CS 17700
Functions, and Files

Week 5
Functions

def functionName(param1, param2, ...):
    statement #1
    statement #2
    statement #3
...

Input → F → Output
Example

def addition(a, b):
    return (a+b)

def main():
    a = 100
    b = 400
    c = addition(a, b)
    print (c)
    print(addition(10, 20))

main()
A function argument can be:
1. A value
2. Expression
3. A variable

```python
def multiplication (a, b):
    mult = a * b
    print(mult)
    return mult
def main():
    x = 2
    y = 3
    z = multiplication (4, 10)
    z = multiplication (x*2, y/3)
    z = multiplication(x, y)
```
Functions: Return Values

def square(x):
    t = x * x
    return t

def main():
    a = 10
    s = square(a)
main()

s=?
Functions: Boolean arguments and Boolean Value Returns

def canWePlay(isRainy):
    if (isRainy == False):
        play= True
    else:
        play = False
    return play

def main():
    isRainy = input("Enter rain status: ")
    play = canWePlay(isRainy)
    if(play == True):
        print ("We can play")
    else:
        print ("We should not play")

main()
def update_salary(sal, inc): # update salary with increment
    sal = sal + inc
    return sal

def main():
    sal = 100000 # initialize salary

    # increment the salary by 20000. print what is returned
    print (update_salary(sal, 20000))

    # increment the salary by 10000. Does that affect sal in main()?
    update_salary(sal, 10000)

    # increment the salary by 5000.
    sal = update_salary(sal, 5000)
Scope

```python
def update_salary(sal, inc):
    # update salary with increment
    sal = sal + inc
    return sal

def main():
    sal = 100000  # initialize salary

    # increment the salary by 20000. print what is returned
    print(update_salary(sal, 20000))

    # increment the salary by 10000. Does that affect sal in main()?  
    update_salary(sal, 10000)

    # increment the salary by 5000.  
    sal = update_salary(sal, 5000)

main()
```

They are different and their scope is different.
def update_salary(sal, inc):  # update salary with increment
    sal = sal + inc
    return sal

def main():
    sal = 100000  # initialize salary

    # increment the salary by 20000. print what is returned
    print (update_salary(sal, 20000))

    # increment the salary by 10000. Does that affect sal in main()?
    update_salary(sal, 10000)

    # increment the salary by 5000.
    sal = update_salary(sal, 5000)

main()
Functions that Modify Parameters
Lists are mutable. Passing the reference of the list

```
def update_salaries(salaryList, updateAmount):
    for i in range(len(salaryList)):
        if(salaryList[i]<90000):
            salaryList[i] = salaryList[i] + updateAmount
    return salaryList

def main():
    salaries = [75000, 90000,80000]
    print (salaries)
    update_salaries(salaries, 10000)
    print (salaries)
```
Functions that Modify Parameters

Lists are mutable. Passing the reference of the list to a function modifies the list.

```python
def update_salaries(salaryList, updateAmount):
    for i in range(len(salaryList)):
        if(salaryList[i]<90000):
            salaryList[i] = salaryList[i] + updateAmount
    return salaryList

def main():
    salaries = [75000, 90000, 80000]
    print (salaries)
    update_salaries(salaries, 10000)
    print (salaries)
```

Output: [75000, 90000, 80000]
[85000, 90000, 90000]

Why?
File Processing

- The process of *opening* a file involves associating a file on disk with an object in program memory.

- We can manipulate the file by manipulating this object.
  - Read from file
  - Write to file
File Processing

• For reading or writing a file, you need to **open** the file initially

• `open(filename, access_mode)` opens the filename

  - *Note:* if you do not provide a full path the file is assumed to be in the same directory where your program exists.
  - If you are typing directly into IDLE the file must be where python is installed. (C:\Python32\)

• **access_mode** specifies the purpose of opening the file
  - “r” means read ONLY
  - “w” means write ONLY, overwrites the file if the file exists
Methods on Files

- Syntax: `<fileobject>.method()`
  - this time files are our object

- `file = open("myfile", "w")`

- `file.read()` - reads the file as one string

- `file.readline()` - read the next line of the file as string

- `file.readlines()` - reads the file as a list of strings
  - `read()` and `readlines()` can only be used once without closing and reopening the file.

- `file.write(data)` - allows you to write to a file

- `file.close()` - closes a file
Extracting Data

• Need to extracting information from the data stored in files or webs.

• Data in files or web are **formatted as string data type**.

