Contents

1 Part 1: Tuesday’s Lecture ........................................ 2
   1.1 Task 1: Hello World ........................................ 2
   1.2 Task 2: Built-ins ........................................... 2
   1.3 Task 3: Store Some Values to Constants .................. 3
   1.4 Task 4: Read in Input ...................................... 4
   1.5 Task 5: Test Out the Program .............................. 5
   1.6 Task 6: Convert the Different Values to Other Values .. 5
   1.7 Task 7: Test Out the Program Again ...................... 5
   1.8 Task 8: Perform the Final Calculation ................... 6
   1.9 Task 9: Test Your Program ................................ 6

2 Part 2: Thursday’s Lecture ..................................... 7
   2.1 Task 10: Typechecking .................................... 7
   2.2 Task 11: File Reading ...................................... 7
   2.3 Task 12: Calculations ..................................... 8
   2.4 Task 13: Printing it out .................................. 8

3 Hours Worked .................................................. 9

4 Extra Credit ................................................... 9

5 Handin .......................................................... 9
1 Part 1: Tuesday’s Lecture

1.1 Task 1: Hello World

Create a new Python script that prints ”Hello World!”. On two additional lines, print out a number of your choice and a boolean value of your choice. Both the number and the boolean should not be in string quotes.

1.2 Task 2: Built-ins

Consult Python’s built-in functions list. Find two functions of interest, besides print(), which require an input and return a result of some kind. That is, if you print out their result, it should not be a NoneType. For each function, print out the results of three valid examples of their use. Your program should not throw any exceptions (i.e. crash with an error).

For each different function, write a brief 1-line comment describing what you think the function does.

To make your search easier, this list contains a number of simple functions that require only one or two parameters. It is recommended you choose from this list.

- abs()
- chr()
- eval()
- float()
- hash()
- hex()
- id()
- input()
- int()
- len()
- long()
- max()
- min()
- oct()
- ord()
- pow()
- range()
- round()
- str()
- type()

Note: For the remaining Tasks of Part 1 (Task 3 – Task 9), we are ultimately concerned with computing how many calories the user burns per minute, which is based on their running speed and body weight. Each task is intended to guide you through creating the program, one part at a time.
1.3 Task 3: Store Some Values to Constants

Assigning constant numbers to variables allows us to give names to specific values, so while writing a program, it is easier to understand the formulas we use. For example, we will create a variable to keep track of the number of meters per mile. This is a much better practice than placing the value 1609 directly in the formula; that is, merely using 1609 directly in our code, without assigning it to a variable, would lack context and thus probably be quite confusing to anyone (including yourself) who would later look at your code.

\[
\text{meters\_per\_mile} = 1609
\]

When creating variable names, Python has strict rules for naming variables, and our course has a few more. Python won’t run your code if you don’t follow its rules and it won’t care about our rules, which are just good guidelines for a consistent style. We will also deduct points if you don’t follow our rules.

Python’s Rules for Creating Variable Names

- Variable names must contain only the characters a-z, A-Z, 0-9 and underscores.
- Variable names are case sensitive.
- A letter must be the first character (98\_degrees is not valid, but boyz\_2\_men would be)

CS0030’s Style Guide Rules

- When naming variables, only use lowercase characters and separate words with underscores.
- Variable names should contain full words or common abbreviations (meters\_per\_mile, not m\_per\_mi).
- Don’t use a reserved Python keyword (anything that is colored in your text editor).
- Variable names should include a word that eludes to its type:
  - integer: e.g. count, number, index, etc.
  - float: e.g. quantity, weight, mass, etc.
  - string: name, title, word, sentence, description, etc.
  - boolean: is\_ascending, is\_sorted, is\_full, etc.

Assign the following values to variables with meaningful but concise names. Each variable assignment should occur on their own line.

- The number of meters in a mile: 1609
- The respiratory exchange ratio: 4.86
- the number of pounds in a kilogram: 2.2
1.4 Task 4: Read in Input

This program requires two pieces of information from the user running the program.

