Unit 1: Programming

1. Hello world

Introduction

Programming is one of those skills where you will get out of it what you put in. If you make an investment of time, your skills will develop and grow. If you do not invest the time, your skills will stagnate or even deteriorate. While programming is most often thought of as a Math or Science derived skill, it is also very much a Language acquisition. Just like learning any other language, a little bit of practice every day, or every second day, goes a long way. The above is a long way to say this: To do well, you should be aiming to spend 30 minutes every two or three days practicing your programming and algorithmic skills (outside of class).

Learning objectives

- Sign up for https://repl.it/ and share your account with Mr Baumgarten
- Run your first Python program
- Understand the basic rules of the Python programming language

Discussion & learning items

Your first program in Python:

```
print("Hello world")
```

That may look straightforward but there are a few things to take careful note of from the very beginning.

- Python is case sensitive: `print` is different to `Print` is different to `PRINT`
- Most instructions require a set of parenthesis (technical name for rounded brackets)
- Inside those parenthesis go the parameters. This is the information the instruction requires in order to run.
- If the instruction requires multiple parameters, the order of the parameters matters. You would separate each parameter with a comma.
- The above instruction contains one parameter, "Hello world!".
- When we need to provide text information, we use "quotes" to indicate the start and end of the text.

Exercises

- Move on to lesson 2

Further practice & resources

Further Python programming practice is available through this Repl.it class. I am the registered teacher for this class so will automatically be able to see your progress on these exercises.

- https://repl.it/data/classrooms/share/9d68c5288e184810f61b489d14ccfa54

2. Data types, variables & numbers

Learning objectives

- 2.2.1a declare and use variables and constants
- 2.2.1b understand and use basic data types: Integer, Real
- 2.2.1d use predefined procedures/functions

Introduction

- Read **Chapter 2 Numbers** of Python Programming for Secondary school students.

Discussion & learning items

There are a few simple rules when it comes to naming variables:

- Must start with a letter
- Case sensitive - NAME and name would be considered two different variables
- Can contain letters, numbers, underscores
- Can not use spaces, hyphens, dots, or other punctuation symbols

All of these are valid…

```
person_name = "Angel"
personName23 = "Barry"
PersonName = "Claire"
PERSONname = "Darren"
PeErsON__naME = "Elise"
```

To help prevent it getting confusing (so you can remember what you called your variables later) it is recommended to stick to one system.

The preferred system with Python is

- All lower case
- Use underscores to join words together

Variables come in different types, depending on the data within it. Four common data types:

- Integers - whole numbers (no decimal)
- Floating point numbers - numbers that are permitted to have decimals
- Booleans - Only two possible values - True or False
- Strings - Text. We use the quote characters to indicate the start and end point of our text.

We use different data types based upon what we want to do with the value. Python will permit different functionality based on the type.

```
age = 11          # Integer
height = 1.56     # Float
name = "Mary"     # String
```

```
person = True     # Boolean
```

For example if we are wanting to perform numerical calculations, or use a value to indicate coordinates, we would probably use an integer, or perhaps a float. To store a person's name or their address, use a String.

Python can be instructed to set a variable to the result of a calculation as well. These calculations can also involve looking up values from existing variables.

Example 1

```
minutes_per_day = 60 * 24                # A simple calculation
print(minutes_per_day)
```

Example 2

```
minutes_per_hour = 60
minutes_per_day = minutes_per_hour * 24   # Looks up an existing variable
print(minutes_per_day)
```

Example 3

```
minutes = 60
minutes = minutes * 24      # Look up the value in minutes
                            # Multiply by 24
                            # Save value back into minutes replacing previous value
print(minutes)
```

The variable receiving the new value must always be on the left of the assignment operator (the equal sign).

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to numbers".
- Complete the problem set on page 14 of my Python Programming book - questions 1 to 7.
- Submit your repl to the Google Classroom assignment.

Further practice & resources

- https://repl.it/data/classrooms/share/9d68c5288e184810f61b489d14ccfa54

3. More about numbers

Learning objectives

- Continue practicing mathematical operations with Python

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code: Create a new repl named "intro to numbers 2".

1. Area of a non-right angled triangle calculator. Given values for length a, length b and angle in degrees C, return the area of the triangle (remember you will have to convert degrees to radians first).
2. Surface area of a sphere calculator given a radius
3. Volume of a cone calculator
4. Work out the Python order of operations by testing the following `3/4+5*2/33-3+8*3` .
5. *Only if you have learnt the quadratic formula already in math...* For any given values for a, b and c, will provide the solutions to the quadratic formula (you may assume both solutions are required). Be careful with your order of precedence. Here is an example solution set for testing: If y=2x^2-4x-10 then the solutions are 3.44949 and -1.44949.

Submit your repl to the Google Classroom assignment.

