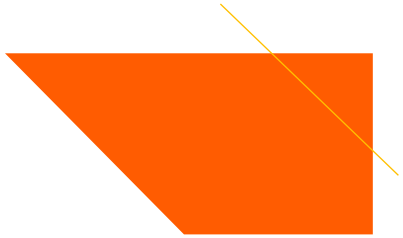
Introduction to Problem Solving in Python

COSI 10A



Class objectives

- Cumulative Algorithms (Section 4.2)
- Indefinite Loops (`While`) (Section 5.1)

String question

- Write a function called `word_count` that accepts a string as a parameter and returns the number of words in the string. For example `word_count("hello")` should return 1, `word_count("how are you?")` should return 3

String question

- Write a function called `word_count` that accepts a string as a parameter and returns the number of words in the string. For example `word_count("hello")` should return 1, `word_count("how are you?")` should return 3

```
def word_count(s):
    count = 0
    if s[0] != ' ':
        count += 1
    for i in range(len(s) - 1):
        if s[i] == ' ' and s[i + 1] != ' ':
            count += 1
    return count
```

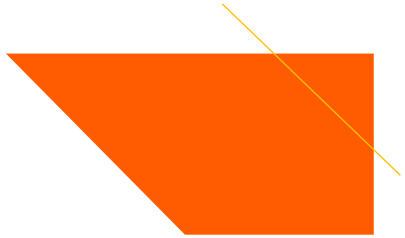
`if/else return` question

- Write a function `count_factors` that returns the number of factors of an integer
- `count_factors(24)` returns `8` (the factors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24)

if/else return question

- Write a function `count_factors` that returns the number of factors of an integer
- `count_factors(24)` returns `8` (the factors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24)

```
def count_factors(number):
    count = 0
    for i in range(1, number + 1):
        if (number % i == 0):
            count += 1       # i is a factor of number
    return count
```



Cumulative Algorithms

Modify and assign operators

- Shortcuts to modify a variable's value

Shorthand	Equivalent longer version
variable += value	**variable = variable + value**
variable -= value	**variable = variable - value**
variable *= value	**variable = variable * value**
variable /= value	**variable = variable / value**
variable //= value	**variable = variable // value**
variable %= value	**variable = variable % value**

```
x += 3                  # x = x + 3
gpa -= 0.5              # gpa = gpa - 0.5
number *= 2             # number = number * 2
```

Adding many numbers

- How would you find the sum of all integers from 1-1000?

```
# This may require a lot of typing
sum = 1 + 2 + 3 + 4 + ...
print("The sum is", sum)
```

- What if we want the sum from 1 - 1,000,000? Or the sum up to any maximum? How can we generalize the above code?

Cumulative sum loop

```
sum = 0
for i in range(1, 1001):
    sum = sum + i

print("The sum is", sum)
```

- **Cumulative sum** refers to a variable that keeps a sum in progress and is updated repeatedly until summing is finished.
  - The `sum` in the above code is an attempt at a cumulative sum
  - **Cumulative sum variables must be declared *outside* the loops that update them**

Cumulative product

- This cumulative idea can be used with other operators:

```
product = 1
for i in range(1, 21):
    product = product * 2

print("2 ^ 20 =", product)
```

`input` and cumulative sum

- We can do a cumulative sum of user input:

```
sum = 0
for i in range(1, 101):
    next = int(input("Type a number: "))
    sum = sum + next

print("The sum is", sum)
```

Cumulative sum question

- Write a program that prompts the user for how many people ate, each person's dinner cost, and calculates the total cost

- Example log of execution:

```
How many people ate? 4
Person #1: How much did your dinner cost? 20.00
Person #2: How much did your dinner cost? 15
Person #3: How much did your dinner cost? 30.0
Person #4: How much did your dinner cost? 10.00

Subtotal: $75.0
Tax: $6.0
Tip: $13.5
Total: $94.5
```

Cumulative sum answer

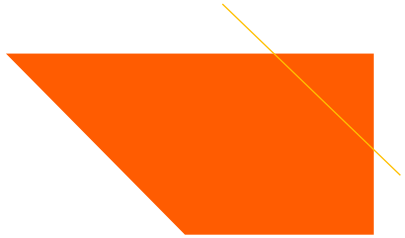
```
def main():
    subtotal = meals()
    results(subtotal)

# Prompts for number of people and returns total meal subtotal.
def meals():
    people = int(input("How many people ate? "))
    subtotal = 0.0;             # cumulative sum
    for i in range(1, people + 1):
        person_cost = float(input("Person #" + str(i) + ": How much did your dinner cost? "))
        subtotal = subtotal + person_cost  # add to sum
    return subtotal
...
```

Cumulative sum answer, cont.

```
# Calculates total owed, assuming 8% tax and 15% tip
 def results(subtotal):
     tax = subtotal * .08
     tip = subtotal * .18
     total = subtotal + tax + tip

     print("Subtotal: $" + str(subtotal))
     print("Tax: $" + str(tax))
     print("Tip: $" + str(tip))
     print("Total: $" + str(total))
```



`while` loops

Categories of loops

- **Definite loop**: Executes a known number of times. The for loops we have seen are definite loops
  - Print "hello" 10 times.
  - Find all the prime numbers up to an integer n.
  - Print each odd number between 5 and 127.

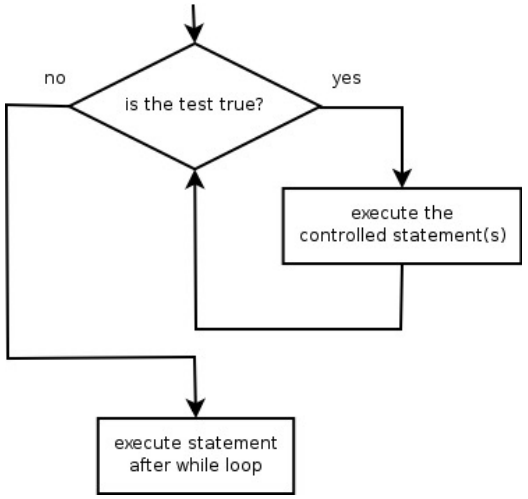
- **Indefinite loop**: One where the number of times its body repeats is not known in advance
  - Prompt the user until they type a non-negative number.
  - Print random numbers until a prime number is printed.
  - Repeat until the user has typed "q" to quit.

The `while` loop

- **`while` loop**: Repeatedly executes its body as long as a logical test is true.

Syntax:

```
while test:
    statement(s)
```



Example:

```
num = 1                                  # initialization
while num <= 200:                        # test
    print(str(num) + " ", end='')
    num = num * 2                        # update
```

Example `while` loop

- Find the first factor of 91, other than 1

Example `while` loop

- Find the first factor of 91, other than 1

```
n = 91
factor = 2
while n % factor != 0:
    factor += 1
print("First factor is", factor)
```

Trace `while` loop

- Trace code with x = 1, x = 6, x = 19, x = 39

```
def mystery(x):
    y = 1
    z = 0
    while 2 * y <= x:
        y = y * 2
        z += 1
    print(y, z)
```