

Python Chapter 2 Getting Started Activities

Contents

Python Chapter 2 Getting Started Activities	1
Practice, Practice, and More Practice - Neural Plasticity	1
Online Tutorials.....	2
Tutorial 1: Hello World	3
Requirements	3
Algorithm	3
Pseudocode	4
TODO.....	4
Tutorial 2: Inches to Centimeters	4
Input - Process - Output.....	6
Input	6
Process	6
Output	6
Tutorial 3: Fun with Math	7
Assignment 1: Susie's Square Calculator.....	10
Assignment Submission.....	11



No AI use.

Time required: 90 minutes

Practice, Practice, and More Practice - Neural Plasticity

Practice makes permanent.

We are going to provide you with a lot of information. You will need to put your brain in gear and write a lot of programs to retain the information. Very few people become expert athletes or musicians without a lot of practice. Similarly, very few people become expert programmers without a lot of practice.

A little bit of programming each day will strengthen the neural connections in your brain. Consistent daily practice literally re wires your brain. This is called **neural plasticity**.

If you want to learn to program, plan to spend a lot of time in front of your computer, not just reading, but programming as well.

Neural pathways are **strengthened** into habits through the repetition and practice of thinking, feeling, and acting.

PRACTICE: Start your morning passionately declaring aloud your goals for the day. Write them down in a to do list. Declarations send the power of your subconscious mind on a mission to find solutions to fulfill your goals.

Online Tutorials

Go through the following tutorials.

- [LearnPython.org Hello World](#)
- [LearnPython.org Variables and Types](#)
- [Python Introduction](#)
- [Python Get Started](#)
- [Python Syntax](#)
- [Python Comments](#)
- [Python Variables](#)
 - [Variable Names](#)
 - [Assign Multiple Values](#)
 - [Output Variables](#)
 - [Global Variables](#)
- [Python Data Types](#)
- [Python Numbers](#)
- [Python Casting](#)
- [Python Strings](#)
- [Python Operators](#)

- [Python User Input](#)

Tutorial 1: Hello World

Hello World is the traditional first program to create in any programming language. It confirms that your programming environment is setup properly.

Create a Python program file called **hello_world.py**

Let's create this program in the style we will use for this class.

1. Type in the following code.

```
1  """
2      Name: hello_world.py
3      Author:
4      Created:
5      Purpose: My first Python program
6  """
```

This is a header for the program. It is a multiline comment letting whoever is reading the code know what the program is about. Python ignores comments.

```
8  # Print the literal string Hello World!
9  print("Hello Python World! Let's start coding!!")
```

- The first line is a comment about what the second line does.
- The second line prints text to the console.

Example run:

```
Hello Python World! Let's start coding!!
```

Requirements

We are going to go through a development process to create a program.

Algorithm

Every program you write solves some type of problem. An algorithm is the method of solving that problem.

Pseudocode

Pseudocode is one way of planning out your program in simple natural language.

Pseudocode is the first step of translating the algorithm to the programming language. The more you figure out before you code, the easier it is to code an efficient and elegant program. An elegant, simple program is like a work of art.

```
Create a greeting for your program.  
Ask the user for the city that they grew up in.  
Ask the user for the name of a pet.  
Combine the name of their city and pet.  
Display their band name.
```

TODO

A **TODO** list is another way to plan out your program.

Anything after the # sign is a comment. Python ignores this line.

Copy and paste the following code to get started with this program.

```
# TODO: Create a greeting for your program.  
  
# TODO: Ask the user for the city that they grew up in.  
  
# TODO: Ask the user for the name of a pet.  
  
# TODO: Combine the name of their city and pet.  
  
# TODO: Display the band name.
```

Tutorial 2: Inches to Centimeters

Getting numeric input from the user involves another method of input. All input from the keyboard comes in as a string. To convert the string to a number we use the **float()** or **int()** function.

The **float()** function in Python converts a number or a string that represents a floating-point number into a floating-point value. It returns a floating-point number, allowing numeric operations with decimal points.

```
# Get string input from the user  
inches = input("Enter inches: ")  
# Convert a string to a float number like 3.114  
inches = float(inches)
```

In Python, we typically combine both of those operations in a single line.

```
# Get a string from the user, store a decimal number in the inches variable
inches = float(input("Enter inches: "))
```

Copy and paste the following code to get started with this program.

```
"""
    Name: inches_to_centimeters.py
    Author:
    Created:
    Purpose: Convert Inches to Centimeters
"""

# TODO: Print a nice title for our program

# TODO: Get inches input from user, cast to float

# TODO: Convert inches to centimeters

# TODO: Display the centimeter result
```

1. Create a new Python program named **inches_to_cm.py**.
2. Type in the following code.

```
1  """
2      Name: inches_to_centimeters.py
3      Author:
4      Created:
5      Purpose: Convert Inches to Centimeters
6  """
```

This is a multiline comment that gives information about the program. It is helpful to have a descriptive header in a program.

