AY 2017 Term 2 – Python-Minecraft Lessons and Reflections

Unit 1 – Introduction to NyokaTofali

- Have students start by double-clicking the NyokaTofali shortcut on their desktop.
- Practice switching between Python and Blocks views.
- In Blocks view, have them drag a *print_st*r block onto the page and enter their name.
- Have them click Python view to see what the block looks like in Python.
- In Python view, have them click Run to see the program run.
- Have them go back to Python view to add another *print_str* block under the first, with whatever text they want inside.
- Have them go back to Python view and run the program.
- If time remains, have them continue to play with the *print_str* block, run them, etc.
- **New blocks to enable in NyokaTofali:**
  - *print_str* | print(" ")
- **Reflections (July 18th, 2017)**
  - I gave Form 1 an entire class (40 mins) to do this, whereas I combined this with Unit 2 in the same class for Form 2. Even so, both Forms had no problems with this, since Form 1 has already been exposed to block programming, whereas Form 2 is familiar with the basics of Python.

Unit 2 – Variables in NyokaTofali

- Have students open NyokaTofali
- Drag a *var_store* block onto the screen for storing number of Form 1 students, ie. *form1 = 52*
- Switch to Python view, run, and note that nothing seems to happen.
- Drag a *print_generic* block to print the *form1* variable, switch to Python view, run.
- Drag another *var_store* block on for storing number of Form 2 students, ie. *form2 = 51*
- Drag another *print_generic* block to print the *form2* variable.
- Now, drag one final *print_generic* block and have them enter *form1 + form2* inside
- Before running, ask them what they think the program will do.
- Have them run to confirm.
- If time remains, have them continue to compose programs using all blocks available to them.
- **New blocks to enable in NyokaTofali:**
  - *var_store* | __ = __
  - *print_generic* | print(____)
- **Reflections (July 19th, 2017)**
  - I gave Form 1 an entire class (40 mins) to do this, whereas I combined this with Unit 1 in the same class for Form 2.
  - Although it was heavily guided, Form 1 had no problems with this unit, despite the introduction of the concept of variables.

Unit 3 – Guided Experimentation in Minecraft

- Show students how to open Minecraft and enter Offline Mode
- Explain that they can play Singleplayer mode by themselves in the future, but that during class we will be using Multiplayer mode.
- Have them join "LocalServer" to enter the world and explore it.
- Over the course of these two classes, **gradually** explain/reiterate the following controls:
  - W, A, S, D to move around
  - Double tap and hold W to sprint

- Space bar to jump
- Double tap space to fly, use WASD to move around; double tap space again for ground
- While flying, hold space to fly higher, shift to fly lower
- Use trackpad to look around
- E to open inventory; click to select items and drop them outside the inventory; Esc to exit inventory and move to collect dropped items
- Two fingers up and down to switch items, or use number keys.
- Left-click to break things
- Right-click to place blocks / use selected items
- F5 to switch camera views

- **New blocks to enable in NyokaTofali:** none
- **Reflections (July 20th, 2017)**
  - I gave Form 1 and Form 2 an entire class (80 mins) to do this.
  - Interestingly, some students were confused and/or disinterested when given time to independently explore Minecraft, but most were enthusiastic. I hope the goal-oriented lessons coming up will help give them some direction.

Unit 4 – Using NoykaTofali to Display Where We Are in Minecraft

- Have students open Minecraft on one half of the screen, NyokaTofali on the other.
- Have them add the following blocks in NyokaTofali:
  - *from mcpi.minecraft import Minecraft*
  - *mc = Minecraft.create()*
  - *playerTilePos = mc.player.getTilePos()*
  - *print(playerTilePos)*
- Have them run the program, then move up/down/left/right and run it again a few times so that they can watch as the numbers change.
- Review the XYZ axis with students, ensure they understand that the 3 numbers correspond to these axes.
- If time remains, let students do as they please in Minecraft.
- **New blocks to enable in NyokaTofali:**
  - *import_minecraft | from mcpi.minecraft import Minecraft*
  - *mc_create | mc = Minecraft.create()*
  - *gettilepos | playerTilePos = mc.player.getTilePos()*
- **Reflections (July 21st, 2017)**
  - Students understood this lesson successfully with plenty of free time remaining. The concept of the Z-axis did not seem to be fully understood by them, but this is understandable and not essential. Exposure is the important thing, especially to prepare them for setting their own locations.

