TRUSTYOU

Intro to Python for Non-Programmers

Ahmad Alhour / Team Lead, TrustYou

23 April, 2019

Overview

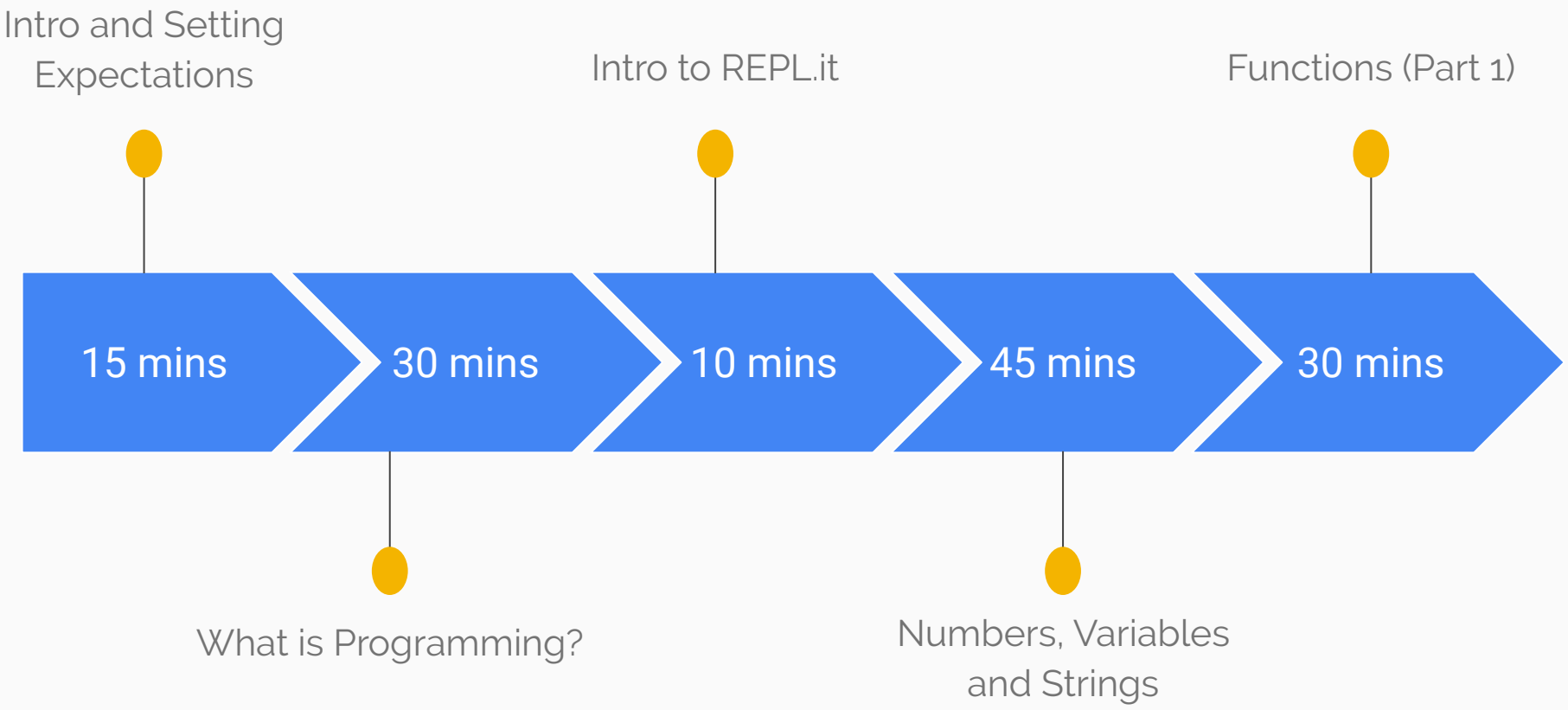
- A very basic intro to programming
- Intended for people who have no clue about it
- Mainly, to give you a taste of what it is about
- Hopefully, it'll put you on the right track to study on your own

Hello!

