A Deeper Introduction to Python

CMSC 201 Section 60
September 4, 2024

Administrative Notes

Section 61 reports to discussion starting next Monday

Today’s lecture

There are a lot of things you need to know to be productive programming in Python. We’re going to cover a lot of them today.

We’ll summarize what you need to take away from this lecture at the end.

Terms you need to know

Syntax refers to the letters, words and punctuation you use

Semantics refers to the underlying meaning of the letters, words and punctuation

You have to get both the syntax and the semantics right for the program to work the way you want it to.

More Terms

Comment

Variable

Literal

Constant

Reserved Word

Type

Comments

Notes you write for yourself and other humans to explain what it is you're trying to do

Mark a ***comment*** in Python with a #

Anything after the # on a line is a comment and is not executed by the python interpreter

#this is a comment

print ("hello, world") # this is also a comment

"""

This is a block comment - you can write anything you want

Between these sets of double-quote marks

"""

Literals

A ***literal*** is an explicit value that is to be taken exactly as it is written in the program.

“Hello, world” is a literal (it’s a string literal, to foreshadow)

3 is a literal

3.14159 is a literal

2.71828 is a literal

Magic Number

A ***magic number*** is a number - either an integer or a floating point number that you use in your code that doesn’t obviously make sense

42

Don’t use them

Declare a constant

Symbol for something that doesn’t change

Variables

A ***variable*** is a symbolic name associated with a value

The value may change during the program's execution

When you use a variable in python, the python interpreter reserves a location in memory and associates the variable's name with that location. Then, any values assigned to that variable are stored to and read from that location

Some examples to come later in the lecture

Constants

A ***constant*** is a symbolic name associated with a value that *WILL NOT CHANGE* during the execution of a program

You use a constant to represent a “magic number” to make code more readable

```
print(3.14159)   # or
PI = 3.14159
print(PI)
```

Python does NOT provide built in support for constants
We use ALL CAPS to represent a constant

Reserved Words

```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

These words have special defined semantics in Python - you can’t use them as variable names or for any other purpose

These are Case-Sensitive.

False is not the same as false

Types

Variables and constants have defined types that determine what operations you can perform on them.

For now, we'll deal with ***int, float, string***, and ***boolean***

int - integer - whole numbers. You can do math on them

float - floating point numbers - integer part and decimal part - you can do slightly different types of math on them

string - zero or more characters treated as a whole

boolean - have the value True or False (note case sensitivity)

Using variables

Declaring variables: unlike some other languages, you do not pre-declare a variable in Python. When you use a variable, that declares it

- The python interpreter recognizes that you have just declared a new variable and allocates a memory location to store its value

But you do have to initialize a variable before trying to use it

- Initialize means assign it a value

A very common error is to try to use an uninitialized variable - use it before it has been assigned a value

How Variables work

- A simplified description of the Python symbol table

Memory - a whole bunch of places to store bits

Each location has an address, and holds some number of bits

a

Symbol (variable name)	Type	Address
a	String	140241794348336

Bits: 1 or 0

More terms

Mutable - able to be changed

Immutable - not able to be changed

Python ints, floats, strings, and booleans are immutable. Once you assign a value to a location in memory associated with a variable of those types, you can never change it.

You need to get a new location in memory.

(Yes, there are some examples)

Python rules for variable names

- Python variable names can contain:
  - Uppercase letters (`A-Z`)
  - Lowercase letters (`a-z`)
  - Digits (`0-9`)
  - Underscore (_)
- Variable names can’t contain:
  - Special characters like `$`, `#`, `&`, `^`, `)`, `(`, `@`

More rules for variables

- variables can be any length.
  - `x`
  - `WhoWillbethenextPresidentofUMBC`
    - ***Which would be better as*** `Who_Will_be_the_next_President_of_UMBC`
  - **`myName`**

- Variables cannot **start** with a digit.
  - **`2cool4school`** is not a valid variable
  - **`cool4school`** is a valid variable

CMSC 201 convention

Long variable names should use snake_case, not camelCase

Snake_case means separate words with underscores

`highest_grade_in_class` # snake_case

camelCase - run words together but use uppercase for the first letter in each word

`highestGradeInClass` # CamelCase

Hello, World

It’s kind of a tradition that when you learn to program in a new language, the first thing you do is make the computer print “Hello, World” to the screen.

We did this last Wednesday, but let’s do it again

What if I wanted to do that using a variable?

“Hello, World” as we just used it is a literal

Suppose we wanted to write the same program using a variable this time?