Unit 5 – Using NoykaTofali to Teleport to Other Blocks

- Have students open Minecraft on one half of the screen, NyokaTofali on the other.
- On the board, draw the XYZ axis again as a reminder.
- Have them add the following blocks in NyokaTofali:
  - *from mcpi.minecraft import Minecraft*
  - *mc = Minecraft.create()*
  - *mc.player.setTilePos(10, 20, -5)*
- Allow them to spend some time changing the numbers themselves and seeing where they end up.

- Then have them set their position to (0, -40, 0) and ask them what they see. Most of them should be stuck underground at this point. As a challenge, have them change the numbers in setTilePos to get back above ground.
- If time remains, let students do as they please in Minecraft.
- **New blocks to enable in NyokaTofali:**
  - *settilepos | mc.player.setTilePos(_, _, _)*
- **Reflections (July 24th, 2017)**
  - This was relatively easy for students to do and grasp. All completed this within 40 minutes. It's better to do this unit within a single 40 minute block for this reason, otherwise they will have a large amount of time left over.

Unit 6 – Change the Ground Underneath You to Tanzanite

- See page 57 of *Learn to Program with Minecraft* for inspiration for this unit.
- Have students open NyokaTofali and Minecraft.
- Walk students through entry of the following blocks in NyokaTofali:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
pos = mc.player.getTilePos()
print(pos)
```

- Have them run the program, have them recall that the three numbers are (X, Y, Z) coordinates
- Walk students through entry of the following blocks, having them stop to run after each print statement to see that we are breaking apart the coordinates:

```
x=pos.x
print(x)
y=pos.y
print(y)
z=pos.z
print(z)
```

- Now have them enter the following block, with special care given to subtracting 1 from the Y coordinate:

```
mc.setBlock(x, y - 1, z, 22)
print("Tanzanite is under you.")
```

- Have students run this and wait for them to observe for themselves that a Tanzanite block was placed underneath them. Then have them move to a different location, change 22 to a number of their choice, and see what happens.
- **New blocks to enable in NyokaTofali:**
  - *setblock | mc.setBlock(x, y, z, 0)*
- **Reflections (July 27th, 2017)**
  - Form 2 did well with this, with many students able to fill in for themselves the middle section breaking the player's position into its constituent components. A few students also seemed to begin to grasp the direction we are headed, namely building/doing complex things in Minecraft using functions like setBlock.
  - Form 1 also did well with this, although they took longer to complete than Form 1, which is understandable. One or two students took the initiative to work ahead and fill in the X/Y/Z variable blocks themselves; less than Form 2, but very promising all things considered.

Unit 7 – Creating a (Completely Solid) Igloo

- See page 57 of *Learn to Program with Minecraft* for inspiration for this unit.
- Have students open NyokaTofali and Minecraft.
- Walk students through entry of the following blocks in NyokaTofali:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
pos = mc.player.getTilePos()
x=pos.x
y=pos.y
z=pos.z
print((x, y, z))
```

- Have them run to check that everyone is together.
- Then have students add the following blocks, asking for their suggestions for the height, width, and length variables:

```
length=5
width=6
height=3
```

- Have them add the final blocks, where 79 is the block type for ice, and the final block sets them on top of the structure so that they're not stuck inside it. Diagram this out on the board before having them copy it down so that they have an understanding of it first.

```
mc.setBlocks(x, y, z, x + width, y + height, z + length, 79)
mc.player.setTilePos(x, y + height, z)
```

- As a reminder for what the final number means, have them put their own number (between 1 and 200) and see what kind of cuboid they end up with.
- **New blocks to enable in NyokaTofali:**
  - *setblocks | mc.setBlocks(x1, y1, z1, x2, y2, z2, blockType)*
- **Reflections (July 31st, 2017)**
  - This lesson worked well for both Forms, it took about 40 minutes to walk through everything and most everyone had no major issues.

