PYTHON AND CONTROL Solutions

COMPUTER SCIENCE MENTORS

January 24 - January 30, 2021

1 Intro to Python

1. What Would Python Display?

```
>>> 3
```

3

```
>>> "csm"
```

'csm'

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

3

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

None

```
>>> print(print(print("csm")))
```

csm
None
None

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

4

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

9

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

ZeroDivisionError: division by zero

```
>>> def f3(x, y):
...     if x > y:
...             return x
...     elif x == y:
```

```
...             return x + y
...     else:
...             return y
>>> f3(1, 2)
```

2

```
>>> f3(5, 5)
```

10

```
>>> 1 or 2 or 3
```

1

```
>>> 1 or 0 or 3
```

1

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

2

```
>>> 0 or (not 1 and 3)
```

False

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

1

2. For the following expressions, list the order of evaluation of the operators and operands of the expression.

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

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

   add, 1, mul, 2, 3

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

   add, mul, 2, 3, add, 1, 4

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

   max, mul, 1, 2, add, 5, 6, 3, mul, mul, 3, 4, 1, 7

CSM 61A Spring 2021: Ivan Penev and Joshua Baum, with
Jamie Ip, Kenneth Chi, Kevin Moy, Mathilde McKeever, Cyrus Bugwadia, and
Kunal Agarwal, Luke Liu, John So, Jemmy Zhou, Ruslana Yurtyn, Vivian Lu, Matt Au, Lily Yang, Mihira Patel, Jessica Yu, Aidan Tong, Matthew Guo, Nikhita Anumukonda, Laryn Qi, Grace Yi, Jennifer Huang, Cindy Lin, James Fletcher, Sean Sananikone, Abinaya Srikant, Uma Unni

2 Control

1. Write a function that returns true if a number is divisible by 4 and false otherwise.

```
def is_divisible_by_4(num):
    return num % 4 == 0
```

2. 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 divisible by 400.

```
def is_leap_year(year):
    return year % 4 == 0 and year % 400 != 0
```

3. Write a function `find_max` that will take in 3 numbers, `x`, `y`, `z`, and return the max value. Assume that `x`, `y`, and `z` are unique. Do not use Python's built-in `max` function.

```
def find_max(x, y, z):
```

```
def find_max(x, y, z):
    if x > y and x > z:
        return x
    elif y > x and y > z:
        return y
    else:
        return z
```

CSM 61A Spring 2021: Ivan Penev and Joshua Baum, with
Jamie Ip, Kenneth Chi, Kevin Moy, Mathilde McKeever, Cyrus Bugwadia, and
Kunal Agarwal, Luke Liu, John So, Jemmy Zhou, Ruslana Yurtyn, Vivian Lu, Matt Au, Lily Yang, Mihira Patel, Jessica Yu, Aidan Tong, Matthew Guo, Nikhita Anumukonda, Laryn Qi, Grace Yi, Jennifer Huang, Cindy Lin, James Fletcher, Sean Sananikone, Abinaya Srikant, Uma Unni

4. Implement `pow_of_two`, which takes in an integer `n` and prints all the positive, integer powers of two less than or equal to n. 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
    """

    curr = 1
    while curr <= n:
        print(curr)
        curr *= 2 # equivalent to curr = curr * 2
```

Since we are multiplying curr by 2 on each iteration of the while loop, curr holds values that are powers of 2. Notice that since there is no return statement in this function, when Python reaches the end of the function, it automatically returns None.

The answer to the follow up question is that the condition of our while loop would change to `curr < n`. Walk through the code for `pow_of_two(16)` with both of the conditions to see why they produce different outputs!

Another way you could have written this function is by using pow or the ** operator. That solution would look something like this where you would keep track of the exponent itself:

```
exponent = 0
while (2 ** exponent) <= n:
    print(2 ** exponent)
    exponent += 1
```