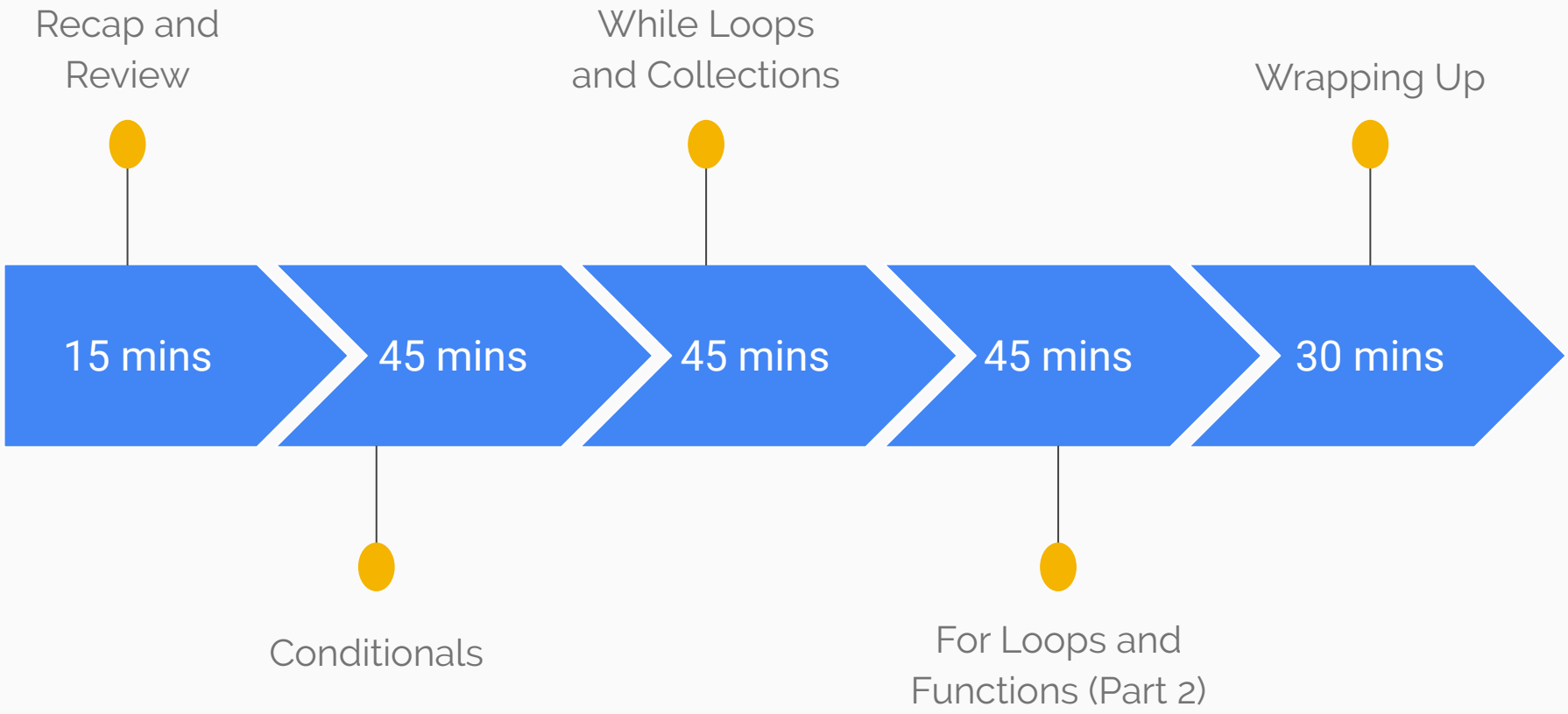
I am Ahmad

I have been doing programming as a hobby since I was a kid, and professionally since 2008. I also have a B.Sc. degree in Computer Science. At TrustYou, I lead several projects, all powered by Python!

Plan Until Lunch



Plan After Lunch



1.

What Are Your Expectations?

“Blessed is he who expects nothing, for he shall never be disappointed.”
— Alexander Pope

Group Exercise:
Let's Set Expectations Together

- Learn the basics of Programming
- Learn the basics of Python
- There will be no installation

2.

What is Programming?


WHAT IF I TOLD YOU
THAT EVERYTHING IS A PROGRAM?
imgflip.com

Programming 101

- Is a way of telling the computer how to do certain things by giving it a set of instructions
- These instructions are called Programs
- Everything that a computer does, is done using a computer program

Example Programs

3.
What is a Programmer?

What is a Programmer?

"A programmer is an Earthling who can turn big amounts of caffeine and pizza into code!"

— Anonymous

Diagram belonging to Note D

Number of Operations	Nature of Operations	Variables for Data						Working Variables								
		$^1V_0$	$^1V_1$	$^1V_2$	$^1V_3$	$^1V_4$	$^1V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	
		+	+	+	+	+	+	+	+	+	+	+	+	+	+	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		m	n	d	m'	n'	d'									$\frac{d'm-dm'}{mn'-m'n} = y$
1	×	m	.....	.....	.....	n'	.....	mn'								
2	×	.....	n	.....	m'	.....	.....	.....	$m'n$							
3	×	.....	.....	d	.....	.....	.....	.....	.....	dn'						
4	×	.....	0	.....	.....	.....	d'	.....	.....	.....	$d'n$					
5	×	0	.....	.....	.....	.....	0	.....	.....	.....	.....	$d'm$				
6	×	.....	.....	0	0	.....	.....	.....	.....	.....	.....	.....	dm'			
7	−	.....	.....	.....	.....	.....	.....	0	0	.....	.....	.....	.....	$(mn' -$		
8	−	.....	.....	.....	.....	.....	.....	.....	.....	0	0	.....	.....	...		
9	−	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	0	0	...		
10	+	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	$(mn' -$		
11	+	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....			

Seriously, what is a programmer?

- A person who writes instructions is a computer programmer
- These instructions come in different languages
- These languages are called computer languages

4.
What is a Programming Language?

What is a Programming Language?