Unit 8 – Creating a (Hollow) Igloo

- See page 61 of *Learn to Program with Minecraft* for inspiration for this unit.
- Have students open NyokaTofali and Minecraft.
- Walk students through entry of the following blocks in NyokaTofali:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
pos = mc.player.getTilePos()
x=pos.x
y=pos.y
z=pos.z
```

- Briefly check that everyone is together, then ask them for numbers to use in the following blocks:

```
l=__
w=__
h=__
```

- On the board, ask students about how we form a cuboid / rectangular prism from an (X, Y, Z) coordinate as review from last class. They should answer that we use X, Y, Z, then X + length, Y + height, Z + width in setBlocks; first encourage them to enter the block if they remember, then walk around with it:

```
mc.setBlocks(x, y, z, x + w, y + h, z + l, 79)
```

- Now ask if we can create prisms inside prisms. How would we create a prism of air from the floor up to the ceiling, such that we create a house with walls and a ceiling one block thick? This will be challenging, but just make them think for a minute or two, then walk around with:

```
mc.setBlocks(x + 1, y, z + 1, x + w - 1, y + h - 1, z + l - 1, 0)
```

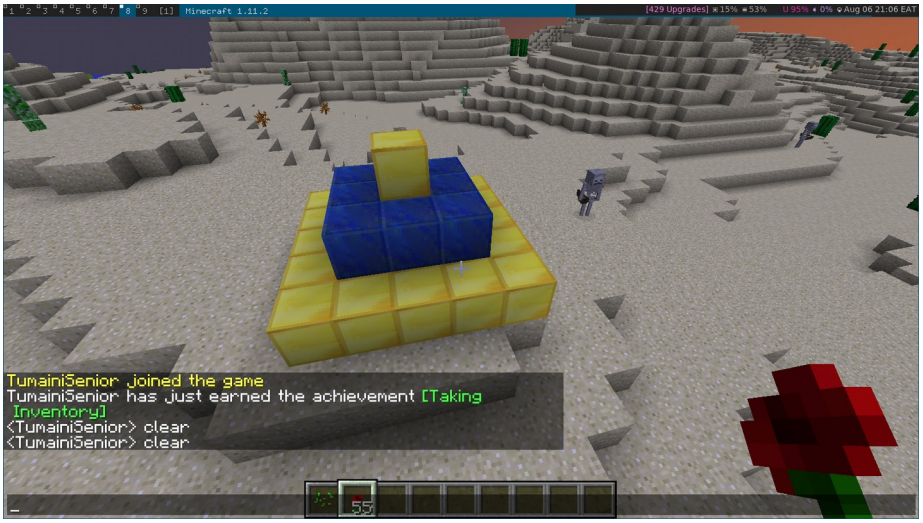
- The final block positions them inside the house, rather than getting them stuck in the corner:

```
mc.player.setTilePos(x + 2, y, z + 2)
```

- **Reflections (August 1st, 2017)**
  - Form 1 did well with this; working through some of the math beforehand is recommended, just not too much or they'll lose focus or motivation I've noticed. Same with Form 2.

Unit 9 – Build a Pyramid

- This unit is intended to be self / group driven. It should take most if not all of a week, assuming there are at least two 80 minute blocks per stream per week.
- Show them an image / screenshot / actual Minecraft rendering of a pyramid, such as this one:



- Challenge them to build one just like this on their own or in groups. Tell them that this will be assigned to them for the entire week, and that while they may choose to do other things in Minecraft during the week, the first person / group that finishes will be given a reward.
- Code is as follows, although students may use any statements they wish to complete this:

```
from mcpi.minecraft import Minecraft
```

```
mc = Minecraft.create()
pos = mc.player.getTilePos()
x=pos.x
y=pos.y
z=pos.z
mc.setBlocks(x, y, z, x + 4, y, z + 4, 41)
mc.setBlocks(x + 1, y + 1, z + 1, x + 3, y + 1, z + 3, 22)
mc.setBlocks(x + 2, y + 2, z + 2, x + 2, y + 2, z + 2, 41)
```

- It's possible that no one completes this, in which case at the end of 2 80-minute blocks, show them how it's done and have them reconstruct on their own computers.
- **Reflections (August 11th, 2017)**
  - Due to last-minute exams for Form 2 followed by a week-long vacation for them, I only did this unit with Form 1. In hindsight I'm glad I made this close enough to the spire example in the book, as students saw the similarity and used the book to at least get a spire. Modifying it to be a pyramid was the hard part for them. One stream had a few groups get close but none exact; later I had two students from this stream come to me after class to solve it, and they did get it. The second stream had one group that got it exactly within the span of only one class, which exceeded my expectations.

Unit 10 – Input Function / If Statement Introduction

- This unit introduces the input() function and if statements.
- Have students enter the following blocks, one by one (the blank line should be filled in with their own name):

```
jina=input("Ingiza jina lako: ")
if jina.startswith("__________"):
  print(jina + " amebariki.")
```

- Ask them to run this but tell them not to enter anything at the prompt. Some of them will have errors; use this opportunity to encourage them to work together to fix their problems by comparing their code with that of those around them who don't have errors.
- Once everyone has no errors, explain to them how to enter input into the prompt. Have them enter their full name so that " amebariki" follows.
- Guide the class in a discussion about what this does, how it can take any name but only says "amebariki" for the student.
- Now ask if we can do something similar for nation. On the whiteboard, write out in English something along the lines of, "Ask the person to enter their country. If their country starts with the letter 'T', print the person's name followed by " anaishi Tanzania." The code should be something along the lines of:

```
nchi=input("Ingia nchi yako: ")
if nchi.startswith("T"):
  print(jina + " anaishi Tanzania.")
```

- **New blocks to enable in NyokaTofali:**
  - *if | if(_____):*
- **Reflections (August 17th, 2017)**
  - Form 1 did well with this. For the country exercise, I gave one stream more code than I should have, so I didn't get to gauge their grasp as well as the other. But the other stream got an English description along the lines above, and many of them translated the code for

processing name successfully into code for processing country. Also, I originally did this lesson after Unit 11 (Blast a Crater Around You), but it really should be done before it.
    - Using Swahili here, and in future units, is a good idea. It gets their attention, results in a good mix of languages as the Python standard library is in English, and shows them that these concepts transcend linguistic boundaries.
    - Only did this with Form 1; skipped for Form 2 as they were out on vacation, and they've had exposure to these concepts already.

Unit 11 – Blast a Crater Around You

- This unit uses the **input()** function and **if** blocks in the context of Minecraft.
- Have students enter the following blocks:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
answer=input("Make a crater?")
print(answer)
```

- Have them run at this point and practice entering things into the terminal. They should see that whatever they enter gets printed back to them.
- Then have them add an if statement:

```
if answer == "Y":
```

- And finally add the following to the inside of the if statement block:

```
pos = mc.player.getTilePos()
x=pos.x
y=pos.y
z=pos.z
mc.setBlocks(x + 1, y + 1, z + 1, x - 1, y - 1, z - 1, 0)
mc.postToChat("Boom!")
```

- Without explaining anything to them beforehand, ask them to run the program and see what it does / how it works. Give them some time to figure out how to enter input into terminal, try different submissions to input(), etc.
- Bring the class together to discuss what input() and how if statements work.
- **Reflections (August 15th, 2017)**
    - Stream 1A did well with this, taking about half of an 80 minute block. A few groups did have technical problems with NyokaTofali due to an unhandled exception in the back-end. I rewrote the back-end to be simpler and handle the exception so that this won't happen again.
    - Skipped for Stream 1B due to recent timetable interference.
    - Ended up doing this for only one Form 2 stream. I think it's worth doing this unit but isn't as worthwhile as I had hoped and could be replaced with something else in the future.