Further practice & resources

- https://repl.it/data/classrooms/share/9d68c5288e184810f61b489d14ccfa54

4. Strings, casting, string operations

Learning objectives

- 2.2.1b understand and use basic data types: String
- 2.2.1d use predefined procedures/functions

Introduction

- Read **Chapter 3 Strings** of Python Programming for Secondary school students.

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to strings".
- Complete the problem set on page 22 of my Python Programming book.
- Submit your repl to the Google Classroom assignment.

Further practice & resources

- https://repl.it/data/classrooms/share/9d68c5288e184810f61b489d14ccfa54

5, 6. Truthiness & selection

Learning objectives

- 2.2.1c understand and use the concepts of
    - sequence,
    - selection,
- 2.2.1d use predefined procedures/functions

Introduction

- Read **Chapter 4 Making decisions** of Python Programming for Secondary school students.

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to selections".
- Complete the problem set on page 27 of my Python Programming book.
- Submit your repl to the Google Classroom assignment.

Further practice & resources

- https://repl.it/data/classrooms/share/9d68c5288e184810f61b489d14ccfa54

7, 8. Repetition

Learning objectives

- 2.2.1c understand and use the concepts of
    - sequence,
    - selection,
    - repetition,
    - totalling and
    - counting
- 2.2.1d use predefined procedures/functions

Introduction

- Read **Chapter 5 Repetition** of Python Programming for Secondary school students.

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to while loops".
- Complete the problem set on page 30 of my Python Programming book.
- Create a new repl named "intro to for loops".
- Complete the problem set on page 32 of my Python Programming book.
- Submit your repl to the Google Classroom assignment.

Further practice & resources

- https://repl.it/data/classrooms/share/9d68c5288e184810f61b489d14ccfa54

9. Quiz

10, 11, 12. Lists, list manipulation & list looping

Learning objectives

- 2.2.2a declare and use one-dimensional arrays, for example: `A[1:n]`
- 2.2.2b show understanding of the use of one-dimensional arrays, including the use of a variable as an index in an array
- 2.2.2c read or write values in an array using a `FOR ... TO ... NEXT` loop

Introduction

- Read **Chapter 6 Lists** of Python Programming for Secondary school students.

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to lists".
- Complete the problem set on page 40 of my Python Programming book.
- Submit your repl to the Google Classroom assignment.

Further practice & resources

- https://repl.it/data/classrooms/share/9d68c5288e184810f61b489d14ccfa54

13, 14. Functions

Learning objectives

- Creating user defined functions
- Requiring parameters on user defined functions
- Supplying a return value on user defined functions

Introduction

- Read **Chapter 8 Functions** of Python Programming for Secondary school students.

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to functions".
- Complete the problem set on page 59 of my Python Programming book.
- Submit your repl to the Google Classroom assignment.

15, 16. Exceptions

Learning objectives

- Best practice on identifying and catching error events known as Exceptions

Introduction

- Read **Chapter 10 Exceptions** of Python Programming for Secondary school students.

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to exceptions".
- Complete the problem set on page 69 of my Python Programming book.
- Submit your repl to the Google Classroom assignment.

17, 18. Read & write files

Learning objectives

- Read a string or list of strings from a text file
- Write a string or list of strings to a text file
- Read a list of strings from a CSV file
- Write a Python object to a pickle file
- Read a Python object from a pickle file

Introduction

- Read **Chapter 9 Files & folders** of Python Programming for Secondary school students.

Discussion & learning items

Text files

The most basic type of file is the text file. It is basically just one big text string. Anytime you read a text file, Python will load it as string values. Likewise anytime you write to a text file, Python will expect you to supply string values. This does mean if you want to read/write numbers or other data, you will have to convert to/from strings in order to do so.

CSV files

CSV files, or comma-separated-values, are a commonly used method of importing and exporting data between programs. Excel and Google Sheets will both allow you to save spreadsheets as CSV files which can make for a convenient way of quickly loading a series of complex values into your Python program.

Pickle files

Pickle files are actually addressed in chapter 14 of the book, but you don't need to worry about the extra complexity of classes and objects in order to make use of them.

Pickle files can be thought of as Python's built-in way of being able to save (almost) any Python variable to a file, such that you can load it back at a later point. You don't have to manually convert everything to/from strings like you do with text files.

For example, the following will save a list of values to a file, and then load it back as a list again.

```
import pickle

values = ["Hello", 3.14, False, 42, "Hmmmm"]

# Save the list to file info.pickle
with open("info.pickle", "wb") as f:
    pickle.dump(values, f)

# Load info.pickle into a new list
```

```
with open("info.pickle", "rb") as f:
    values2 = pickle.load(f)

print(values2)
```

Exercises

Practice reading code - Complete the online quiz when prompted during the lesson.

Practice writing code

- Create a new repl named "intro to files".
- Complete the problem set on page 64 of my Python Programming book.
- Submit your repl to the Google Classroom assignment.

Further practice & resources

- To-do app on page 64
- Closest cities problem on page 65

19. Quiz