- A programming language is a type of written language that tells computers what to do
- They are used to make all the computer programs
- A P.L. is like a set of instructions that the computer follows to do something

Machine Language

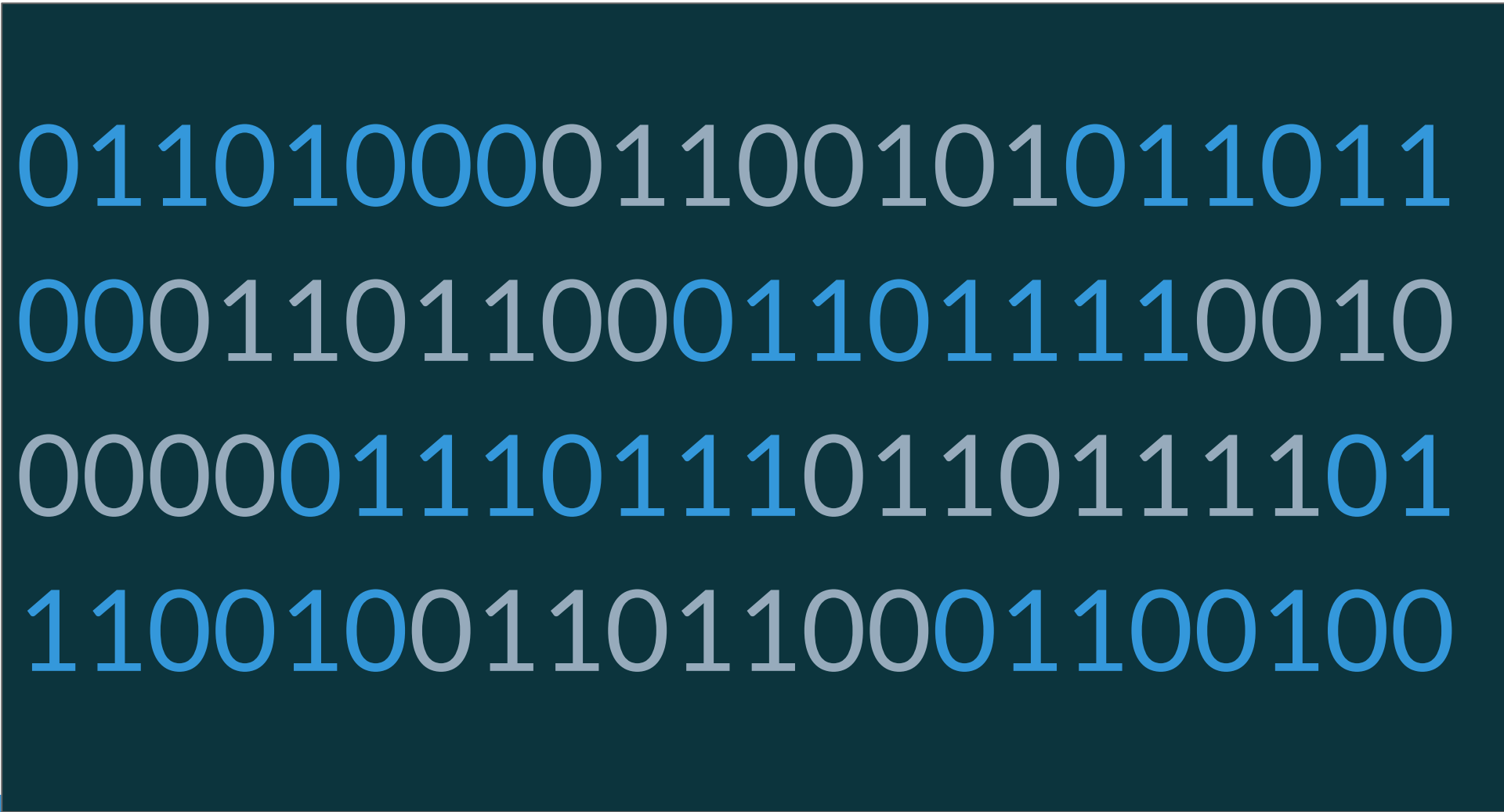
- The computer's native language
- The set of instructions have to be coded in 1s and 0s in order for the computer to follow them
- Writing programs in binary (1s and 0s) is tedious
- Lowest level language

High-level Languages

- Programming languages that are closer to English than Machine code
- Human-readable
- Expresses more by writing less

5.
Example Programs

Displaying “Hello, World!” in Machine Code



Displaying “Hello, World!” in Assembly

```
  global _main
  extern _GetStdHandle@4
  extern _WriteFile@20
  extern _ExitProcess@4

  section .text
_main:
  ; DWORD bytes;
  mov   ebp, esp
  sub   esp, 4

  ; hStdOut = GetstdHandle( STD_OUTPUT_HANDLE)
  push  -11
  call  _GetStdHandle@4
  mov   ebx, eax
```

```
  ; WriteFile( hstdOut, message, length(message), &bytes,
0);
  push  0
  lea   eax, [ebp-4]
  push  eax
  push  (message_end - message)
  push  message
  push  ebx
  call  _WriteFile@20

  ; ExitProcess(0)
  push  0
  call  _ExitProcess@4

  ; never here
  hlt
message:
  db    'Hello, World', 10
message_end:
```