Unit 12 - If-Else Statement Introduction

- Only NyokaTofali for this unit, no Minecraft involved
- At each table, have each student give themselves a unique number, starting from 1. Write these down on the whiteboard.

- Now have students add the following blocks, replacing the blank with the first name of student #1:

```
namba=int(input("Ingiza namba ya mwanafunzi: "))
if namba == 1:
  print("__________ ana namba %d" % namba)
```

- Have everyone run this twice: the first time enter 1, the second time entering any other number. Talk about what happens in both cases.
- Now add an empty else block to the end of the program:

```
else:
```

- And ask students what we could put in this block to also print out students other than just #1. If there are no immediate answers, have them talk about it themselves and work on it without instructor help. Give them a hint if they get stuck for too long: i.e., can we put another if statement inside the else, this one for student #2?
- Have them add blocks for all students in their group, and raise their hands when they've finished. When their program works for all students in their group, their group can do whatever they want for the rest of class.
- **New blocks to enable in NyokaTofali:**
  - *else* | *else:*
- **Reflections (August 19th, 2017)**
  - Only did this with Form 1; skipped for Form 2 as they have already had exposure to if-else last year, plus I did not see them for several class periods recently.
  - This went well for Form 1, many students were able to understand what was needed and did it without hesitation. Highly recommend doing this.

Unit 13 – Jenga Taa Yenye Urefu Fulani

- Introduces the **elif** statement.
- This unit is for students to work through mostly independently.
- First present the following blocks:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
pos = mc.player.getTilePos()
jengaTaa(pos, 1)
```

- Have them run this and verify that they can get a working lamp.
- Then have them outline the following on the board:
  - We want to ask the user for a number: *jibu = input("Tujenge taa A, B, au C: ")*
  - If *jibu* is "A", then use the *jengaTaa* block to build a lamp of *urefu* 1
  - Else if *jibu* is "B", then use the *jengaTaa* block to build a lamp of *urefu* 2
  - Else if *jibu* is "C", then use the *jengaTaa* block to build a lamp of *urefu* 3
  - Else they should print the message: *"Uingize A, B, au C."*
- Students can work by themselves or with others. Tell them that the only rule is that they can use ONLY ONE **if** statement and ONLY ONE **else** statement. They can use as many **elif** statements as they wish. When they have finished, they should raise their hand for instructor to review.

- **IMPORTANT:** When reviewing each student's work, flip the switch of each lamp to the "on" position. This will expose them to how levers work and the basics of redstone / electricity in Minecraft, which may inspire them to look into it more on their own.
- Example solution is:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
pos = mc.player.getTilePos()
jibu = input("Tujenge taa A, B, au C: "))
if jibu == "A":
  jengaTaa(pos, 1)
elif jibu == "B":
  jengaTaa(pos, 2)
elif jibu == "C":
  jengaTaa(pos, 3)
else:
  print("Uingize A, B, au C.")
```

- **New blocks to enable in NyokaTofali:**
  - *elif | elif _____:*
- **Reflections (August 27th, 2017)**
  - For Form 2, this did not go as well as I had hoped. I'm not sure whether it was just an off day, or whether it was genuinely difficult, or what. I did have one students in each stream complete it, with minimal assistance from me, which was positive. But most students struggled severely with this, if they were motivated to complete it at all.
  - Form 1 seemed to do better at this than Form 2, not exactly sure why.
  - Some students were visibly excited about seeing the electricity aspect of this, but I'm not sure many realized the broader implications of it, or at least the long-term interesting side of it.

