Iteration

CSC 1200 - Principles of Computing

Overview

- Multiple Assignment
- Updating Variables
- Increment and Decrement Operators
- While Statement (Loop)
- Infinite Loops
  - The Break Statement
- Algorithms
- Default Parameters

Multiple Assignment

- It is legal to make more than one assignment to the same variable.

```
ave = average( 3, -5 )
x_coord = ave
ave = average( 5, 1 )
y_coord = ave
```

- The assignment operator is different from "=" in a math equation.
  - In math, = is symmetric. So, if you can write a = 7, then it is also legal to write 7 = a. This is NOT the case with the assignment operator.
  - In math, = is permanent. If a = 7, then a will always be 7. This is NOT the case with the assignment operator. a is 7 only until another assignment operator changes the value of a.
  - In math, both sides of the = must represent equivalent quantities. This is NOT the case with the assignment operator. In Python, we often write assignment statements like x = x + 3

Updating Variables

- One of the most common forms of multiple assignment is an **update**, where the new value of the variable depends on the old value.
  - Increment: increase the value of the variable by 1
    - x = x + 1
  - Decrement: decrease the value of a variable by 1
    - x = x - 1
- If you try to update a variable that does not exist, you will get an error. Before you can update, you must **initialize** the variable with a value.

updating variables.py - C:/Users/bgann/App

File Edit Format Run Options Window Help

```
def sum_from( m, n ):
    for num in range(m, n):
        sum = sum + num
    return sum

sum = sum_from( 2, 10 )
print( sum )
sum = sum_from( 4, 6 )
print( sum )
sum = sum_from( -3, 3 )
print( sum )
```

```
Traceback (most recent call last):
  File "C:/Users/bgann/AppData/Local/Programs/Python/Python310/Ch 7/updating var
iables.py", line 6, in <module>
    sum = sum_from( 2, 10 )
  File "C:/Users/bgann/AppData/Local/Programs/Python/Python310/Ch 7/updating var
iables.py", line 3, in sum_from
    sum = sum + num
UnboundLocalError: local variable 'sum' referenced before assignment
```

Increment and Decrement Operators

Increment and Decrement are such common operations that there are special operators to perform them.

x += i ← increment x by i (equivalent to x = x + i)

x -= d ← decrement x by d (equivalent to x = x - d)

And closely related are…

x *= n ← x = x * n

x /= n ← x = x / n

- Note to C++ and Java programmers: There is no ++ operator!

The While Statement (Loop)

- We have used a for statement (loop) to repeat the same statements multiple times.
- Another statement used for repetition is the while statement (loop)
  - General format of while loop example:

```
while condition:
<tab> statement(s)
```

```
n = 10
while n > 0:
        print( n )
        n -= 1
print( 'Blastoff!')
```

Infinite Loops

- The body of the loop should change 1 or more variables that affect the condition of the loop so that eventually the condition becomes false, and the loop terminates.
- If the condition of the loop never becomes false, we will have an **infinite loop**.

```
def countdown_by_2( n ):
    while n != 0:
        print( n )
        n -= 2
    print( 'Blastoff!' )

start = int(input('Enter an integer: '))
countdown_by_2( start )
```

```
Enter an integer: 10
10
8
6
4
2
Blastoff!
>>>
```

If I enter 11, after a couple minutes it's still going strong…

```
-100399
-100401
-100403
-100405
-100407
-100409
-100411
-100413
-100415
-100417
-100419
-100421Traceback (most recent call last):
  File "C:/Users/bgannod/AppData/Local/Programs/Python/Python39/Ch 7/Infinite Lo
op.py", line 8, in <module>
    countdown_by_2( start )
  File "C:/Users/bgannod/AppData/Local/Programs/Python/Python39/Ch 7/Infinite Lo
op.py", line 3, in countdown_by_2
    print( n )
KeyboardInterrupt
>>>
```

The Break Statement

- You can use the break statement to exit the loop.

Example: Keep track of the non-numeric entries

```
print('Enter numbers or words. Enter DONE when you are finished.')
words = ''
while True:
    user_in = input('--> ')
    try:
        num = float(user_in)
    except ValueError:
        if user_in == 'DONE':
            break
        else:
            words = words + user_in + ' '

print('The non-numeric entries were:', words )
```

```
Enter numbers or words. Enter DONE when you are finished.
--> Forty
--> 40
--> is
--> -23
--> 23.56
--> a
--> -78.3
--> number
--> DONE
The non-numeric entries were: Forty is a number
```

The Same Loop Without a Break

```
print('Enter numbers or words. Enter DONE when you are finished.')
words = ''
user_in = input('--> ')
while user_in != 'DONE':
    try:
        num = float(user_in)
    except ValueError:
        words = words + user_in + ' '
    user_in = input('--> ')

print('The non-numeric entries were:', words )
```

```
Enter numbers or words. Enter DONE when you are finished.
--> I
--> -42
--> 2.3
--> don't
--> 357
--> need
--> a
--> 0.3
--> break
--> DONE
The non-numeric entries were: I don't need a break
```

Algorithms

An algorithm is a mechanical process for solving a category of problems.

- Examples:
  - adding numbers (with carry)
  - subtracting numbers (with borrow)
  - multiplying multi-digit numbers
  - dividing numbers using long division
- Algorithms do not require “intelligence” to carry out – just follow the rules step by step.
- DESIGNING algorithms *does* require intelligence and is a central part of computer programming.

Default Parameters

- The parameters in a Python function can be given default values using assignment to a parameter.
  - If the programmer doesn't give an argument, the default will be used automatically.
  - Any parameters with default values must go LAST in the parameter list.

Example:

```
def greet( name, message = 'Nice to meet you.'):
    print('Hello, ' + name +'. ' + message)


greet( 'Aiden' )
greet( 'James', "I've heard so much about you." )
greet( 'Greg', 'Long time no see!' )
greet( 'Tristen' )
```

```
Hello, Aiden. Nice to meet you.
Hello, James. I've heard so much about you.
Hello, Greg. Long time no see!
Hello, Tristen. Nice to meet you.
>>>
```

Programming Practice

- Write a program that asks the user for an integer, n. Then find the sum of the first n odd integers.
- Modify the program to find the sum of n consecutive odd integers starting at a user-specified value.