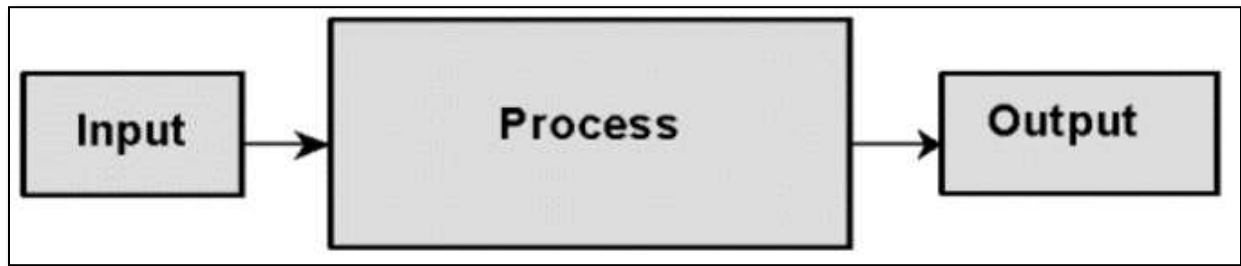
```
8  # TODO: Print a nice title for our program
9  print("-----")
10 print("|      Inches to Centimeters Converter      |")
11 print("-----")
```

The first line is a single comment to let another programmer know what is going on.

The next lines print the program title to the console.

Input - Process - Output

All programs have three basic components.



Input

```
13 # TODO: Get inches input from user, cast to float
14 inches = float(input("Enter inches: "))
```

The first line is a comment describing what happens in the second line.

The next line is an input function. Its job is to ask the user to type something in and to capture what the user types. The part in quotes is the prompt that the user sees. It is called a string and will appear to the program's user exactly as it appears in the code itself.

We use the **float()** function to convert the incoming string to a float.

We use the assignment operator **=** to assign the float value to the **inches** variable.

Process

The process section is where the program does a calculation of some kind.

```
16 # TODO: Convert inches to centimeters
17 centimeters = inches * 2.54
```

This line uses the standard math formula to convert centimeters to inches. Notice that the **=** is different in programming than in math. It is called the assignment statement. The calculations or values on the right are assigned to the variable on the left.

Output

```
19 # TODO: Display the centimeter result
20 print(inches, "inches is equal to", centimeters, "centimeters.")
```

The last line uses the print function to print out the conversion result.

Run the program by pressing the F5 key.

The program will ask you to Enter inches. Type in 24 and press enter.

Example run:

```
-----  
|   Inches to Centimeters Converter   |  
-----  
Enter inches: 24  
24.0 inches is equal to 60.96 centimeters.
```

This program may seem too short and simple to be of much use. There are many websites that do similar conversions. Their code is not much more complicated than the code here.

Tutorial 3: Fun with Math

This tutorial shows some of the common math operations and math library functions. `+-*/` are not included as they are assumed. Some math functions are part of the standard Python library, some are part of the math library.

```
1  """  
2      Name: fun_with_math.py  
3      Author: William A Loring  
4      Created: 08/01/22  
5      Purpose: Python math demonstration - exploring various mathematical operations.  
6  """  
7  
8  import math  
9  import random  
10 import statistics
```

Start by importing the math and random libraries for later use.

Modulus (%) - Remainder after division. This is the remainder when 45 is divided by 2.

For example, this can be used to find out if a number is even or odd.

```
mod = 45 % 2  
print(f"Modulus of 45: {mod}")
```

Here is a function to demonstrate a use of modulus. This code also includes a very nice docstring. (If you aren't sure what a docstring is, please look it up.)

```

21 def is_even(number):
22     """
23     This function checks if a number is even using the modulus operator.
24
25     Args:
26     |   number: The number to check for evenness.
27
28     Returns:
29     |   True if the number is even, False otherwise.
30     """
31
32     # The modulus of a number divided by 2 will be 0 if the number is even.
33     return number % 2 == 0
34
35
36 # Example usage:
37 num = 10
38 if is_even(num):
39     print(f"{num} is an even number.")
40 else:
41     print(f"{num} is an odd number.")

