Chapter 2

The Basics

Learning Objectives

- Data(types)
- Variables
- Basic Arithmetic
- Basic functions
- Debugging

2.1 Data(types) and Variables

Programming revolves around applying operations to data. It is therefor important to know about your data. Python is not explicitly typed, that means you do not have to say what type of data you expect. There is however still this concept of data types. Not all operations can be applied to the all types of data and operations behave differently based on the data you give it.

To learn the type of *something* you can put `type()` around it.

That *something* can be either a *literal* or a *variable*. A literal is when you input the value you want directly. A variable refers to a location in computer memory where the value you want is stored.

2.2 Naming

As we have discussed you write your program for humans. If you take care when naming things in your program you'll find that your program reads almost like regular English. There are some conventions to keep in mind, following these conventions lead to more beautiful (i.e. readable) code.

Variable names should not contain capital letters. Variable names cannot contain spaces, if a name consists of multiple words put an underscore (_) instead of space.

These are only a few of the conventions for Python. You can find all of them in PEP 8 `https://www.python.org/dev/peps/pep-0008/`. A PEP or Python Enhanced Proposal, it is the way new features are added to the language. PyCharm will give grey squiggles when you break a PEP8 convention and will try to help you to fix it.

2.3 Debugging

The book (sections 4.11 and 4.12) gives a good overview of *why* you should embrace debugging. However, we'll get to that when we get to it. The method it suggests for debugging works fine, but given that we have PyCharm, we have a better tool.

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_str)

final_time_int = current_time_int + wait_time_int

final_answer = final_time_int % 24

print("The time after waiting is: ", final_answer)
```

Listing 1: Code we will debug

The PyCharm Debugger When you start the PyCharm debugger the view changes to what you see in Figure 2.1. The code used here is the sample code *db_ex3_4* from section 3.2, see Listing 1.

The important functions of the debugger are highlighted in Figure 2.1. This is how you use them:

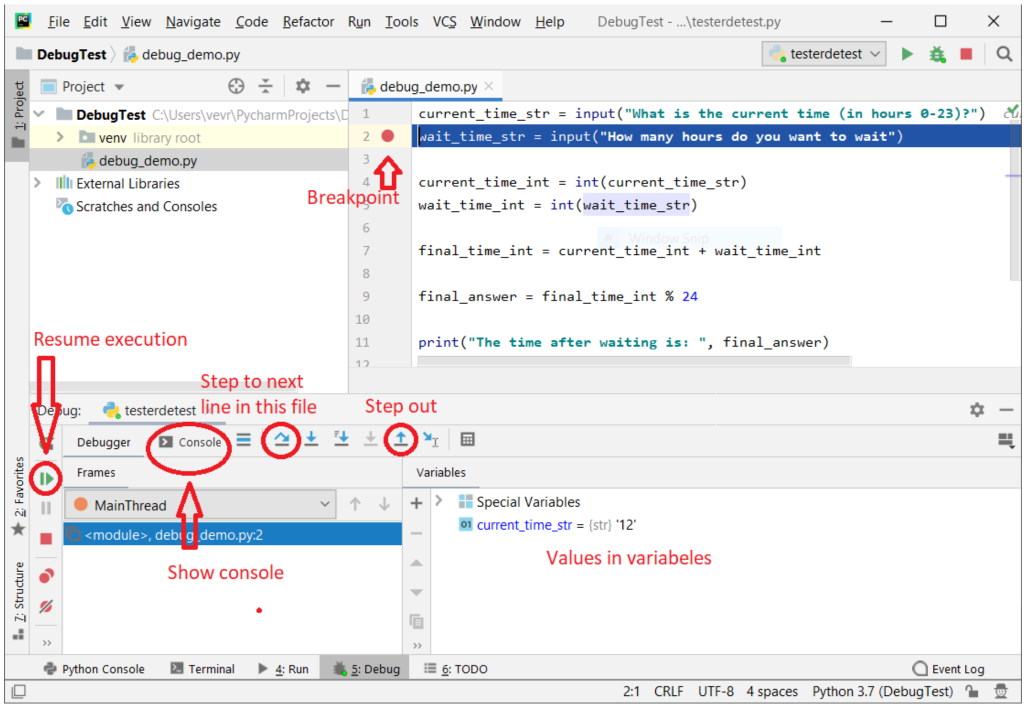
Breakpoints When you start the debugger your program runs until it hits a breakpoint. The program now shows you the state of the computer so you can learn what is going on.

Resume execution Resumes your program until it hits another breakpoint.

Step to next line in this file Executes one command, but stays in the current file. You could also step to other files but typically you'll want to look at your own code.

Step out This one will become useful later.

Console Here you can switch between the Console and the Debugger, depending on what information you need at the moment.



```python
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_str)

final_time_int = current_time_int + wait_time_int

final_answer = final_time_int % 24

print("The time after waiting is: ", final_answer)
```

```
current_time_str = {str} '12'
```

Figure 2.1: The PyCharm debugger

2.4 Homework

Reading

- Chapter 2: Variables, expressions and statements

Assignments

1. The exercises in 2.14
2. Write a program to calculates the time it takes to get somewhere using different modes of travel (walking, biking, driving by car).
    - Your program must ask the user for the distance (in Km).
    - Your program outputs the travel times for the different modes of travel.
    - The first version of your program should be as simple as possible, get it to work first, add complexity later!
    - Your program may ask the user for additional information, or you can assume values for these, e.g.;
        - Speeds for walking/biking/driving,
        - Time to get going (take bicycle out of the shed, scratch ice from the window),
        - Time to find parking (and, when some distance away, time to get to the actual destination).