CS65: Introduction to Computer Science

Displaying Texts
Comments
Variables
Assignment Statements
Expressions
Basic input/output

Md Alimoor Reza
Assistant Professor of Computer Science

Announcements

Code examples available via blackboard

Lab #1 out today

Due date (section#1 + section#2): before next class on Tuesday, September 16th

Due date (section#3): before next class on Wednesday, September 17th

Read Chapter 2: Python for Everyone – Variables, expressions, and statements

https://www.py4e.com/html3/02-variables

In a word, how are you doing?

How did the installation of Thonny go on your personal computer?

Announcements

I don’t have expectations of experience with programming

- If things start to go too fast, *let me know*
- If a percentage of students start asking for extensions for a lab, that gets my attention

Quick Review

Recap: Introduction to Computer Science

Study of Computers?

Programming?

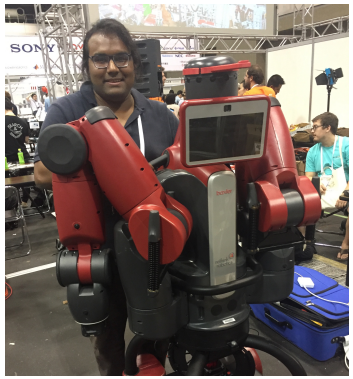
Computer science is the study of **Algorithms**

Algorithm: A step-by-step procedure for producing a solution to a given problem

Recap: Introduction to Computer Science

- Algorithm:
  - step-by-step instructions to be executed by the machine
  - more realistic algorithm in robotics (from my research)

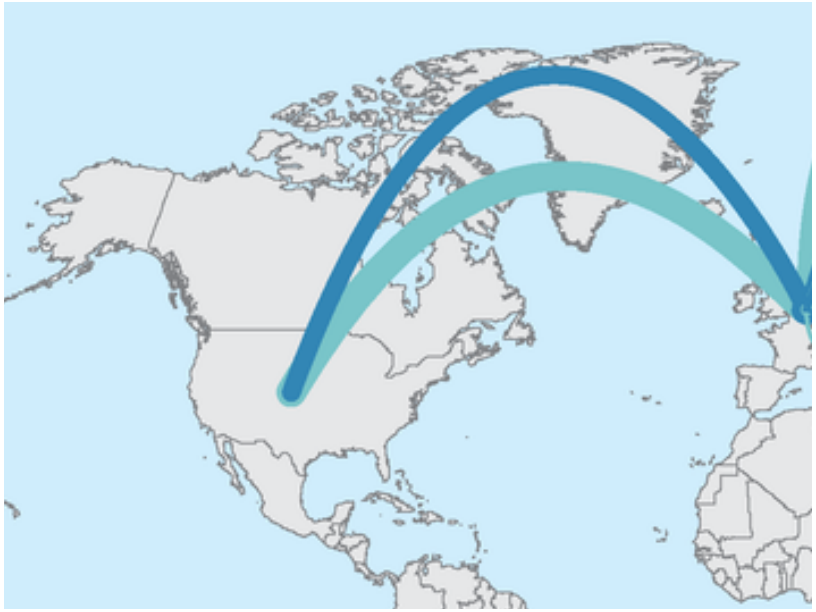
Baxter robot with a goal of fetching an specific object ie, Pringles from the shelf



Me with Baxter

Recap: Algorithm Exercise

- Step-by-step instructions to be executed by the machine
- Describe the process of making a trip from USA to Europe?
  - Person next to you is your partner
  - Write down the steps



Why Program?

Why is programming such a valuable skill?

Computers are tools that can be programmed to perform many functions.

