Modular Programming

Python Bootcamp, Day 2

Anna Rosen

Overview of Day 1

- Yesterday covered program structure, programming logic, and pseudocode
    - The structure of a program contains: variables, assignment, input/output, selection, and repetition
    - Programming logic is used to allow the computer to make decisions on what code to execute.
    - This is done with selection, repetition, and conditional statements.
    - Pseudocode is a simple way to structure a program without language specific jargon and also help you plan out your program.

Overview of Day 2 (Morning)

- This morning covered:
  - Defining functions
  - Importing modules
  - Control flow (if statement and comparison operators, iteration, exception handling)
  - Advanced data types (lists, tuples, dictionaries)

Today we'll cover modular programming.

What is modular programming?

Modular programming is a software design technique which emphasizes dividing up the functionality of a program into independent, interchangeable modules.

Each module contains everything necessary to execute only one aspect of the desired functionality.

Usefulness of modular programming

Separating your program into modules (aka functions, routines) allows the user to call a module many times without having to rewrite the functionality every time.

Pseudocode example 1

```
Program GetTotal:
    get itemPrice
    salesTax = call GetSalesTax(itemPrice)
    print itemPrice + salesTax
END

Function GetSalesTax(itemPrice):
    caSalesTax = 0.0875
    salesTax = itemPrice * caSalesTax
    return salesTax
END
```

Functions - Practice 1

1. Write a function in python named ‘sum’ which sums two integers and returns the sum
2. Generalize your function named ‘sum’ to calculate the sum of an array of numbers

```
def sum(num1, num2):
    """
    Takes in two numbers and returns the sum
    """
    sum = num1 + num2
    return(sum)

def sum(num_arr):
    """
    Takes in an array of numbers and returns the sum
    """
    sum = 0
    for num in num_arr:
        sum = sum + num
    return(sum)
```

Writing Modular Code in Python

```
import module1 as md1
import module1 as md1

#Define functions outside of main function

def function1(arg1, arg2):
    """
    This function does x and y
    """
    command1
    x = command2
    #return your value
    return(x)

def function2(arg1, arg2, arg3):
    """
    This function does w and z
    """
    x =command1
    y = command2
    if (y > 5):
        #return your value
        return(y)
    else:
        return(x)
#this is your main program which calls your functions
if __name__=="__main__":
    a1 = 5
    a2 = 4
    a3 = 3
    z = function1(a1, a2)
    x = function1(a2, a1)
    y = function2(a1, a2, a3)

#END
```

Two ways to include modules in your scripts:

1. Write functions in different script (i.e., mydefs.py) and load into main program script with
   import mydefs
   import mydefs as md

   Access functions by
   mydefs.myfunc(args)
   md.myfunc(args)

2. Write definitions in main program script. Separate your main program with the command
   if __name__==“__main__”:

In Class Activity 1

In Class Activity 1

File Handling:

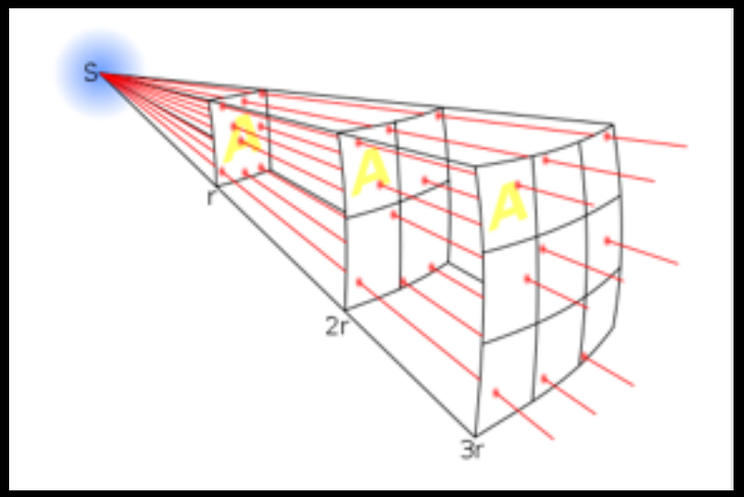
Mark will cover file input and output on Thursday, so use this code to read in your files

```
#Open file to read
f = open('state_sales_tax.txt', 'r')

for line in f:
    #Add state to list
    states.append(line[:2])
    #Get tax rate for state and remove newline character
    tax = line[3:]
    tax = tax[:-1]
    #Append to tax_rates list and
    tax_rates.append(float(tax))
    print line

#close file
f.close()
```