```

- **def is_even(number):** defines a function named `is_even` that takes a single argument `number`.
- **return number % 2 == 0** checks if the remainder (%) when `number` is divided by 2 is equal to 0. If it is, the number is even, and the function returns `True`. Otherwise, it returns `False`.
- **if is_even(num):** calls the `is_even` function with `num` and checks the returned value (`True/False`).

In Python, if a number is evenly divisible by 2, the remainder after division (modulus) will be 0. This makes the modulo operator a convenient way to identify even or odd numbers.

Integer division (//) - Whole number quotient, discarding the remainder. This gives the whole number result, which is 7. This gives the same result as **math.floor()**.

```

int_div = 65 // 9
print(f"Integer division 65 // 9: {int_div}")

```

math.floor() - Rounds a number down to the nearest whole number (integer).


```
flr = math.floor(65 / 9)
print(f"math.floor rounds 65 / 9 down: {flr}")
```

math.ceil() - ceiling rounds a number up to the nearest whole number (integer).

```
cel = math.ceil(65 / 9)
print(f"math.ceil() rounds 65 / 9 up: {cel}")
```

round() - Rounds a number to a specified number of decimal places.

```
rnd = round(20.0 / 3.0, 2)
print(f"Rounded to 2 decimal places: {rnd}")
```

min() - Returns the smallest of a sequence of numbers.

```
67 # Minimum (min()) - Returns the smallest of a sequence of numbers
68 # This gives the result of 3, which is the smallest item of this sequence
69 my_list = [3, 4, 5]
70 mn = min(my_list)
71 print(f"Min of 3, 4 and 5: {mn}")
```

max() - Returns the largest of a sequence of numbers.

```
my_list = [3, 4, 5]
mx = max(my_list)
print(f"Max of 3, 4 and 5: {mx}")
```

statistics.mean() - Returns the average of a sequence of numbers.

```
# statistics.mean() - Returns the average of a sequence of numbers
my_list = [3, 4, 5]
av = statistics.mean(my_list)
print(f"Average of 3, 4 and 5: {av}")
```

math.pi - Mathematical constant pi (approximately 3.14159)

```
print(f"Value of pi in Python: {math.pi}")
```

math.sqrt() - Calculates the square root of a number.

```
sq = math.sqrt(4)
print(f"Square root of 4: {sq}")
```

Exponentiation (**) - Raises a number to a power. Same as math.pow().

```
pw = 4**10 # Same as math.pow(4, 10)
print(f"4 to the power of 10: {pw}")
```

abs() - Returns the non-negative version of a number.

```
absolute_value = abs(-5)
print(f"Absolute value of -5: {absolute_value}")
```

random.randint() - Generates a random integer within a specified range (inclusive).

```
ra = random.randint(1, 6)
print(f"Random integer between 1 and 6 inclusive: {ra}")
```

Example run:

```
----- Fun with Math -----
45 modulus 2: 1
10 is an even number.
Integer division 65 // 9: 7
math.floor() rounds 65 / 9 down: 7
math.ceil() rounds 65 / 9 up: 8
Rounded to 2 decimal places: 6.67
Min of 3, 4 and 5: 3
Max of 3, 4 and 5: 5
Average of 3, 4 and 5: 4
Value of pi in Python: 3.141592653589793
Square root of 4: 2.0
4 to the power of 10: 1048576
Absolute value of -5: 5
Random integer between 1 and 6 inclusive: 1
```

Assignment 1: Susie's Square Calculator

Let's calculate the area and perimeter of a square.

1. Create a Python program named **square_calculator.py**

You can pseudocode your code by using a TODO list as shown below. This allows you to plan and start commenting your code.

2. Copy and paste the following to start your program.

```
"""
    Name: square_calculator.py
    Author: William A Loring
    Created: 07/08/22
    Purpose: Calculate the area and perimeter of a square
"""

# TODO: Print a creative title for our program

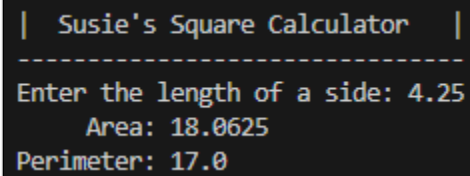
# TODO: Get side and length from user, cast to float

# TODO: Calculate area

# TODO: Calculate perimeter

# TODO: Display results
```

Example run:



```
-----
|  Susie's Square Calculator  |
-----
Enter the length of a side: 4.25
      Area: 18.0625
Perimeter: 17.0
```

Assignment Submission

1. Use pseudocode or TODO.
2. Comment your code to show evidence of understanding.
3. Attach the program files.
4. Attach screenshots showing the successful operation of the program.
5. Submit in Blackboard.