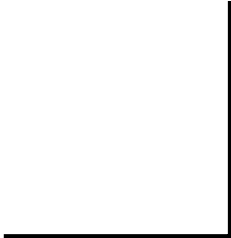


Lab 6: More Python



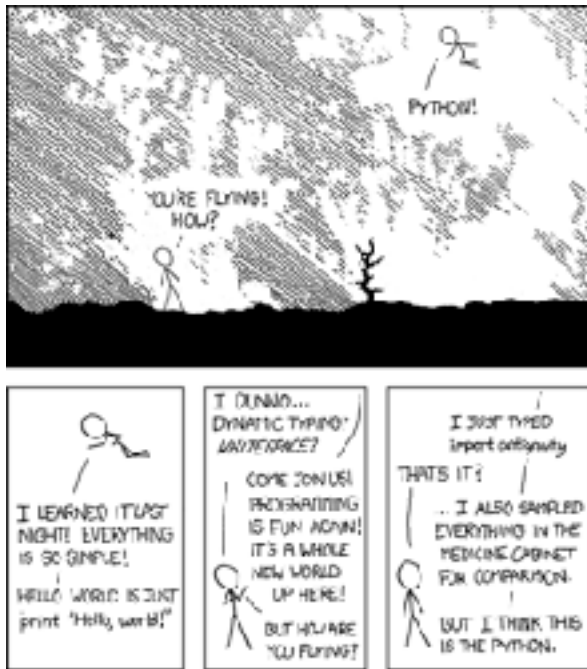
Command Line Input

- `sys.argv`
 - This module works exactly like C `argv` works. Except you don't pass it as an argument, it is available through the `SYS` module
 - `sys.argv` is a list
 - There is no `argc` variable, however, you can just get the size of the list:
 - `argc = len(sys.argv)`
 - The module `argparse` is a fancier version of `SYS.argv` (Google it!)
 - <https://docs.python.org/3/howto/argparse.html>
- `sys.stdin`
 - To get user input for your program, you can use `SYS.stdin`

```
for line in sys.stdin:  
    do_something_with(line)
```

Importing modules

- You can import libraries (and rename them!)
 - Rename the library **argparse** as **ap**
 - `import argparse as ap`
- You can also import just the stuff you want
 - Only load **argv** from the library **sys**
 - `from sys import argv`
- Remember, if the module is not installed in ECS 238
- Google will be your friend
 - Python is very popular, there are a lot of online tutorials and guides



Function

- Any code that exists at the outermost indentation level is considered *global*
 - Better idea, use functions!
- Functions are defined using **def**

```
def main():                # ':' indicates change in scope
    print("I ran from the command line")
```

```
if __name__ == "__main__": # Everything on the leftmost indent is global
    main()
```

Functions

- Order matters because there are no function prototypes
 - The function you want to use must be defined earlier in the file before you use it
- Parameters
 - Immutable Datatypes are call-by-value
 - Mutable Datatypes are call-by-reference
- You can set default values for your function arguments
- You can have zero or more return values (of any datatype)
 - If zero return values, Python returns None

```
def my_function_name(arg1, arg2="default_val"):  
    printf("I am in the function scope")  
    foo = arg2  
    bar = arg1  
    return foo, bar
```

Dictionaries

- Dictionaries are created using curly-brackets
 - `A = { key1 : value1, key2 : value2 }`
 - `A = {}`
- They consist of a set of Key-Value pairs
 - There are no restrictions of the values (they can be anything)
 - No duplicate keys are allowed, and a key must be **immutable**
 - Valid keys would be *strings, tuples, numbers*.

Dictionaries

- Dictionaries are like lists in that they are **mutable and unordered!**
 - `A[key1] = value3` # {key1 : value3, key2 : value2}
 - If the key doesn't exist it will be added
 - `x = A[key3]` # `KeyError` if key3 doesn't exist in the dictionary
 - `del A[key1]` # {key2 : value2}
- To loop through a dictionary:

```
for key, value in A.items():  
    # do something here...  
    print(key, value)
```

Dictionary Methods

- Get list of keys: `D.keys()`
 - Get list of value: `D.values()`
 - Get list of (key,value) tuples: `D.items()`
 - Remove all elements: `D.clear()`
 - For more examples on lists, tuples, dictionaries:
 - <https://docs.python.org/3/tutorial/datastructures.html>
-
- Remember, like lists, if you modify a dictionary in a function it will be modified outside the function!

List Comprehension

- List comprehensions are a tool for transforming lists (or anything iterable, really) into another list
- An easier (and faster) way of doing the for-loop ...

```
A = [ 2*x for x in range(0,7) if x % 2 == 0]
```

```
# A → [0, 4, 8, 12]
```

Examples

- Change list values
 - `doubled = [n * 2 for n in range(0,10)]`
- Nested loops
 - `matrix = [[1,2],[3,4]]`
 - `flattened = [n for row in matrix for n in row]`
- Dictionaries
 - `_dict = {1:2, 3:4} ;`
 - `flipped = {value: key for key, value in _dict.items()}`

File I/O - Opening

- Opening a File

```
file = open("filepath.txt", "r") # use "w" to open for write
# do something with the file here...
file.close() # when done
```

Or

```
with open("filepath.txt", "r") as file:
    # Do something with file here
    # file closes automatically after end of 'with' scope
```

File I/O - Opening

- Mode options
 - **r** → open for reading
 - **w** → open for writing. Truncates files
 - **b** → binary mode
 - **a** → open for writing (appending). Starts at end of file

File I/O - Reading and Writing

- Get File contents after opening (many options)
 - `contents = file.read()` # reads entire file contents
 - `line = file.readline()` # reads a line at a time until EOF
 - `lines = file.readlines()` # reads all lines into a list
 - `for line in file:` # loops through each line until EOF
- Writing
 - `file.write("this line will be written to the text file\n")`

In-Lab Activity: Word Count

Given input from an file path provided at the command line, we are going to count the number of occurrences of each word in the file and format the output to print the top 10 results.

To run the program:

```
python word_count.py input.txt
```

Results for the provided input.txt file:

```
the : 4387  
and : 3931  
of : 2698  
to : 2364  
I : 1605  
a : 1289  
that : 1265  
in : 1168  
with : 0945  
my : 0880
```

In-Lab Activity: Isogram

Determine if a string is an isogram

An isogram is ***word or phrase without a repeating letter***

Examples:

- lumberjacks
- background
- downstream

A test script has been downloaded from exercism.io and an empty function for you to code your solution.