leap_year(2002)  
    False  
    >>> is_leap_year(2004)  
    True  
    >>> is_leap_year(2000)  
    True  
    >>> is_leap_year(1900)  
    False  
    >>> is_leap_year(2100)  
    False  
    """  
    return _____
```

4. Complete the function `fact_limit`, which calculates factorials up to a specified limit. Specifically, `fact_limit` takes in two positive integers, `n` and `limit`, and calculates the product of `n`, `n-1`, `n-2`, etc., working downward until it attains the greatest product that doesn't exceed `limit`. If there is no product less than or equal to `limit`, `fact_limit` should return 1.

Hint: The output of `fact_limit` is always less than or equal to `limit`.

```
def fact_limit(n, limit):
    """
    >>> fact_limit(5, 20)
    20 # 5 * 4 = 20, but 5 * 4 * 3 = 60 > 20
    >>> fact_limit(5, 200)
    120 # 5 * 4 * 3 * 2 * 1 = 120 < 200
    >>> fact_limit(5, 3)
    1 # no partial product is less than 3
    """
    if _____:
        _____

    product = _____

    _____ = n - 1

    while _____:
        _____ = _____
        _____ = _____

    return _____
```