NETWORK PROGRAMMING IN PYTHON

INTRODUCTION TO PYTHON PROGRAMMING II

By: Isaac Armah-Mensah & Obed Ademang

WHAT WILL BE COVERED

- Part 2:
  - Control Flow Statements 〽️
  - Advanced Arithmetic Operators 🕹️
  - I/O Extended 🗿
  - Functions 🌟
  - OOP Ninja 🎯
  - Git Crash Course 👾

I/O EXTENDED 🗿

COMMAND LINE ARGUMENTS

```
import sys

...
sys.argv
```

- command line argument is treated like a list
- the item at index **0** is the file name

Write a program that accepts two or more arguments and prints them back to the user. Make the program more intuitive to the user.

PYTHON IN BUILT I/O

```
var1 = input("Whatever is placed in here is a string")
```

- be reminded that anything that the user inputs from here is a string and it is utf-8 character encoded.
- input function in Python v2.x is different from input function in Python v3.x
- `input` in v2.x is evaluated whiles in v3.x to get the same behaviour you do `eval(input("....."))`
- To just return what the user typed use `raw_input()` and `input` in v2.x and v3.x respectively.

CONTROL FLOW STATEMENTS

IF STATEMENTS

```
if boolean_expression1:
    suite1
elif boolean_expression2:
    ... suite2
elif boolean_expressionN:
    suiteN
else:
    else_suite
```

- Python unlike other programming languages do not have a `switch ... case` statement.
- There are several implementations of this behaviour though. You can explore along this end to see which implementation works best.

WHILE STATEMENT

The while statement is used to execute a suite zero or more times, the number of times depending on the state of the while loopâ€™s Boolean expression. Here is the syntax:

```
while boolean_expression:
    suite
```

- *Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.*
- *Write a program that accepts a sequence of numbers between 200 and 5000 inclusive as input and prints their sum else it print the sum and then the number which was entered*

FOR ... IN STATEMENT

```
for variable in iterable:
    suite
```

- *Write a program to iterate through the English alphabets, printing whether they are consonants or vowels.*
- *Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200 (both included). The numbers obtained should be printed in a comma-separated sequence on a single line.*

BASIC EXCEPTION HANDLING

- An exception is an object like any other Python object, and when converted to a string (e.g., when printed), the exception produces a message text. A simple form of the syntax for exception handlers is this:

```
try:
    try_suite
except exception1 as variable1:
    exception_suite1
    ...
except exceptionN as variableN:
    exception_suiteN
```

ADVANCED ARITHMETIC OPERATORS

FUNCTIONS 🌟

```
def add(a, b):
    sum = a + b
    return sum


def difference(a, b):
    diffy = a - b
    return diffy


if __name__ == "__main__":
    x = int(input("Enter a: "))
    y = int(input("Enter b: "))

    print("Sum: {}".format(add(x, y)))
    print("Difference: {}".format(difference(x, y)))
```

OOP NINJA 🎯

```
class Student:
    def __init__(self, first_name, last_name, index_number, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age

    def get_name(self):
        return "{} {}".format(self.first_name, self.last_name)

    def get_index_number(self):
        return "{}".format(self.index_number)

if __name__ == "__main__":
    stud_1 = Student("Obed", "Ademang", "PS/ITC/11/0035", 10)
    print(stud_1.get_name())
    print(stud_1.get_index_number())
```

GIT 👾

SETUP

Download git for MAC OSX

Download git for Windows

Download git for Linux

create a new repository

create a new directory, open it and perform a

`git init`

to create a new git repository.

checkout a repository

create a working copy of a local repository by running the command

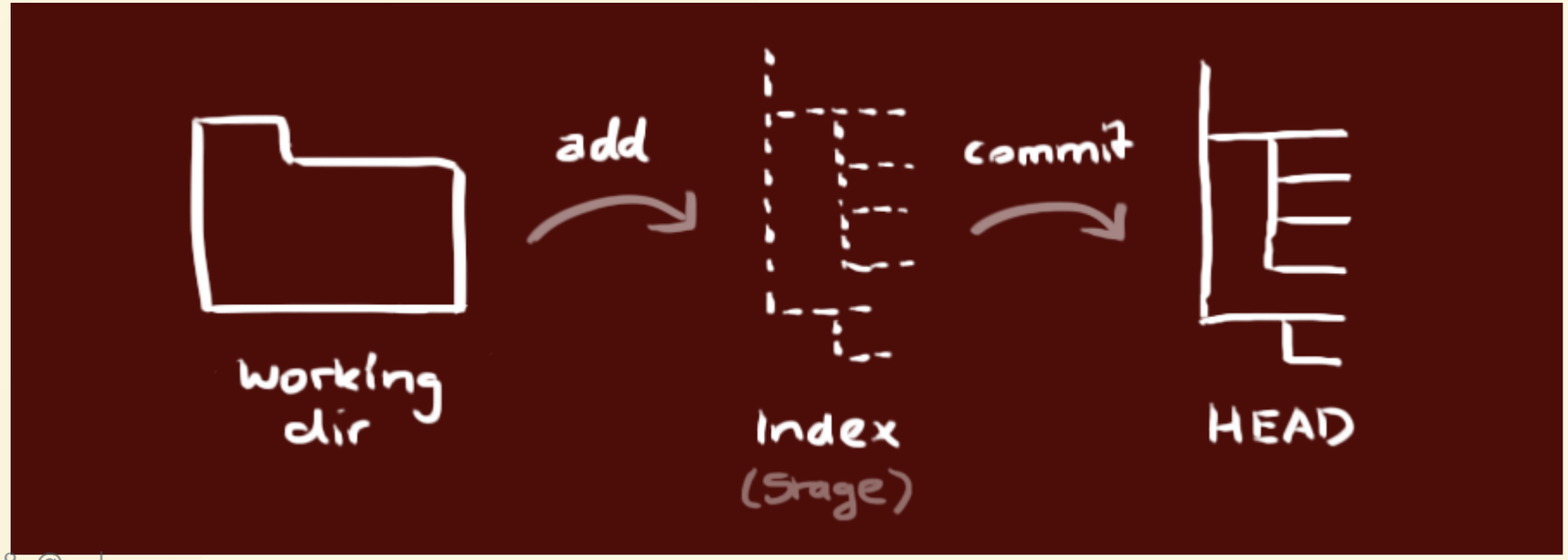
```
git clone /path/to/repository
```

when using a remote server, your command will be

```
git clone username@host:/path/to/repository
```

workflow

your local repository consists of three "trees" maintained by git. the first one is your `Working Directory` which holds the actual files. the second one is the `Index` which acts as a staging area and finally the `HEAD` which points to the last commit you've made.



add & commit

You can propose changes (add it to the Index) using

```
git add <filename>
```

```
git add --all
```

This is the first step in the basic git workflow. To actually commit these changes use

```
git commit -m "Commit message"
```

Now the file is committed to the HEAD, but not in your remote repository yet.

pushing changes

Your changes are now in the HEAD of your local working copy. To send those changes to your remote repository, execute

```
git push origin master
```

Change master to whatever branch you want to push your changes to.

If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

```
git remote add origin <server>
```

Now you are able to push your changes to the selected remote server

References

- Summerfield, Mark. Programming in Python 3 : A Complete Introduction to the Python language / Mark Summerfield.â€”2nd ed.
- Python v3.6.0 Documentation
- Python v2.7.13 Documentation
- Git - The simple guide - no deep shit

> **It's not at all important to get it right the first time. It's vitally important to get it right the last time.**
>
> Andrew Hunt & David Thomas

> **First, solve the problem. Then, write the code.**
>
> John Johnson