- The number of minutes it takes for them to run a mile
- Their weight in pounds

Use `input()` to ask the user for each piece of information, and assign the responses to their own respective variables. `input()` can accept an optional string input, which it will print before accepting input from the user — e.g., `input("Who are you?")`. Then, after being prompted with the input question, the user running the program will then be able to type whatever they want, and when they press Enter, `input()` will return whatever the user typed. Importantly, whenever you use `input()`, it reads in each value as a string, even if you type only a number. To use it in our calculations, you must cast (convert) it to a `float`. You can do that with the `float()` function. Again, your variable names should be in all lowercase with words separated by underscores.

For example, to collect and assign the user’s mile time to variable, we could write

```python
minutes_per_mile_string = input("How many minutes does it take for you to run a mile?")
minutes_per_mile = float(minutes_per_mile_string)
```

Perform the same task to collect the user’s weight.

To keep the code readable, it is often helpful to manually wrap lines of code. In the above lines of code, the first line is actually 88 characters long. As lines of code get longer and longer, it makes it harder to read. We have difficulty tracking lines of code horizontally. Therefore, it is often a requirement in style guides used by professional developers to keep lines of code to a strict limit. For this course, we ask that you keep your lines of code to a max of 80 characters.

Python will attempt to interpret each line of code as its own expression, so when you break a line of code, you must tell Python explicitly that you are including a line break. You can do this by using a backspace character (`\`). For the next line of code, it should be indented to help our eye see that it is a continuation of the previous line. Sublime and Atom should automatically indent, but if not, press tab to indent the line.

```python
minutes_per_mile_string = \\
   input("How many minutes does it take for you to run a mile?")
minutes_per_mile = float(minutes_per_mile_string)
```
1.5 Task 5: Test Out the Program

Use `print()` to print out the two number variables you just created in Task 4. For example, you can print out the minutes per mile value with `print(minutes_per_mile)`. Run your program at this point to ensure that it works as expected and doesn’t have any errors. It should ask the user for the two pieces of information and then print it out.

Once you are confident the program works as expected, remove the print statements and continue.

1.6 Task 6: Convert the Different Values to Other Values

We need to convert the user’s mile time to a running speed with units of meters per minute. We can do this in two steps. First, we convert mile time to the time it takes to run a meter. Divide minutes per mile (`minutes_per_mile`) by the meters per mile (`meters_per_mile`) to calculate the user’s time to run 1 meter. For example, I would write the following

```
minutes_per_meter = minutes_per_mile / meters_per_mile
```

Be sure to replace the variable names above with the variable names with those that you defined yourself.

Then, to calculate their speed in meters per minute, you just need to invert the minutes per meter value.

```
meters_per_minute = 1.0 / minutes_per_meter
```

Convert their weight in pounds to their weight in kilograms. You will need to divide their weight in pounds by 2.2, the number of pounds in a kilogram. You stored this value to a variable in a previous task. Be sure to use your variable instead of the actual number.

1.7 Task 7: Test Out the Program Again

Before continuing on, add print statements to print out the user’s speed in meters per minute and their weight in kilograms variable. Run the program again, this time after asking for the user’s inputs, it should print out those two values. If your program throws any exceptions (i.e. has an error and crashes), or it doesn’t output the correct values, carefully look at your code to identify any issues. Try using 155 pounds for the weight and 10 minutes for the mile, it should come out to 70.5 kilograms and 160.9 meters per minute.
1.8 Task 8: Perform the Final Calculation

Estimate their oxygen consumption with the following formula

\[
\text{estimated } O_2 \text{ consumption} = 0.2 \times (\text{running speed in meters / min}) + \frac{3}{5}
\]

Now you can estimate their final calories per minute burned when running. It is just \(0.005 \times (\text{weight in kg}) \times (O_2 \text{ consumption})\). Assign their calories burned per minute value to a variable.

**Output the Converted Values Using String Concatenation**

String concatenation in Python requires that each value that is concatenated must be of type string. Therefore, to print the sentence, “You burn approximately 15 calories per minute”, you would have to write something like

```python
print("You burn approximately " + str(calories_per_minute) + " calories per minute running")
```

In a nicely worded sentence like the example above, print out their weight, their running speed, and the amount of calories burned per minute.