Displaying “Hello, World!” in C

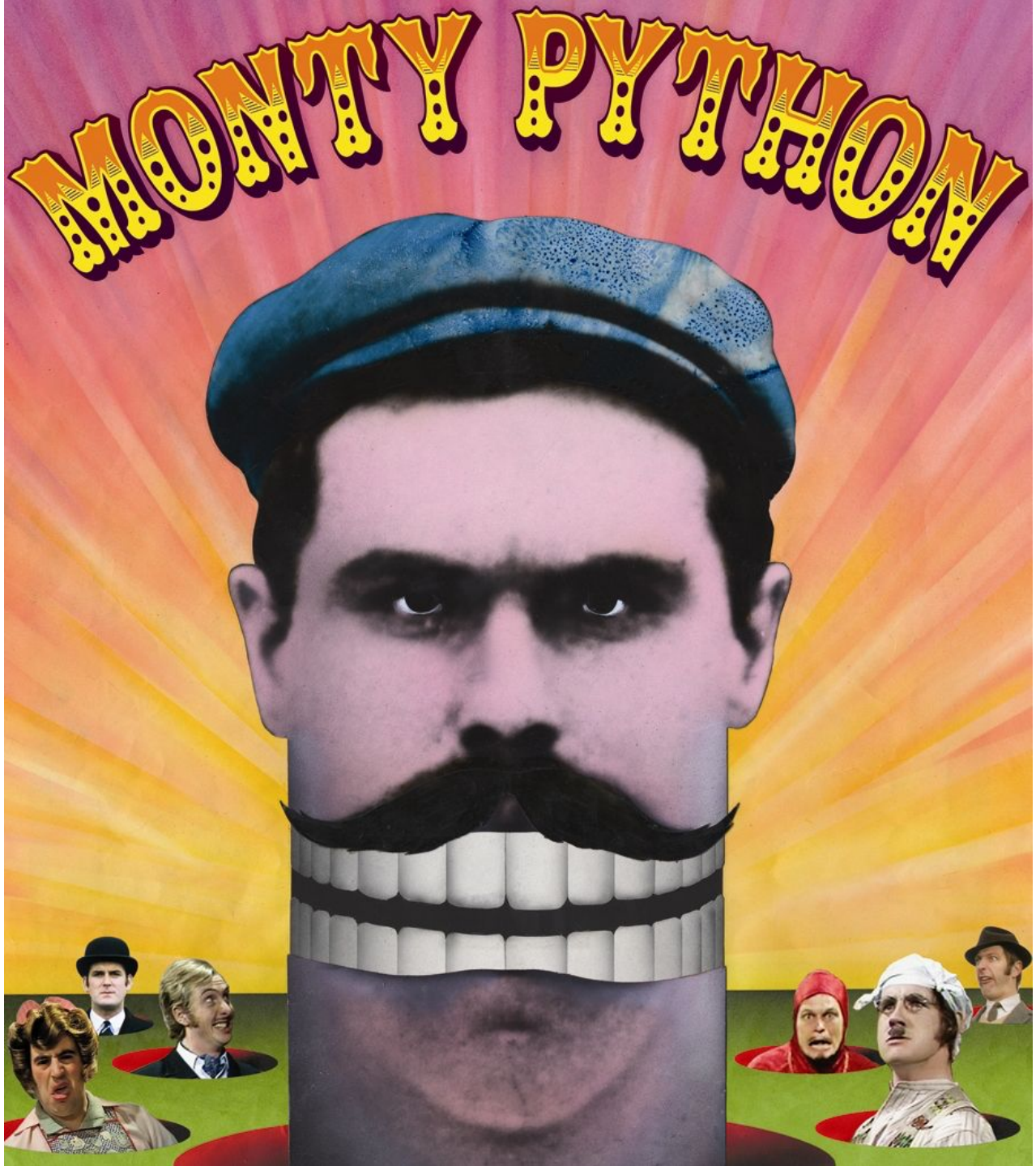
```c
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

Displaying “Hello, World!” in Python

```
print(“Hello, World!”)
```

6.
Enter: Python!


MONTY PYTHON

The Python Programming Language

- High-level programming language
- Named after BBC's "Monty Python's Flying Circus" show
- One of the most adopted languages world-wide
- Most TY products are powered by it

Example Python Program: Counting to 10

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for each_number in numbers:
    print(each_number)
```

Let’s take a break!

7.
Let's REPL.IT

Demo Time!

- General walk through REPL.IT

8.

Hello, World!

Your First Python Program

Numbers in Python

- Create a new repl on REPL.IT
- Type in the main.py tab:

```
print("hello, world!")
```

- Click on **run >**!

9.
Numbers
The Art of Calculating Stuff

Numbers in Python

- One of value types
- Demo:
  - Numbers in interactive mode
  - Basic arithmetic

Evaluation Order of Operations

- First, "*" and "/" are evaluated
- Then, "-" and "+" are evaluated
- All operations are evaluated left-to-right

Guess the result of the following:

- 9 * 9 - 9 * 10 / 15
- 1 + 1 + 1 * 2 + 6 / 2
- 10 + 1 * 2 - 4 / 20
- 50 - 5 - 5 - 10 / 2
- 1 + 2 - 5 * 3 - 7 + 4 * 9 / 12

Grouping Calculations with Parentheses

- Used for separating calculations into distinct groups
- For example:
  - (10 * 2) + (5 - 3) / (4 + 2)
  - → 20 + 2 / 6
  - → 20 + 0.333
  - → 20.333

Evaluation Order of Operations: Revisited!

