

CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2018

Swarthmore College

Informal quiz: discuss with a new partner

Code draft from Student X: `mystery_error.py`

```
def mystery(lst):
    s = 0
    for i in range(10):
        s = s + lst(i)
    print(s)

def main():
    my_lst = [8, 3, 7, 2, 4, 9, 1, 19, 2, 17]

    mystery(my_lst)
    print("result1 is:", s)
    print("result2 is:", s/len(my_lst))

main()
```

- 1) What is Student X trying to do?
- 2) What errors do you notice in this program?
- 3) What style modifications would you make?

Outline Sept 26:

- Hand back Quiz 1
- Continue Functions (go over **factorial.py**)
- Scope and program execution
- Stack diagrams
- Multi-function example: **first_last.py**

Notes

- **Lab 3** due **Saturday** night
- Ninja session tonight! 7-10pm
- **Office hours Friday 3-5pm**

Continue Functions

factorial.py example solution

```
"""
Practice with functions. Write a function that takes one argument, a
non-negative integer n, and returns  $n! = n*(n-1)*(n-2)...3*2*1$ . Note:  $0! = 1$ 

Author: Sara Mathieson
Date: 9/24/18
"""

def factorial(n):
    """
    Given a non-negative integer n, return  $n! = n*(n-1)*(n-2)...3*2*1$ .
    """
    fact = 1 # set up an accumulator variable
    for i in range(n):
        fact = fact * (i+1) # accumulator pattern
    return fact

def main():
    """
    In the main function, test the factorial function.
    """
    # test on 100 different numbers
    for n_test in range(100):

        # call/invoke the factorial function
        result = factorial(n_test)

        # print the result
        print("%i! = %i" % (n_test, result))

main()
```

Scope and program execution

[What are shared sessions?](#)

Python 3.6

```
1 def factorial(n):
2     fac = 1 # set up an accumulator variable
3     for i in range(n):
4         fac = fac * (i+1) # accumulator pattern
5     return fac
6
7
8 def main():
9
10    for n_test in range(10):
11        # call the factorial function
12        result = factorial(n_test)
13
14        # print the result
15        print("%d! = %d" % (n_test, result))
16
17    main()
```

[Edit code](#) | [Live programming](#)

→ line that has just executed

→ next line to execute

Print output (drag lower right corner to resize)

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
```

Frames

Objects

Global frame

factorial

main

function

factorial(n)

function

main()

main

n_test 5

result 120

pythontutor.com, stack diagrams

function stack

heap (values)

```
1 def f(x,y):
```

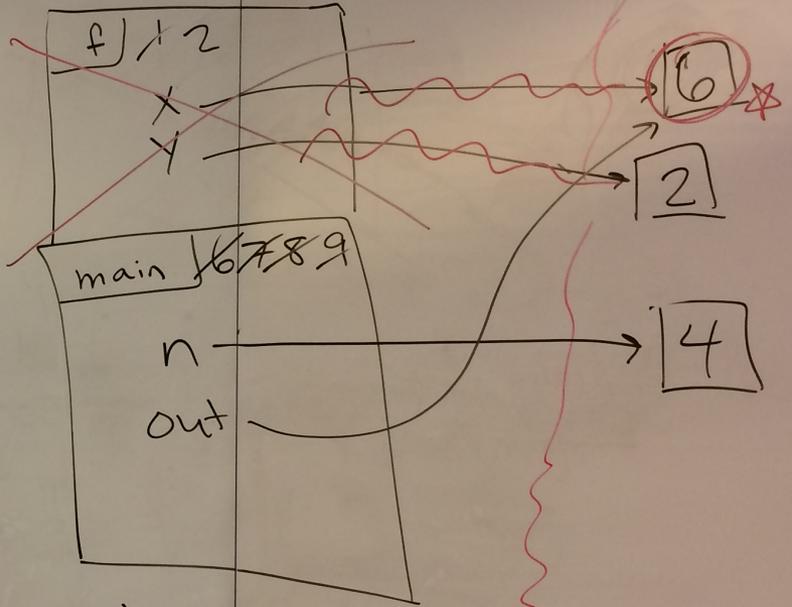
```
2     x = x + y
3     # draw stack/heap here
4     print(x)
```

```
5     return x
```

```
6 def main():
```

```
7     n = 4
8     out = f(n, 2)
9     print(out)
```

```
11 main()
```



output

6
6

first
symbols

! * + = . >

second
uppercase

A B : Z

third
lowercase

a b : z



Program for today

- [cs21/inclass/week04/first_last.py](#)
- Work with a partner
- Write the functions in order, testing each one
- No need to change main! Only to uncomment test cases