Implement algorithms to solve problems or calculate answers to problems, listen to music, give recommendations, display images, play videos, …

Python

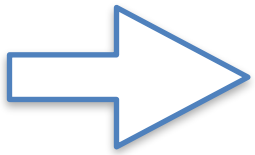
- Python is a programming language

- Purpose is to convert
  - what a programmer writes —> machine executable instructions
- Programmer writes python source code following specific syntax
- There is an interpreter (another program which executes computer code)

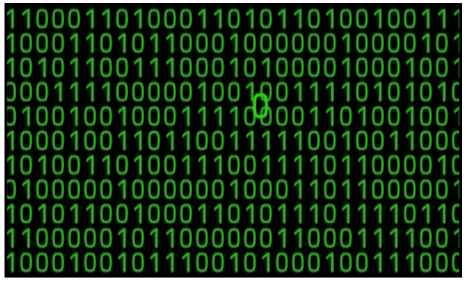
```
n = 5
string = "Hello!"

print(string * n)
```

Python code



Interpreter



Machine readable instructions

Pseudocode

Pseudocode is an informal language that has no syntax rule

- Not meant to be compiled or executed
- Used to create model programs
  - No need to worry about syntax errors, can focus on the program's design.
  - Can be translated directly into actual code in any programming language

I encourage you to always start with pseudocode before you build a program.

Recap: Integrated Development Environment

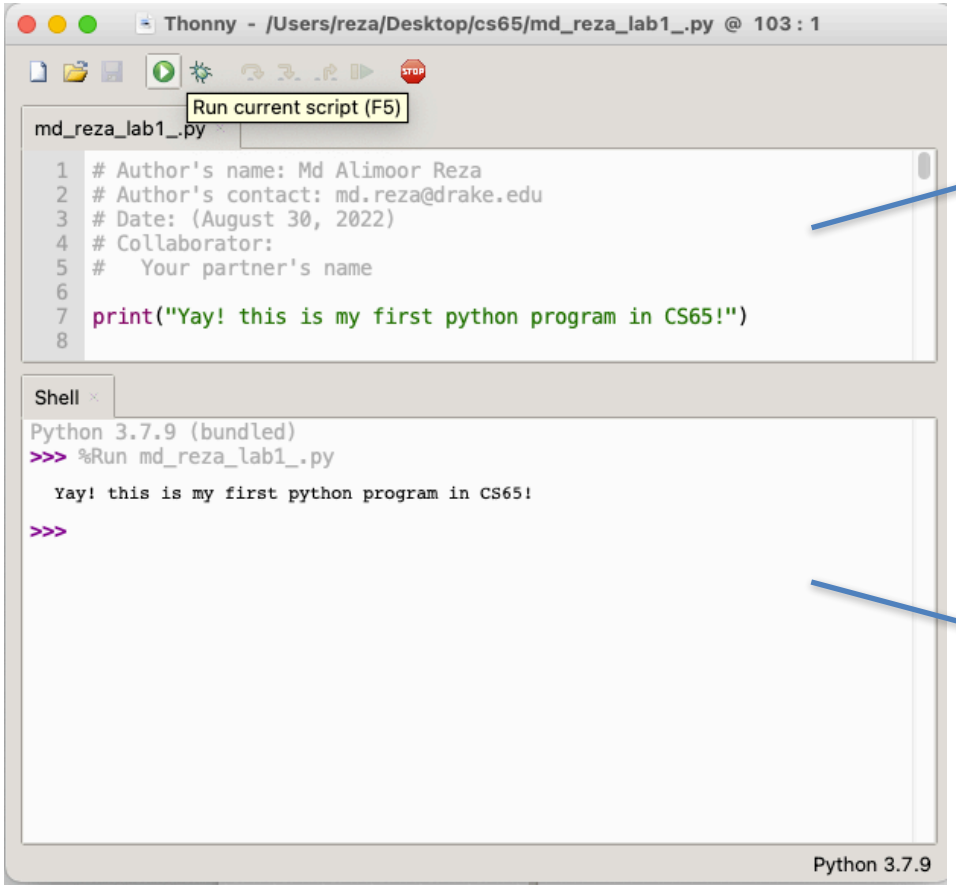
- Integrated Development Environment (IDE) is tool or software system that programmers use to create, run, and test new programs
  - text editor
    - writing python code
  - compiler/interpreter
    - for translating the code into machine understandable instructions
  - executable environment
    - for showing the result of the program

- We will be using the Python programming language along with an IDE for creating Python programs

- Thonny as an IDE
  - very user friendly tool
  - freely available online

Recap: Integrated Development Environment (IDE)

- Demo by Reza in his computer



Questions?