- First, what is in “(” and “)” is evaluated
- Then, “*” and “/” are evaluated
- Then, “-” and “+” are evaluated
- All operations are evaluated left-to-right

Guess the result of the following:

- 9 * (9 - 9) * 10 / 15
- (1 + 1 + 1) * (2 + 6) / 2
- 10 + 1 * (2 - 4) / 20
- 50 - 5 - (5 - 10) / 2
- (1 + 2) - 5 * (3 - 7 + 4) * 9 / 12

10.

Variables

The Art of Naming Stuff

Variables in Python

- Used to give names to values
- You give names to stuff to refer to them later on in your program
- You give names to stuff using the “=” symbol, known as “assignment operator”
- Variable names must begin with a letter

Demo Time

- Implement one plus one operations using variables

Variables in Python

Example usage:

- Calculate the average population for Earth in 2015
- Give it a name
- Everytime you want to refer to it you just use the name rather than to recalculate it again

Demo Time

Break the calculation down using variables:

- (10 * 2) + (5 - 3) / (4 + 2)
- X + Y / Z
- Store result in “result” name

Exercise

Break the following calculations down into sub-calculations and store them into variables:

- 9 * (9 - 9) * 10 / 15
- (1 + 1 + 1) * (2 + 6) / 2
- 10 + 1 * (2 - 4) / 20
- 50 - 5 - (5 - 10) / 2
- (1 + 2) - 5 * (3 - 7 + 4) * 9 / 12

11.

Strings

The Art of Texting Stuff

Strings in Python

- Used to represent text in Python
- Examples:
  - “Hello, World!”
  - “My name is Ahmad.”
  - “Bier und Brezel.”
- Any text in between “” is a string in Python

Demo Time

- Type in the following strings in the interactive shell:
  - “Hello, World!”
  - “My name is Ahmad.”
  - “Bier und Brezel.”
- Use all of the above with print
- Store all of the above in variables and then print them

Exercise

Print the following poem:

Doubt thou the stars are fire;
Doubt that the sun doth move;
Doubt truth to be a liar;
But never doubt I love.
— William Shakespeare, Hamlet

Let’s take a break!

12.

Functions

The Art of Naming Computations

Functions in Python

- Just like how variables give name to values, functions give name to calculations/instructions
- It is easier to refer to a group of instructions with a name than write them all over again
- You need three main words to define functions: “def”, “<name>” and “return”

Functions in Python

- You know functions already!
- Print is a Python function that takes whatever value you'd like and prints it on the screen
- Print can take several values separated by commas, e.g.:
  - print("My name is:", "Slim Shady")
  - print("One plus one is:", 2)

Demo Time

- Define a new function
- Print William Shakespeare's poem
- repl.it/@aalhour/WilliamShakespeare

Demo Time

- Define a new function
- Code the one plus one example in variables
- Return the result

Exercise

- Define a function to calculate each of the following, with variables, and return the result:
  - 9 * (9 - 9) * 10 / 15
  - (1 + 1 + 1) * (2 + 6) / 2
  - 10 + 1 * (2 - 4) / 20
  - 50 - 5 - (5 - 10) / 2
  - (1 + 2) - 5 * (3 - 7 + 4) * 9 / 12

Functions in Python

- Functions can take values as arguments!
- Useful for doing tasks with user input, for example:
  - Given 5 numbers, return the avg.
  - Given weight and height of a person, calculate their BMI.

Demo Time

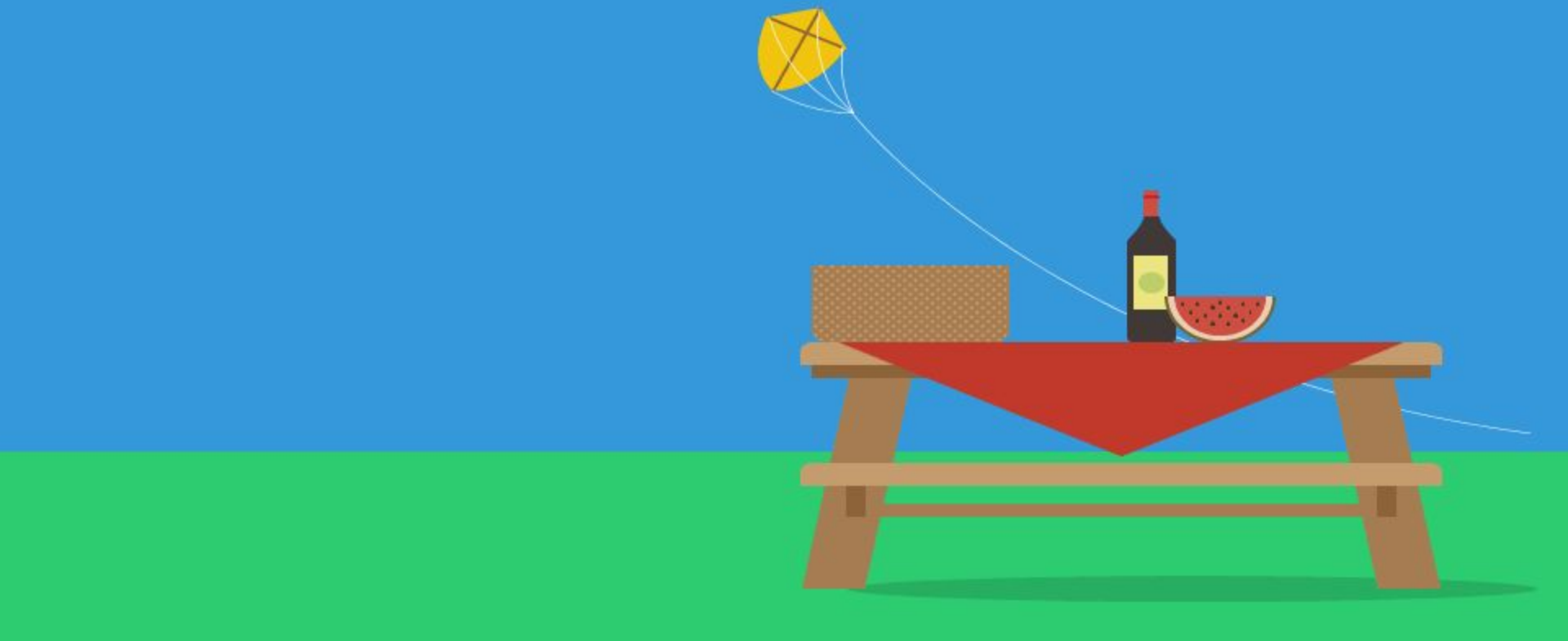
- Define a new function for printing different values
- Define a new function for calculating BMI given height and weight
- repl.it/@aalhour/FuncsWithVars

