Introduction to Programming

Spring 2022

Objectives

- Elements of a program
  - Blank spaces
  - Output statements (print function)
  - Assignment operator
  - Input (input function)
  - Simultaneous assignment
  - Definite loops

Blank Spaces

- Blank spaces
  - Sometime called white spaces
  - Spaces are irrelevant within an expression
- 4*5+10
- 4 * 5 + 10
- 4*(5+10)
- 4 * (5 + 10)
  - Blanks lines are also irrelevant

Output Statements

- Use print function
- Syntax
  - print ()
  - print (<exp1>, <exp2>, <exp3>, ..., <expn>)
  - ... denotes an indefinite series of expressions
- Semantics
  - Display information in textual form

Output Statements

- A print function can print any number of expressions
- Expressions must be separated with a comma (,)
- They will be displayed on same line
- A blank space will be inserted between expressions
- A bare print statement will print a blank line
- Successive print statements will display on separate lines

Output Statements

- Print Statements

```
print (3+4)
print (3, 4, 3+4)
print()

print (“Result is: “,
result)
print (3, 4, end=” “)
print (3 + 4)
```

- Output

Assignment Statements

- Simple Assignment
- Syntax
  - <variable> = <exp>
- Semantics
  - Evaluate the expression on right hand side of the assignment operator (=) and store it in memory location pointed to by variable

Assignment Statements

- Note:
  - = is an assignment operator – It is not equality operator

Assignment Statements

- Example
  - x = 5 + 4
  - fahrenheit = 9/5 * celsius + 32
  - x = 5
  - 5 = y
  - 4 + 7 = 9
  - name = “Joe” + “Doe”
  - name = first_name + last_name

Assignment Statements

- Variables can be reassigned as many times as you want!
- Variables are dynamically typed
  - You can store any type of data in a variable
  - You do not have to declare the type of a variable

- myVar = 0
- myVar = 7
- myVar = myVar + 1

- alpha = 4
- alpha = 14.5
- alpha = “Jack”

Assignment Statements

- Technically, this model of assignment is simplistic for Python.
- Python doesn't overwrite these memory locations (boxes).
- Assigning a variable is more like putting a “sticky note” on a value and saying, “this is x”.

Assigning Input

- Use input function
- Syntax
  - input (<prompt>)
- Semantics
  - Prompt the user for an input
  - Wait for the user to enter a value and press enter
  - It is called blocking statement
  - Read the input
  - All input is treated as string

eval Function

- What if input is a numerical value
- We can use eval(evaluated) function
- Syntax
  - eval (expression)
- Semantics
  - Evaluate the expression and return the result
  - Expression must be a string
- Example:
- eval (‘4’) eval (‘4 + 5’) eval (‘4 + x’)

eval Function

- **Beware**: the eval function is very powerful and potentially dangerous!
- When we evaluate user input, we allow the user to enter a portion of program, which Python will then evaluate.

eval Function

- Someone who knows Python could exploit this ability and enter malicious instructions, e.g. capture private information or delete files on the computer.
- This is called a code injection attack, because an attacker is injecting malicious code into the running program.
- More about this later in the course!

eval Function and input Function

- If expected input is numerical
  - We can wrap our input function inside an eval function
- eval (input(<prompt>))
- input and eval function will return the result of evaluating an expression
- We need to make sure we use assignment operator to save the value
- variable = eval (input(<prompt>))

Simultaneous Assignment

- Several expression can be calculated at the same time and assigned to variables
- <var>, <var>, ... = <expr>, <expr>, ...
- Evaluate the expressions in the right side and assign them to the variables on the left side.

Simultaneous Assignment

- Example
- a, b = 4, 5
- sum, diff = x + y, x – y
- x, y = y, x

Simultaneous Assignment

- We can use this same idea to input multiple variables from a single input statement!
- Use commas to separate the inputs
- Example

```
score1, score2 = eval(input("Enter two scores separated by a comma: "))
```

Definite Loops

- Loops are used for repeating a set of statements multiple times
- Definite loops are easiest to use.
- A definite loop executes a definite number of times
  - At the time Python starts the loop it knows exactly how many iterations to do.

Definite Loops

- Syntax
  - for <var> in <sequence>:
    <body>
- Semantics
  - Assign members of the sequence to var (one at a time)
  - Repeat the body of loop as many time as there are elements in the sequence

Definite Loops

- Sequence can be a list of elements enclosed in []
- Sequence can be specified using range function

Definite Loops

- for loops alter the flow of program execution, so they are referred to as control structures.

Definite Loops

- Examples

```
for i in [0,1,2,3]:
    print (i)

for odd in [1, 3, 5, 7]:
    print(odd*odd)

for i in range (10):
    print (i)
```