New Material

Roadmap

- Displaying a text
- Comments in Python
- Variables
- Assignment statements
- Expressions
- Receiving input from user
  - Counterpart of showing an user some outputs

Print Statement

The `print` statement is a Python line of code that will output a value when the program is run

The value output will be whatever is between the parentheses ()

hello_world.py

```
1  # name: Alimoor Reza
2  # date: September 16th, 2025
3  # description: my first program in Python programming language
4
5  print("Hello Qingdao University!")
```

Print Statement

The `print` statement is a Python line of code that will output a value when the program is run

The value output will be whatever is between the parentheses ()

hello_world.py

```
1  # name: Alimoor Reza
2  # date: September 16th, 2025
3  # description: my first program in Python programming language
4
5  print("Hello Qingdao University!")
```

In Python, any sequence of characters between double-quotes (“) or single-quotes (‘) is known as a string

String examples:

- **“Hello World”**
- **‘123 ABC’**
- **“Greetings, What is your name?”**

Roadmap

- Displaying a text
- Comments in Python
- Variables
- Assignment statements
- Expressions
- Receiving input from user
  - Counterpart of showing an user some outputs

Comments

Comments are notes of explanation that document lines or sections of a program

Comments are part of the program, but the Python *interpreter* ignores them

They are intended for people who may be reading the source code

- Like your professor
- Or teammates

Python comments are denoted with a #

Comments Example

comments.py

comments.py

```
1  # Alimoor Reza
2  # you should always put your name in the comments at the top of your code
3
4  print('this is a string') # comments can go after the lines of code, too
5  print("This, too, is a string")
```

More Comments

Text that is surrounded by triple quotes (either ‘ or “) are also comments

```
# This is a comment

'''This is also a comment'''
""" Yes,
    this
    is
    also
    a
    comment """
```

Comments

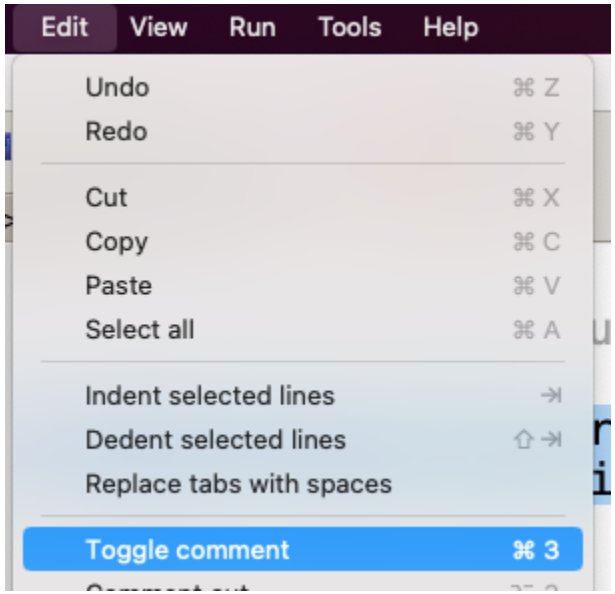
- Ignored by the Python interpreter
  - Won't see any output in the shell environment or any error message

- Helpful for people who may be reading the source code
  - Peer/partner
  - Grader
  - Professor

```
10 """
11  Author's name: Md Alimoor Reza
12  Author's contact: md.reza@drake.edu
13  Date: (September 1st, 2021)
14  Collaborator:
15    Your partner's name
16
17 #print("Yay! this is my first python program in CS65!")"""
18
```

Pro Tip

In Thonny, you can select a chunk of code, and comment all of the lines out by Control (or Command) 3



Demo

Roadmap

- Displaying a text
- Comments in Python
- Variables
- Assignment statements
- Expressions
- Receiving input from user
  - Counterpart of showing an user some outputs

Variables

A **variable** is a *name* that represents a value stored in the computer's memory

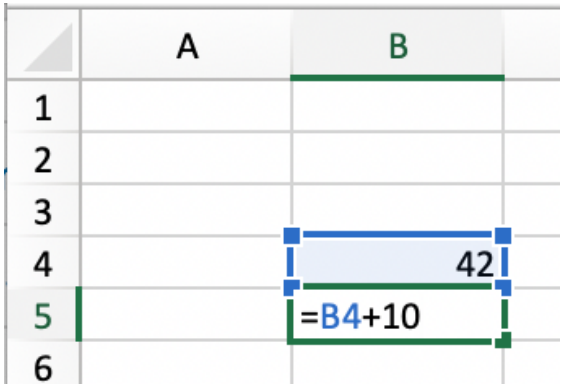
Programmers use variables to hold values that might change