Unit 14 : Coin Flip Game

- Introduces the **import, while,** and += statements.
- Walk students through the following code, block by block:

```
import random
jumla=0
while jumla < 3:
  _=input("Tayari?")
  kichwa_au_mkia=random.randint(0, 1)
  if kichwa_au_mkia == 1:
    jumla += 1
    print("Kichwa. Umeongeza pointi.")
  else:
    print("Mkia. Jaribu tena.")
print("Umeshinda.")
```

- When complete, have students run the program and play until they have won, at which point they should raise their hand for instructor review.
- Once they've won, they can do as they wish for the rest of class.
- **New blocks to enable in NyokaTofali:**
  - *import | import _____:*
  - *while | while _____*
  - *varplus | ___ += _____*

- **Reflections (August 28th, 2017)**
  - Both Forms benefited from this, highly recommend doing it.

Unit 15 : More While Loops / Trail of Flowers in Minecraft

- Involves two more programs using **while** loops.
- The first program doesn't involve Minecraft; walk students through adding these blocks:

```
import time
jumla_sekunde=0
while True:
  print("Sekunde zilizopita: %d" % jumla_sekunde)
  jumla_sekunde += 1
  time.sleep(1)
```

- After pulling the class together to briefly ask what this does and how (be sure to discuss meaning of "while True"), have students open Minecraft and walk them through this program:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
import time
while True:
  pos = mc.player.getTilePos()
  mc.setBlock(pos.x, pos.y, pos.z, 38)
  print("Maua yamepandwa.")
  time.sleep(0.2)
```

- **New blocks to enable in NyokaTofali:**
  - *sleep | time.sleep(sekunde)*
- **Reflections (August 31st, 2017)**
  - Form 2 has already had exposure to infinite loops, so no problems here. Good reinforcement for them.
  - Served as a good introduction to infinite loops for Form 1, no problems with them.

Unit 16 – Brute Force Searching for Other Group's Passwords

- **Before they turn on their computers,** have each table write down a name for their group, along with a **THREE digit** password (any longer will not be doable within the span of a class). Have them keep this secret in their group, fold the paper up, and turn it in.
- In the *nyokatofali.js* file, modify the *jaribuNambaYaSiri* function definition to map each group name to their respective password.
- **Now let them turn on their computers. If they do so before this point, they will not get the updated nyokatofali.js file.**
- On the whiteboard, write down each group name, then next to each, write down the name of the group whose password they will try to break.
- Take them through the following code, having them replace the blank "_____" with the name of the group whose password they are trying to break:

```
for a in range(0,10):
  namba_ya_siri="%d" % a
  print("Tunajaribu: %s" % namba_ya_siri)
  if jaribuNambaYaSiri("_____________", namba_ya_siri):
    print("Tumeiona: %s" % namba_ya_siri)
    import sys
    sys.exit()
```

```
print("Tumeshindwa kutafuta namba ya siri.")
```

- Have them run this and observe what happens.
- Obviously this will fail to find any passwords, as it only searches the space of one-digit possibilities. Ask for any suggestions to how we can change this to search for all two-digit passwords, then have them change their program so it starts as:

```
for a in range(0,10):
  for b in range(0,10):
    namba_ya_siri="%d%d" % (a, b)
```

- Have them run this and observe that it will also fail to find any passwords. Ask them to make the necessary changes to search for three-digit passwords, initially giving them as little assistance as possible. Ideally one group will manage to make their program start like this:

```
for a in range(0,10):
  for b in range(0,10):
    for c in range(0,10):
      namba_ya_siri="%d%d%d" % (a, b, c)
```

- **New blocks to enable in NyokaTofali:**
  - *for | for ____ in _____:*
  - *exit | sys.exit()*
- **Reflections (August 31st, 2017)**
  - Form 2A did well with this, with everyone finding each other's passwords within the span of a single class, despite the fact that I messed up and accidentally asked them for 4-digit passwords instead of three. They were able to independently add the third and fourth loops to accomplish this, with modifications to the namba_ya_siri variable as well, which was exciting to see.
  - Form 2B also did well with this, one group even jumped ahead and simply modified the first for loop to cover all numbers from 0 to 999. For some students this involved too much typing and they fell behind, but I'd say for the majority it was worthwhile.

Unit 17 – Comparison of Various For Loop Constructions

- Have students add the following blocks in NyokaTofali:

```
for kitu in [1,2,3]:
  print(kitu)
```

- Have them run and verify their output.
- Then have them append the following blocks:

```
for kitu in (4,5):
  print(kitu)
```

- Have them run and verify their output.
- Then have them append the following blocks:

```
for kitu in "tumaini":
  print(kitu)
```

- Have them run and verify their output.

- Then have them append the following blocks:

```
for kitu in {'Makuyuni' : 'TSSS', 'Karatu' : 'TJS'}:
  print(kitu)
```

- Bring the class together to discuss the similarities between the loops, and their differences. Keep this conversation high-level; the important point to get across is the fact that each of these "things" / "types of things" can be broken down into smaller things.
- **Reflections (September 17th, 2017)**
  - As I expected, most Form 2A students found this to be boringly easy. I still think it was worth it to remind them that for loops can be used for various types, and thus remind them of the general concept backing a for loop.
  - Form 2B, Forms 1A and B: _____

Unit 18 – Function Basics

- This unit introduces/reinforces the concept of defining functions.
- Have students enter the following blocks:

```
neno=input("Ingiza neno la Kiswahili: ")
print("Kwa Kiingereza: %s" % kamusi(neno))
```

- Then have them run and ask them what happens / what the problem is. Focus should be on the error "kamusi is not defined." So now we will define the kamusi function:

```
def kamusi(neno):
  if neno == "mti":
    return "tree"
  else:
    return "Mimi sijui."
```

- Have them run this and ask them to first enter "mti," then run again and enter another word. Walk them through adding another word to the dictionary:

```
  elif neno == "rangi":
    return "color"
```

- and run again, this time entering "rangi" at the prompt.
- **New blocks to enable in NyokaTofali:**
  - *def* | *def* ____ (_____)
  - *return* | *return* _____

Unit 19 – Recursion

- This unit introduces the concept of recursion. The goal is not to harp on the details, but merely to expose them to the fact that functions can be called from within themselves, and also to archetypal structure this program has / demonstrates.
- Have students enter the following blocks:

```
dna=list(input("Ingiza DNA: "))
print(dna)
print("Adenine: %d" % adenine_ngapi(dna))
```

- Have them run after the first print statement, then after the second. Ask them what the problem is when running after the second (we need to define a function called "adenine_ngapi"):

```
def adenine_ngapi(dna):
  if dna == []:
    return 0
```

- Have them run this and just press Enter at the prompt; they should all get 0 (ask them: does an empty string of DNA have any adenine bases in it?)
- Then have them add this to the conditional inside "adenine_ngapi:"

```
elif dna[0] == "A":
    return 1 + adenine_ngapi(dna[1:])
```

- Have them run this and just enter a single "A" at the prompt; they should all get 1.
- Finally have them add this final case to the conditional:

```
else:
    return 0 + adenine_ngapi(dna[1:])
```

- Have them run this and enter any strand of DNA that they choose, with at least a few As inside. Go around and verify that their program works for the DNA they gave it.

Unit ___ - Binary / Simple Search Algorithm Using Recursion

Unit ____ - Factorials Using Recursion

To Review: minecraft101.net for lesson ideas.
Idea: Redstone Powered Rail with Lever / Some kind of start/stop mechanism

Unit ___ – Path of Diamonds (using while loop)

Unit ___ – Grow a Forest (using function defns and calls)

Unit ___ – Build a Staircase (using for loop)

Unit ___ – Build a Pyramid (using for loop, range)

At this point, should be about 6 weeks into term (approx. beginning of September). For rest of term, have people get in groups or by themselves to work on a Minecraft/Python project of their choice. Form 1 uses NyokaTofali exclusively, Form 2 can use NyokaTofali or Thonny. Provide them a list of suggestions from the book and allow them to suggest their own; remember that they have access to the book, so can do similar things but require a twist for originality. For final exhibitions, set up a few pairs of computers networked together, then have them run their programs in those shared worlds.

IDEA: May need to add ability to have multiple Spigot games per laptop, one for each student, to prevent interference among students, inadvertent destruction, etc.