1.9 Task 9: Test Your Program

We need to further verify that the program works correctly (even if it can run completely through without an exception). Put in a weight of 155 lbs and 10 minutes for the weight and mile time. It should result in about 11.5 calories/minute. Then try 180 lbs and 8 minutes. The result should be about 16.7 calories/min.
2 Part 2: Thursday’s Lecture

2.1 Task 10: Typechecking

What happens if you ask for someone’s weight in pounds and the user types blue? We can perform something called typechecking, where we see if we’re getting the right kind of information from a source (source could be a user, a dataset, or a function, among other things).

There are multiple ways to do this! For example, if you ask for weight and the user types in a number, the number is still interpreted as a string. The first thing you need to do with the input is convert it to a float. But if the user puts in blue, you’ll get an error saying that you can’t convert the string blue to a float value.

Typechecking is how we avoid these errors that will crash our programs! One way we can type-check is using try and except blocks. The syntax goes something like this, but you can use your researching skills to figure out how to adapt it to your program:

```python
try:
    # try running code in this block
    float('a')
except Exception:
    # this block will run when any exceptions are thrown in the try block
    print("Error")
```

Let’s typecheck all of the inputs on the previous part of the homework. Now your user can screw up and the program won’t crash :)

2.2 Task 11: File Reading

This part of the homework asks you write a program that reads Office Character information from a file and analyzes the data. This is real, actual data that we fact checked.

Download the hw03_source.py file to your cs0030 directory. It contains stencil code that you will complete.
Your program should begin by reading a line from the office.csv file which contains the following information:

- Their name
- Height (inches)
- Age (years)
- Weight (pounds)
- Activity Level (0-4), where 0 is not very active and 4 is the most active

### 2.3 Task 12: Calculations

Calculate the Basic Metabolic Rate for men, which is estimated by:

\[
BMR = (10 \times \text{weight in kg}) + (6.25 \times \text{height in cm}) - (5 \times \text{age in years}) + 5
\]

Calculate the Basic Metabolic Rate for women, which is estimated by:

\[
BMR = (10 \times \text{weight in kg}) + (6.25 \times \text{height in cm}) - (5 \times \text{age in years}) - 161
\]

Calculate their daily recommended intake, which is estimated by:

\[
DRI = BMR \times (1.2 + (0.175 \times \text{activity level}))
\]

*Hint:* There are 2.54 cm per inch! Would it help to store this as a variable?

### 2.4 Task 13: Printing it out

In a nicely formatted and well-worded paragraph, print the following using string concatenation:

- Their name
- Height in centimeters
- Weight in kilograms
- Basic Metabolic Rate if they are a man
- Basic Metabolic Rate if they are a woman
- Daily Recommended Intake if they are a man
- Daily Recommended Intake if they are a woman
Your (first bit of) output for this task might look something like this:

```
carolinerebet@macbook:Desktop carolinerebet$ python week3answerkey.py
Dear Michael Scott. You are 175.26 cm tall. You weight is 77.272727272727277 kilos. If you are a man, your Basic Metabolic Rate is 1623.10227272727275 and your Daily Recommended Intake is 2231.7656249999995 calories. If you are a woman, your Basic Metabolic Rate is 1457.10227272727275 and your Daily Recommended Intake is 2003.5156249999998 calories.

Dear Jim Halpert. You are 190.5 cm tall. You weight is 77.272727272727277 kilos. If you are a man, your Basic Metabolic Rate is 1768.3522727272725 and your Daily Recommended Intake is 2740.94602272727272 calories. If you are a woman, your Basic Metabolic Rate is 1602.35227272727275 and your Daily Recommended Intake is 2483.6460227272723 calories.
```

3 Hours Worked

Please comment at the top of your script the number of hours you worked on this assignment, whether you went to TA hours and any students you collaborated with.

4 Extra Credit

As with every homework, you are welcome to expand on the tasks above for extra credit. For example, can you improve your program so that by asking an additional question, your program will only need to compute a single BMR and DRI specific to the user’s gender? Alternatively, regardless of the user’s chosen activity level, can you find a way to limit the actual range to 0-4 (e.g.: if they input 5, it gets converted automatically to 4, or -2 to 0)? Lastly, you could also choose to additionally implement another calculator based on the given information above, like BMI, target heart rate, etc. These are just ideas! Feel free to use your imagination :)

5 Handin

Name your submission file hw03.py and submit it through Canvas before Monday at 11:59PM. hw03.py should contain:

Tasks 1-13 (Task 2 can be commented out with ```` since it'll include English sentences)