Exercise

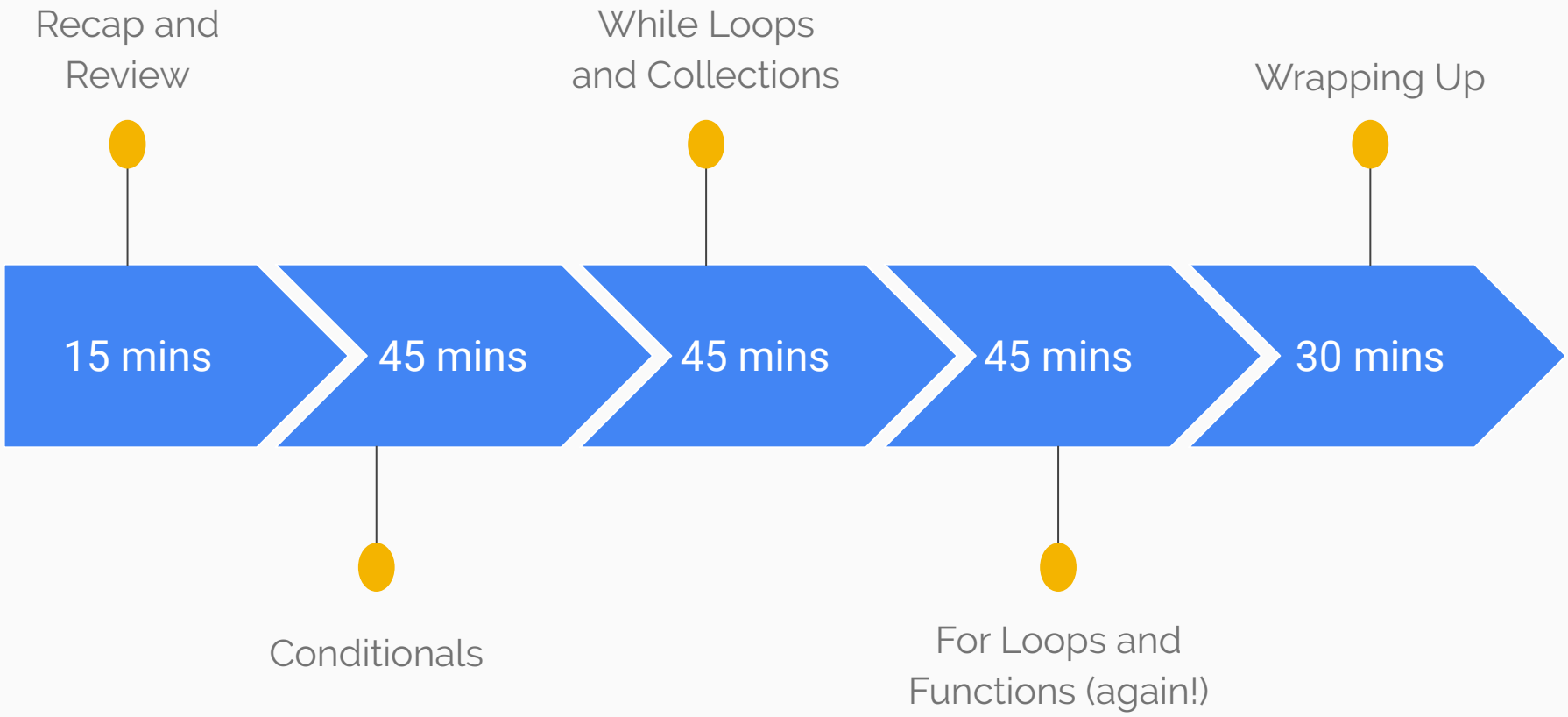
- Define a function that accepts two arguments and returns the result of their addition
- Define a function that takes five arguments and returns the average

Lunch Break

See you in an hour!



Plan After Lunch



13.

Recap

What Have We Learned So Far?

