Statements, Variables, Types and Functions

Let's start programming!

- We will be using Python 2.7 as our programming language
- To run our programs, we will use the Python Interpreter
- Please follow the set up instructions if you have not already done so
  - https://github.com/TranscodeSF/classwork/blob/master/preclass_setup.md
- Even though we are using Python, these concepts are pretty universal, regardless of what programming language you use.

Our First Python Program

```
print "Hello World!"
```

Running our program

- Open Terminal
- Type “python” and hit enter
- Then type in the program from the last slide
- Hit enter again and the Terminal should output “Hello World!”

- For example:

```
>>> python
>>> print “Hello World!”
Hello World!
```

Statements

- A statement is a single line of logic
- Our first program used just one statement, the “print” statement
- The logic of our statement was to write the string “Hello World!” to standard output

Keywords

- Keywords (also known as reserved words) are the words (called identifiers) that make up a computer language
- When naming your own identifiers, you cannot use keywords

Python Keywords

'and’	'elif’	'if’	'print’
'as’	'else’	'import’	'raise’
'assert’	'except’	'in’	'return’
'break’	'exec’	'is’	'try’
'class’	'finally’	'lambda’	'while’
'continue	'for’	'not’	'with’
'def’	'from’	'or’	'yield'
'del’	'global’	'pass’	

Identifiers

- Identifiers are names used to identify things to refer to them in the program
- Identifiers start with a letter or underscore and are followed by 0 or more letters, underscores, and digits

- For example:

hello

hello_world

hw3

Variables

- Variables are used to store values
- A variable is an identifier associated with a storage location that contains a value
- The identifier is used to reference the value

- For example:

```
>>> hello = “Hello World!”
>>> print hello
Hello World!
```

  - The variable is “hello” and the value is “Hello World!”

Changing Variables

- They are called “variables” because they can change

- For example:

```
>>> greet = “Hello World!”
>>> greet = “Goodbye World!”
>>> print greet
Goodbye World!
```

  - The variable “greet” changed its value from “Hello World!” to “Goodbye World!”

Variables and Types

- Values of variables have types
- Python is a dynamically typed language
  - It doesn't care about changing a variable from one type to another
  - It doesn't need to know what type your variable is ahead of time

Types

- You can check the type of an object by using Python's built in type function

- For example:

```
>>> type("Hello World!")
<type 'str'>
```

- You can read more about types in Python at https://docs.python.org/2/library/types.html

Common Types

- Listed below are some common types in Python, however there are many more
  - str
  - int
  - float
  - bool
  - tuple
  - list
  - NoneType

Strings

- str, short for string, is the type for words
- Strings are made of characters enclosed in matching apostrophes or double quotes

- For example:

>>> “Hello World!”

>>> ‘Hello Wolrd!’

String Escape Characters

- The backslash (\) character is used to escape characters that otherwise have a special meaning, such as:
  - newline (\n)
  - backslash itself (\\)
  - the quote character(\”)

- For example:

```
>>> my_string = "Hello,\n\"World\"!”
>>> print my_string
Hello,
"World"!
```

Ints

- int, short for integer, is the type for whole numbers

- For example:

```
>>> 3
>>> 1000
>>> -2
```

Floats

- float, short for floating point number, is the type for numbers with a decimal

- For example:

```
>>> 0.1
>>> 3.14159265359
>>> -11.0
```

Bools

- bool, short for Boolean, is a type that has only two values

- For example:

>>> True

>>> False

Tuples

- tuple is a type that is a sequence of values
- The values are separated by commas and enclosed in parenthesis
- The sequence is immutable (cannot be changed)

- For example:

>>> (1, 2, 3, 4, 5)

>>> (‘hello’, ‘world’)

Lists

- list is a type that is a sequence of values
- The values are separated by commas and enclosed in square brackets
- The sequence is mutable (can be changed)

- For example:

>>> [1, 2, 3, 4, 5]

>>> [‘hello’, ‘world’]

None

- NoneType is a type used to represent the absence of value
- It has only one value: “None”
- It is used as a default when no value is specified

- For example:

```
>>> None
```

Functions

- A function is a sequence of statements that performs a specific task and returns a value
  - By default, functions in Python return the value "None"
- Python has many built-in functions
  - We used Python's built in "type" function earlier
- You can also define your own functions

Defining Functions

- Functions, like variables, use identifiers
- The first line of a function is the “def” keyword, followed by (a space then) the identifier, followed by parenthesis and a colon
- Each statement in the function follows on its own line and should be indented with 4 spaces

- For example:

```
def hello():
    print "Hello World!”
```

Returning Values from Functions

- The “return” statement is used to return a value from a function
- The “return” statement exits the function
  - You can only return once in a function
  - It should be the last statement in the function
- Only one value can be returned from a function
  - To return multiple values, use a tuple

- For example:

```
def hello():
   return "Hello World!”
```