CS 17700
Variables, Assignment statement and Functions

Week 3
Announcements

- Project1
  - Will be posted Friday, September 7
  - Due Saturday, September 15th 11:58pm
  - We will post instructions on how to turn in from home
  - Note: you can always turn in from the lab machines

- Codelab:
  - Sign up for CodeLab; instructions on home page
ANY QUESTIONS?
Table of Contents

• Variables

• Assignment Statement

• Functions
  • What is a function
  • Why to use a function
  • Functions: arguments, and returned values
Variables

• Are names for objects that have values in main memory

• We define variables by assigning values to names:
  Example:
  >>>a = 9
  >>>b = 13.54
  >>>c = “Computer”

• Variables can change their values during the program

• It’s important to choose good names that describe the variable. Example: firstName, courseNo, etc.cc
Assignment Statement

Example:
\[ X = 10 \]

- Variable is also called **Left Hand Side**
- Value is also called **Right Hand Side**
Assignment Statement

Variable ← Value

- The value can be: a number, a string of characters, but also a variable, or an expression

- Example (decimal integer numbers):
  
  \[
  \begin{align*}
  X &= 10 \\
  Y &= X \\
  Y &= X + 5 \\
  Z &= Y
  \end{align*}
  \]
Example

What is the output of the following statements:

```python
>>> X = 10
>>> Y = 20
>>> result = X + Y
>>> print(result)
>>> Y = 100
>>> print(x + 10)
>>> print(x)
>>> print(y)
```

Output:

```
result: 30
X: 10
Y: 20
output:
30
20
10
100
```
Exercise 1

```python
>>> name = "John"
>>> price = 9.99
>>> total = price / 3.0
>>> name = "Jane"
>>> print(name)
>>> print(price)
>>> print(total)
```

output:
```
Jane
9.99
3.33
```
>>> var1 = "Purdue"
>>> var2 = "University"
>>> print (var1)
>>> print (var2)
>>> var2 = var1
>>> print (var1)
>>> print (var2)

"Purdue"
"University"
"Purdue"
"Purdue"

output
Purdue
University
Purdue
Purdue
What about …

>>> X + 2 = 15
>>> Y = “Hello” + 10

Tips:
• Left Hand Side should always be a variable
• You can’t perform operations (such as sum) on two values of different types (numeric and string)
Functions

• Collection of instructions that perform a task as:
  o Printing your name and course
  o Calculating the average of a set of numbers
  o Editing a picture or video
Functions

• Like in algebra, a function is a kind of “box” into which you put one value and out comes another. We represent (“denote”) a function by a name (in math we use f or F).
Why to use a function?

- If we define a function to perform a task, then we will write it **once** but we can use it (or call it) **many times**.
How to write functions?

```python
def functionName():
    statement #1
    statement #2
...
```

- Indentation is very important in Python, it marks the beginning of function body
- Python will give errors if your function is not properly indented

It is a programming practice to define a function that is called main to call the other functions in our program
def SayHello():
    print("Hello world!")

    print("--From Python")

Note: 1. Don’t forget the colon(:)

2. Align the statements in one function
Functions: Arguments

- A function may or may not receive one or more argument

No Argument

```python
def Greet():
    print("Hello Jack")

def main():
    Greet()
```

One Argument

```python
def GreetWithArg(message):
    print(message)

def main():
    msg = "Hello Jack"
    GreetWithArg(msg)
```
Functions: Arguments

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

```python
def Sum(a, b):
    total = a + b
    print(total)
def main():
    x = 5
    y = 10
    Sum(4, 10)
    Sum(x+2, y-3)
    Sum(x, y)
```

Arguments are values
Arguments are expressions
Arguments are variables
Function: Arguments

- On function call, Python assigns the value of the argument to the variable declared in function.

- When the argument passed to a function is the value of a variable, the name of that variable is irrelevant to the function.

```python
def Sum(a, b):
    total = a + b
    print(total)

def main():
    x = 5
    y = 10
    Sum(x, y)
```

- The value of `x` and `y` were put into `a` and `b` respectively via function call.

- `x` and `y` are called: local variables to function main.

- `a` and `b` are called: local variables to function `Sum`
Function: Arguments

- The name of argument passed to functions may or may not match the name of the variable used in the function.

```python
def Sum(a, b):
    total = a + b
    print(total)

def main():
    a = 5
    b = 10
    Sum(a, b)
```

- The value of `a` and `b` that are local to function `main` were put into the local variables `a` and `b` respectively via function call.
Functions: Returned Values

- Functions may return values (example: the result of a computation).

- Returned values can be:
  1. Printed
  2. Used in assignment statement
  3. Used in expression

```python
def Average(a, b):
    return (a+b)/2

def main():
    print (Average(10,2))
    avg = Average(3, 4)
    Total = Average(4,3) * 0.95
```
Functions with Multiple Returned Values

- Functions in Python may return multiple values

```python
def getabc():
    a = "Hello"
    b = "World"
    c = "!"
    return a,b,c

def main():
    a, b, c = getabc()
```
Example

def Sum(a, b, c):
    return (a+b+c)

def Greet(name, GPA):
    print("Hello", name)
    print("You have a GPA of ", GPA)

def Div(a, b):
    return a/b

def Mul(a, b):
    return a*b

def main():
    x = 3
    y = 4
    z = 2
    myStr = "Mike"
    Total = Sum(x, y, z)
    print (Greet(myStr))
    Result = Sum(x, y, z)+ Mul(a, b) - Div(y, z)
Functions that Modify Variables

- What is the output of the following program:

```python
def Bonus(grade):
    grade = grade + 10

def main():
    myGrade = 75
    print (myGrade)
    Bonus(myGrade)
    print (myGrade)

Output: 75
    75

Why?
```
Functions that Modify Variables

- **myGrade** is an argument passed to function Bonus. Only the value of myGrade matters.

- The function call will put the value of **myGrade** into **grade**

- **grade** is only known ‘locally’ to the function Bonus

- If you want to export the value from function Bonus back to main, function Bonus **MUST** use a **return** statement

- Then you can use the function call in an assignment statement
def Bonus(grade):
    grade = grade + 10
    return grade

def main():
    myGrade = 75
    print(myGrade)
    Bonus(myGrade)
    myGrade = Bonus (myGrade)
    print(myGrade)
What can go wrong?

• **If your parrot is dead, consider this:**
  o Did you use the exact same names (case, spelling)?
  o All the lines in the block must be indented, and *indented* the same amount.
  o Variables in the command area don’t exist in your functions, and variables in your functions don’t exist in the command area.
  o The computer can’t read your mind.
    ▪ It will only do exactly what you tell it to do.
    ▪ In fact, programs always “work,” but maybe not how you intended!
Final QUESTIONS???