## Module Checklist

# Programming with Python

By Techworld with Nana

## Video Overview

DEVOPS BOOTCAMP

★ Introduction to Python
★ Install Python & Local Setup
★ Our first program
★ Python IDE vs simple file editor
★ Strings & Numbers
★ Variables
★ Functions
★ User Input
★ Conditionals (if / else) & Boolean Data Type
★ Error Handling with Try / Except
★ While Loops
★ Lists and For Loops
★ Comments
★ Sets
★ Built-In Functions
★ Dictionaries
★ Modules
★ Demo Project: Countdown App
★ Packages, PyPi & pip
★ Demo Project: Automation with Python - Working with Spreadsheets
★ Classes and Objects
★ Demo Project: API Requests

| Demo Projects | |
|---|---|
| Git Project | https://gitlab.com/nanuchi/python-programming |

# Check your progress... 1/5

## Introduction to Python

- ❏ Watched video

## Install Python & Local Setup

- ❏ Watched video
- ❏ **Install Python & PyCharm:**
  - ❏ Python3 installed
  - ❏ PyCharm downloaded & setup

**Useful Links:**

- Python for different OS:
  - https://www.python.org/downloads/
  - Installation Guide for Windows: https://docs.python.org/3/using/windows.html#installation-steps
- PyCharm Download: https://www.jetbrains.com/pycharm/download/

## Our first program

- ❏ Watched video
- ❏ **Demo executed**

## Python IDE vs simple File Editor

- ❏ Watched video

## Check your progress... 2/5

### Strings and Numbers

- ❏ Watched video
- ❏ **Demo executed**

### Variables

- ❏ Watched video
- ❏ **Demo executed**

### Functions

- ❏ Watched video
- ❏ **Demo executed**

### User Input

- ❏ Watched video
- ❏ **Demo executed**

### Conditionals (if / else) & Boolean Data Type

- ❏ Watched video
- ❏ **Demo executed - validate User Input to make our program more robust**

### Error Handling with try / except

- ❏ Watched video
- ❏ **Demo executed**

## Check your progress... 3/5

### While Loops

- ❏ Watched video
- ❏ **Demo executed**

### Lists and For Loops

- ❏ Watched video
- ❏ **Demo executed**

### Comments

- ❏ Watched video

### Sets

- ❏ Watched video
- ❏ **Demo executed**

### Built-In Functions

- ❏ Watched video
- ❏ **Demo executed**

### Dictionaries

- ❏ Watched video
- ❏ **Demo executed**

# Check your progress... 4/5

## Modules

- ❏ Watched video
- ❏ **Demo executed**

## Project: Countdown Application

- ❏ Watched video
- ❏ **Program implemented**
  - ❏ Get user Input: goal and deadline
  - ❏ Convert deadline string into a proper date type
  - ❏ Calculate how many days are left until the deadline
  - ❏ Print a proper message to the user including the entered goal
  - ❏ Calculate how many hours are left until the deadline

**Exercise:**

Write a program that accepts a user input of a goal and a deadline date.

And print back to the user, how much time they have remaining till that deadline.

## Packages, PyPi & pip

- ❏ Watched video
- ❏ **Demo executed**

# Check your progress... 5/5

## Project: Automation with Python - Spreadsheet

- ❏ Watched video
- ❏ **Program implemented**
  - ❏ Exercise 1 - List companies with respective product count
  - ❏ Exercise 2 - List companies with respective total inventory value
  - ❏ Exercise 3 - List products with less than 10 in inventory
  - ❏ Exercise 4 - Write to Spreadsheet and save into a new File

## Classes and Objects

- ❏ Watched video
- ❏ **Demo executed**

## Project: API Requests

- ❏ Watched video
- ❏ **Program implemented - API Request to list your GitLab projects**

# More Resources...

## Best Practices

- Use Spaces for Indentation

Naming Conventions for Variables and Function Names

- Use **descriptive variable names**!
- Use lowercase and underscores to separate words
- Class file should be lowercase, whereas the class name itself should start with an uppercase. Examples: *user.py* file, but *class User*
- Constants are usually defined on a module level and written in all capital letters with underscores separating words. Examples: MAX_OVERFLOW and TOTAL

Comments:

- **Don't overuse comments**! The program itself with descriptive variable and function names should already be very descriptive.
- Only use comments, when it adds more information. Example: *a + b*, bad would be to write a comment here: *# here I add a plus b*

Functions:

- DRY - Don't Repeat Yourself - a general principle of software development aimed at reducing repetition. Whenever you code something more than once you can actually create a function for the logic or a variable for this value.
- Functions should be small and encapsulate one single functionality. If your function needs many parameters, you probably include too much logic and the function should rather be splitted. With this approach you will also have no problem in naming the function and it's easier to reason about the program.

Check out Official Style Guides for Python here:
https://www.python.org/dev/peps/pep-0008/