Recap

- Numbers: 1, 2, 3
- Order of operations: (,), *, /, +, -
- Strings: “Hi!”
- Variables: x = 1
- Functions:

```
def greet():
    print(“Hello, world!”)
```

14.
Conditionals
The Art of Comparing Stuff

Conditionals

- They are the Yes/No questions of programming
- The answer is either a **True** or a **False**
- These values are called **Booleans**
- Conditionals compare values together
- Using simple operators such as:
  - >, <, >=, <= and ==

Demo Time

Compare several numbers together and check the results:

- 1 < 10
- 1 <= 10
- 5 > 2
- 5 >= 2
- 1 == 1
- 0 <= 0

Exercise

Evaluate the following in your mind and then on REPL.IT:

- 1 + 1 + 1 + 1 + 1 **>** 4
- 2 * 2 **<** 5 / 3
- 5 * 3 **>=** 15
- 10 / 2 **==** 5
- (25 * 3) - 100 / 2 **<** (25 * 2) + 1

Evaluation Order of Operations: Expanded

- First, what is in “(” and “)” is evaluated
- Then, “*” and “/” are evaluated
- Then, “-” and “+” are evaluated
- Lastly, comparisons are evaluated
  - “>”, “>=”, “<”, “<=”, “==”
- Operations get evaluated left-to-right

The If Statement

- A Python statement for making decisions based on a condition
- In a nutshell: if this, then that
- For example:
  If your age is greater than 18,
  then you are allowed to enter the bar

Demo Time

- If old enough to enter bar, print “proceed to bar!”.
- If not old enough to enter bar, print “too young still!”

Exercises

Teenager Exercise 01:

Define an “age” variable and insert your age. Write an if statement to display “teenager” if age is less than 21

Exercises

Positive Number 01:

Define a “number” variable and insert a random value. Write an if statement to display “positive” if the number is greater than 0

The If-Else Statement

- Expanded If-Statement for handling cases that don't match the condition.
- In a nutshell: *if this, then that; otherwise, something else*
- For example:
  - If your age is greater than 18, then you are allowed to enter the bar; otherwise, you are asked to leave

Demo Time

Translate the following into Python:

If old enough to enter bar
then print “proceed to bar!”
else print “please, leave!”

Exercises

- Teenager Exercise 02:
  Expand the code and add an else block which prints “adult” if the teenager check is not met

- Positive Number 02:
  Expand the code and add an else block which prints “negative” if the positive check is not met

Exercises

- Teenager Exercise 03:
  Wrap the exercise's code in a function that accepts an "age" argument and returns nothing.

- Positive Number 03:
  Wrap the exercise's code in a function that accepts a "number" argument and returns nothing.

Let’s take a break!

15.
While Loops
The Art of Repeating Stuff

While Loops

- Loops are good for repeating instructions for a number of times
- They're good because we don't have to duplicate code, but we just tell the program to keep repeating stuff
- Loops repeat stuff according to a condition as well

While Loops

- Example:
  - While my BMI is above 100, keep working out!
  - While my coffee is not empty, keep writing code!
- You need two words to loop stuff:
  - **“while”** and **“break”**
  - Break is used to stop looping or else you will loop forever!

Code Examples

```
while True:
  print(1)      # Prints 1 forever

while 1 == 1:
  print(1)      # Prints 1 forever

while 5 > 2:
  print(1)      # Prints 1 forever
```

Code Examples

```
# Prints 1 once, then stops
while True:
  print(1)
  break

# Prints 1 once, then stops
while 1 == 1:
  print(1)
  break
```

Play Time

- Print “Hello, World!” forever!
- Print your own name forever!
- Calculate the following formula and print the result forever:
  - 10 * 312 / 57 + 23 * 5 + 823 / 74

Counters and Limits

- To avoid running forever, we need to maintain a counter and a limit
- Counters are variables that keep track of how many times a loop ran
- Limits allow us to break out of the loop once the loop runs for a max. number of times

Code Example

```
counter = 1

while counter < 10:
  print(counter)
  counter = counter + 1
```

Demo Time

- Add a counter number and increment it inside the loop. Print the counter to show that its value changes.
- Add a limit check before the work with a break.

Exercises

- Print numbers from 0 until 1 million
- Teenager Exercise 04:
  Define age variable and assign it to 1.
  Run the loop with the teenager check.
  The loop should print the age and then increment it until the age is no longer teeanger, after which the loop should terminate

User Input

- You can ask for the user's input using the built-in "input" function
- For example:

```
input("Enter a number:")
```

- Useful for asking the user for data.

Demo Time

- Ask the user for their name and then print it out.
- Ask the user for their age and print it out with the statement: “Your age is:”

Exercises

Teenager Exercise 05:

Ask the user for their age. Run the loop with the teenager check on the user input. The loop should print “You are still a teenager” alongside the age and then increment it until the age is no longer teeanger, after which the loop should terminate.

Exercises

Teenager Exercise 06:

Wrap the Teenager Exercise #05 with a function that asks the user for their age and then prints out the message: “You are still a teenager:” alongside their age. Increments the age number until the loop no longer runs.

Let’s take a break!

16.

