CS 106
Introduction to Computer Science I

07 / 07 / 2021

Instructor: Michael Eckmann

Today's Topics

- Recap
  - Functions: print, input, int
  - Variables and value types (int, str, float, bool)
  - Operators: = (assignment), + (concatenation or addition), / (division)
- Other arithmetic operators: -, %, *, //, **, ( )
- If else statements

Recap

- Variables and how to assign values to them
- = is assignment operator
- + concatenates two str's together
- + is addition when used on numeric values
- / is division

Recap

- Printing: print is a function that takes in any number of arguments, separated by commas. sep and end are two parameters and we can change their default values
- Keyboard input: input is a function that takes a str as an argument (which is output to the screen) and waits for user input until enter key is pressed and returns what was typed as a str
- Converting str to int: int is a function that takes a str as an argument and returns the int version of that string

Examples

```
user_name = “Jane”
print(“Hello”, user_name)


user_age = 21
print(user_name, “will be”, user_age + 10,
   “years old in 2031”)
```

Input example

```
height = input("How many feet tall are you?")

height = int(height)
# above line converts the str value to int

print("there are", height*12, "inches in",
   height, "feet.")
```

Program Flow

- Python programs start executing at the top and they execute sequentially, in order (unless something changes the program flow --- more on that later today and throughout the semester) until the end of the source code file which is when the program ends running.

Arithmetic operators

We already used + and /

- is subtraction

* is multiplication

// is division resulting in an int

(whereas / is division resulting in a float)

% is modulus (remainder)

** is exponentiation

All work on two operands (one on the left of the operator and one on the right)

Operands could be variables OR values

The modulus operator (remainder)

- **%** is the modulus operator

  val1=16

  val2 = 5

  remain = val1 % val2

- The modulus operator results in the remainder after **val1** is divided by **val2**.
- **remain** would have the value 1 because 16//5 is 3 with a remainder of 1
- val1//val2 results in 3 (because // is int division)
- val1/val2 results in 3.2 (/ is float division)

Arithmetic Operator precedence levels

- **( )** parentheses are evaluated first
- if parens are nested then the innermost pair is evaluated first.
- *, /, //, **%** multiplication, division and modulus are next --- if several of these, they are evaluated **left to right**.
- +, **-** addition and subtraction are last --- if several of these, they are evaluated **left to right**
- It's useful to memorize this precedence list

Arithmetic Operator precedence example

some_result = (5 + 4 - 1) * 6 + (2 - 10)

- *(correct way)*
- 5+4-1 → 8,
- 2-10 → -8,
- 8*6 → 48,
- 48-8 → 40

Equality and Relational operators

==	is equal to operator
!=	is not equal to operator
<	is less than operator
>	is greater than operator
<=	is less than or equal to operator
>=	is greater than or equal to operator

if/else statements

An if/else statement syntax is

```
if bool_expr:
    # do something if bool_expr is True
    # do something else if bool_expr is True
else:
    # do something if bool_expr is False
    # do something else if bool_expr is False
```

Notice the colons that is part of the syntax of if/else statements. Also, the code inside if and the else portions are indented. Indentation matters in python

else is optional

1 or more statements can be inside the if or the else portion

if / else

- bool_expr is something that will evaluate to True or False (usually using relational or equality operators.)
- After the if, one can have arbitrary many elif bool_expr: portions.
- Let's write some examples and use some relational and equality operators
  - An if alone
  - An if and an else
  - An if, elif, elif, else
  - An if, elif, elif

Do lab work now

- You should now do lab 01

Pseudocode

- pseudocode is an informal use of English to describe what a program is to do and in what order
- pseudocode is not an actual computer programming language
- it is used prior to writing actual code to help the programmer in the planning stages of a program

Example Application Exercise

- write a program to compute the number of projected home runs a baseball player will hit for the season based on how many homers he's hit so far.
- Output should look like:

  player's name is projected to hit *number* home runs in 162 games.

- Any ideas?

Pseudocode for our example

get player's name from the user

get the number of homeruns so far

get the number of games played so far

compute the number of projected homeruns for this player based on a season of 162 games by using the following calculation

$$\frac{\text{projected homers}}{162} = \frac{\text{homers so far}}{\text{games played so far}}$$

Pseudocode for our example
(continued)

from this equation,

$$\frac{\text{projected homers}}{162} = \frac{\text{homers so far}}{\text{games played so far}}$$

we can multiply both sides of the equation by 162 and get

projected homers =
    homers so far * 162 / games played so far

Pseudocode for our example
(continued)

Print out the following with actual values for *player's name* and *number*

player's name is projected to hit *number* home runs in 162 games.

Pseudocode could be more fleshed out than what we have done here --- use as much or as little detail in pseudocode as you prefer.

Pseudocode for our example (continued)

- Now we can write the program based on our pseudocode.