Background for Class Activity 2: Calculating Flux

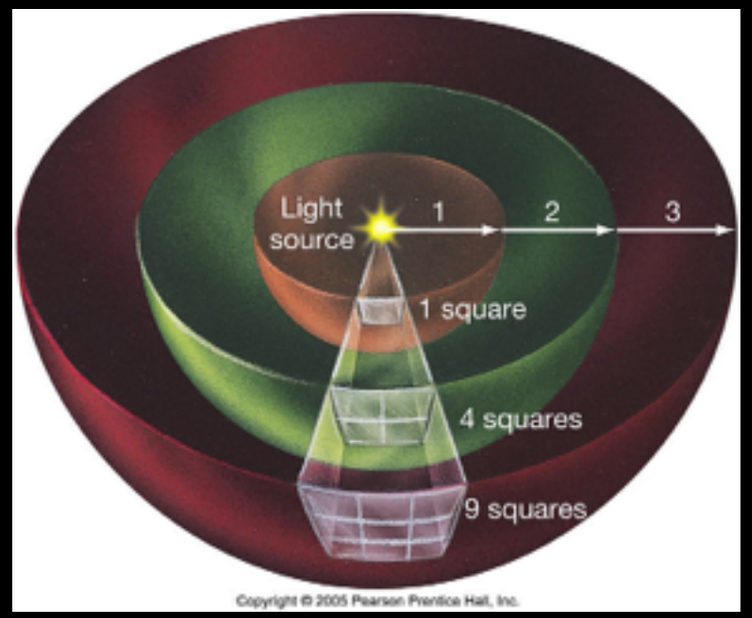
The flux or apparent brightness is the amount of power (energy/second) radiated through a given area.

Flux follows the inverse square law.



Flux of a star with luminosity, *L*, at distance *d* is:

$$F = \frac{L}{4\pi d^2}$$

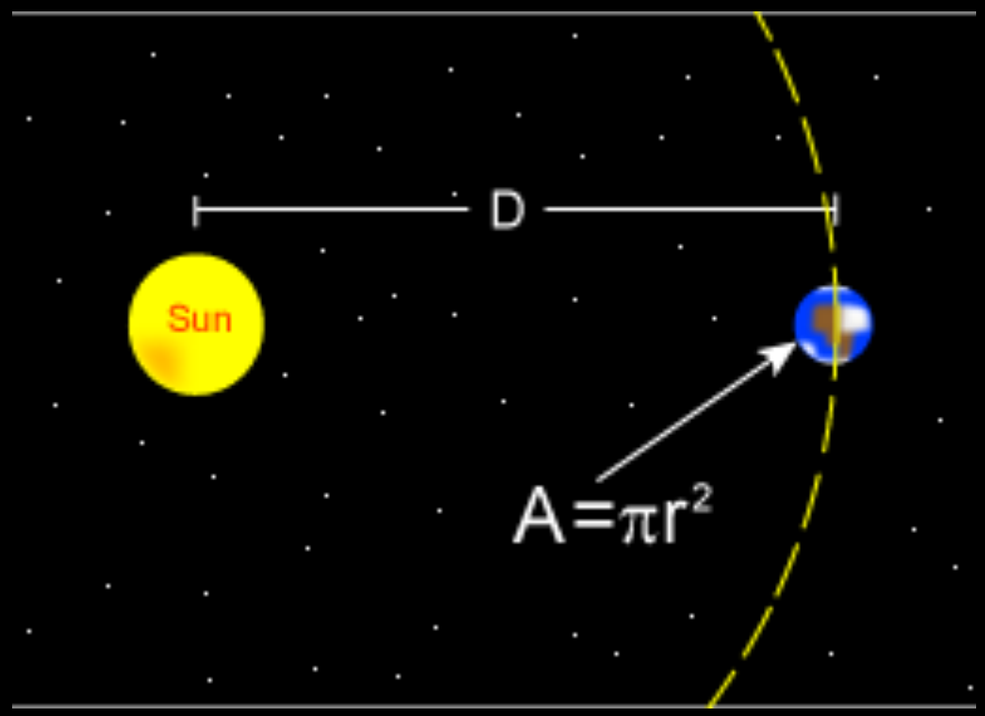


Background for Class Activity 2: Calculating energy absorbed by a planet

Power absorbed by planet with albedo *a*:

$$P_{\text{abs}} = F \times \pi R^2 \times (1 - a)$$

Flux received — *Planet cross-sectional area* — *Amount of flux not reflected* $0 < a < 1$



Assuming the flux does not vary with time (*L constant*), the total energy absorbed for a time *t* is:

$$E_{\text{abs}} = P_{\text{abs}} \times t$$

The albedo of a planet describes how much light is reflected from the planet. An albedo of 0 implies that all radiation is absorbed, an albedo of 1 implies all radiation is reflected. The Earth's albedo is 0.39.

In Class Activity 2