• We use File I/O, Strings and Lists to extract data.
# The read() operation:

```python
def main():
    myfile = open("students.txt", "r")
    content = myfile.read()
    print(content)
    myfile.close()
```

students.txt
main()
# The read() operation:

def main():
    myfile = open("students.txt", "r")
    content = myfile.read()
    print(content)
    myfile.close()

students.txt

main()

Output:

>>> Jim 75
    Alice 95
    Alex 75
    Kate 75
    John 95
    Suzan 55
    Tim 55
    Sarah 85
    Ann 95
>>> |
# The `readlines()` operation:

def main():
    myfile = open("students.txt", "r")
    content = myfile.readline()
    print(content)
    myfile.close()

students.txt
main()
Print File Contents

#The `readlines()` operation:

def main():
    myfile = open("students.txt", "r")
    content = myfile.readlines()
    print(content)
    myfile.close()

students.txt

main() >>>
['Jim 75\n', 'Alice 95\n', 'Alex 75\n', 'Kate 75\n', 'John 95\n', 'Suzan 55\n', 'Tim 55\n', 'Sarah 85\n', 'Ann 95']

Output: >>>
# The readline() operation:

def main():
    myfile = open("students.txt", "r")
    content = myfile.readline()
    print(content)
    myfile.close()

students.txt
main()
# The readline() operation:

def main():
    myfile = open("students.txt", "r")
    content = myfile.readline()
    print(content)
    myfile.close()

students.txt
main()

Output:

>>> Jim 75

students.txt

| Jim 75
| Alice 95
| Alex 75
| Kate 75
| John 95
| Suzan 55
| Tim 55
| Sarah 85
| Ann 95 |
Find Average

In this Example, the file students.txt contains the following data:

Jim 75
Alice 95
Alex 75
...

• We are interested in finding the average of the class.
Finding Average of Grades

1. Open “students.txt” for reading
2. Read the file content into a string using read().
3. Extract grades
4. Calculate the average.
5. Print out the average
Important Note

• Data in files are stored as **strings**.
• If we are interested in the **numeric value of the data**, then we need to **convert string values into numeric** using functions: int or float

**Example:**

```python
>>> x = "150"

>>> xValue = int(x)

>>> y = "159.7895"

>>> yValue = float(y)
```
def findAverage(fileName):
    file = open(fileName, "r")
    for line in file.readlines():
        # do work here!

• In each iteration of the for loop, the variable line will contain a line within the file.
• Remember in the previous slides that file.readlines creates a list, having as many elements as the lines in the file.
Find Average

By observing the list content, we see that each string contains: name space score

def findAverage(fileName):
    file = open(fileName, 'r')
    Sum = 0
    Count = 0
    for line in file.readlines():
        sublist = line.split(' ')  # fix: split(' ') instead of split(' ')
        Sum = Sum + int(sublist[1])
        Count += 1
    print (Sum/Count)  # fix: use parentheses around Sum/Count
    file.close()
QUESTIONS???
def swap(a, b):
    t = a
    a = b
    b = t
    return a, b

main():
    a = 10
    b = 20
    a, b = swap(a, b)
    print a, b

Output: ???
```python
def changePassword(cur, new):
    cur = new
    return cur

def main():
    current_password = "123456"
    print(current_password)
    changePassword(current_password, "09876")
    print(current_password)

main()
```

Output: ???

def changePassword(cur, new):
    cur = new
    return cur

def main():
    current_password = "123456"
    print(current_password)
    current_password = changePassword(current_password, "09876")
    print(current_password)

main()
1. Write a function that takes two integers as arguments and returns the value of the larger one.

2. Write a function that takes three integers as arguments and returns the value of the largest one.

3. Write a function that takes a real number as an argument and returns the absolute value of that number.

4. Write a function that takes a positive integer $n$ as an argument and returns the largest power of two greater than or equal to $n$.

5. Write a function that takes a positive integer $n$ as an argument and returns 1 if $n$ is prime, and 0 otherwise.

6. Write a function that takes a positive integer as input and returns the leading digit in its decimal representation. For example, the leading digit of 234567 is 2.

Exercise
Exercise

Write a Python script to copy one file to another

```
print "Copying from %s to %s" % (from_file, to_file)
# we could do these two on one line, how?
in_file = open(from_file)
indata = in_file.read()

out_file = open(to_file, 'w')
out_file.write(indata)
out_file.close()
in_file.close()
```