Collections

The Art of Grouping Stuff

Collections

- Collections are a way of keeping items together, e.g.: bags!
- We will take a look at lists in Python
- Lists are good for memorizing the order in which we keep things too

Examples of Collections

A list of numbers:

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

A list of strings:

```
["Alice", "Bob", "Claire"]
```

A mixed list of things:

```
[123, "Ahmad", message]
```

How can I create a collection?

Anything in between “[” and “]” is considered a list

A list can contain anything in Python:

- Numbers
- Strings
- Variables
- Other lists

Exercises

- Create a list of the English alphabet
- Create a list that contains your:
  - first name
  - last name
  - your lucky number
  - your current address
- Print the above lists using the **print()** function

Ranges

- Ranges in Python are useful functions for generating collections of numbers
- More like a shortcut
- Easy to use
- Example:
  - **range(1, 10):** generates a collection of all numbers from 1 until 9 (stops at 10 but doesn't include it)

Ranges: In Detail

- range() takes three arguments:
  - Start: number to start with
  - End: number to stop at
  - Step: increment size
- Examples:
  - `range(0, 5, 1)` → `[0,1,2,3,4,5]`
  - `range(0, 6, 2)` → `[0, 2, 4]`
  - `range(0, 6, 3)` → `[0, 3]`

Ranges: Default Behavior

- **start** = **0** by default, unless specified
- **step** = **1** by default, unless specified
- **end** is mandatory and must be always specified
- Examples:
  - `range(6)` → `[0,1,2,3,4,5]`
    - Same as: `range(0, 6, 1)`

Exercises

- Generate a list of all odd numbers greater than 0 and less than 1000
- Generate a list of all even numbers greater than 29 and less than 250
- Generate a list of all numbers less than -1 and greater than -132
- Generate a list of all even numbers less than -250 and greater than -277

Let’s take a break!

17.

For Loops

The Art of Iterating Over Collections

For Loops

- For loops are another way to apply the same computation more than once according to a check in your program
- Similar to While Loops but easier to maintain
  - No need for a counter
  - No need for a beaking check

For Loops

For loops need a collection in order to… well, loop! **:P**

Example: for every item in shopping cart, print it out on the screen:

```
for item in [“Shoes”, “Tablet”]:
    print(item)
```

For Loops: Examples

```
for number in [1, 2, 3, 4, 5]:
    print(number)

for number in range(6):
    print(number)

for person in [“AA”, “AM”, “DW”, “DV”]:
    print(person)
```

How do I make a for loop?

- Syntax:
  - for <variable name> in <collection>:
    - List of commands
- The variable name can be whatever you want
- The collection can have whatever you want but it must be a **collection**, not anything else

How do for loops work?

1. For every item in the collection:
    a. Assign the item to the variable name that the programmer wrote
    b. Enter the block
    c. Execute all commands
    d. Go to 1 and grab the next the item
2. If the collection is empty or has no more items to see, then exit the block

Demo

```
for number in [1, 2, 3, 4, 5]:
    print(number)
```

Loop execution log:

number = 1 → print(1)

number = 2 → print(2)

number = 3 → print(3)

number = 4 → print(4)

number = 5 → print(5)

Play time!

- Using range() and a for loop:
  - Print your name 25 times
  - Print “Hello, World!” 10 times

Exercises

Teenager Exercise 01: Reloaded

Ask the user for their age. Check if the user is a teenager. Run a for loop over a range of ages until an age value that is not a teenage number anymore. Run a for loop and display the “you’re a teenager” message according to the list of ages (range)

Exercises

Teenager Exercise 02: Reloaded

Wrap your solution for the "*Teenager Exercise 01: Reloaded*" exercise with a function that asks the user for their age and then prints out the message:
"You are still a teenager:" alongside their age.

Let’s take a break!

18.

Functions: Reloaded

The Art of Naming Computations

Functions: Reloaded

Let's solve the FizzBuzz challenge!

17.
Wrapping Up

What You've Learned So Far

- Numbers: `1, 2, 3`
- Order of operations: (,), *, /, +, -, <, >, >=, <=, ==
- Strings: `"Hi!"`
- Variables: `x = 1`
- Functions:

```
def greet(person):
    print("Hello, ", person)
```

- Conditionals: `1 >= 10`

What You've Learned So Far

- While Loops:

```
while True:
    greet("Ahmad")
```

- Lists:

```
[1, 2, 3, "Alice", "Bob"]
```

- Ranges:

```
range(10) → [1, 2, 3, …, 9]
```

- For Loops:

```
for number in range(10):
    print(number)
```

18.
Resources for Further Study

Resources for Further Study

Online Courses

- Intro to Python - by Udacity
  - https://bit.ly/2N45h6F
  - 5 weeks long. Completely FREE.
- Intro to Python: Absolute Beginner - by edX
  - https://bit.ly/2NJg2MW
  - 5 weeks. Completely FREE.
- Python for Everybody - by Coursera
  - https://bit.ly/1KSzsbb
  - 7 weeks long. Only first seven days are free.

Resources for Further Study

Interactive Learning

- Codecademy - Learn by Coding
  - https://www.codecademy.com
  - Completely Free.

Books

- Think Python, 2nd edition
  - http://greenteapress.com/thinkpython2
  - Completely Free Online.

Thanks!

Any questions?

Find me at:
aalhour.com
github.com/aalhour
twitter.com/ahmad_alhour