They're kind of like cells in a spreadsheet

A **variable** is a *name* that represents a value stored in the computer's memory

Programmers use variables to hold values that might change

They're kind of like cells in a spreadsheet



Cell with name B4 has the value 42, and we can use that value in other computations by referring to its name.

variables.py

```
1  # Alimoor Reza
2  # syntax for variables
3
4  B4 = 42
5  B5 = B4 + 10
6  print(B4)
7  print(B5)
```

What will be output when this code is run?

A.

```
42
52
```

B.

```
B4
B5
```

C.

```
B4
B4 + 10
```

D.

None of these

Variable and assignment operator

- Need to use assignment operator (**=**) to store a value
- Location of assignment on the left
- Single value or some calculated value on the right
- **variable_name** = value

```
33  time_sec = 60
34  temp_degree = 27
35
36  mile_to_kilometer = 1.609
37  price_in_dollars = 1500.89
```

Numbers

```
first_name = "Md Alimoor"
last_name = "Reza"
```

Textual data

Outputting a variable's value

Don't miss that the print statement will output a variable's **value** if the variable is between the parenthesis

```
value = 34
print(value)
```

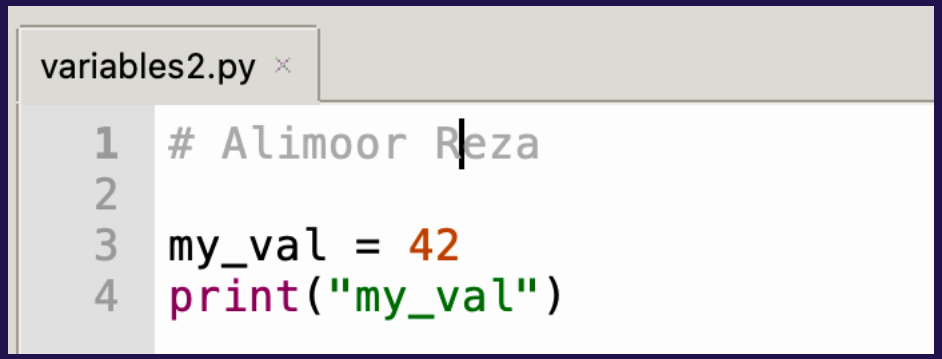
Will output

```
34
```

A tricky question

What will be output when the following code is executed?

variables2.py



Printing text and variables together

We've seen printing out text by itself and variables by themselves. What if I want both in the same print statement?

You can pass multiple things to `print()` using *commas* between them.

Printing text and variables together

We've seen printing out text by itself and variables by themselves. What if I want both in the same print statement?

You can pass multiple things to **`print()`** using *commas* between them.

variables3.py

```
1  # author: Alimoor Reza
2  # description: introduction to variables
3
4  number_of_eggs = 12
5  print("There are ", number_of_eggs, " in the basket.")
6
```

Printing text and variables together

We've seen printing out text by itself and variables by themselves. What if I want both in the same print statement?

You can pass multiple things to **`print()`** using *commas* between them.

variables3.py

```
1  # author: Alimoor Reza
2  # description: introduction to variables
3
4  number_of_eggs = 12
5  print("There are ", number_of_eggs, " in the basket.")
6
```

Shell

```
There are  12  in the basket.
>>> %Run variables3.py
There are  12  in the basket.
>>>
```

Demo

Variable Naming Rules

You can give variables (almost) any name you want

```
x = 42
y = x + 10
print(y)
```

```
52
```

Variable Naming Rules

You can give variables (almost) any name you want

```
x = 42
y = x + 10
print(y)
```

```
52
```

But… you should give them descriptive names for the sake of humans reading the code

```
num_students = 42
num_adults = 10
total_field_trip_lunches = num_students + num_adults
print(total_field_trip_lunches)
```

```
52
```

Variable Naming Conventions

Variable name should reflect its use

- Don’t use `x` when `total_sale` is more descriptive

Variable names should begin with a lowercase letter

If a variable has more than one word, connect them with an underscore

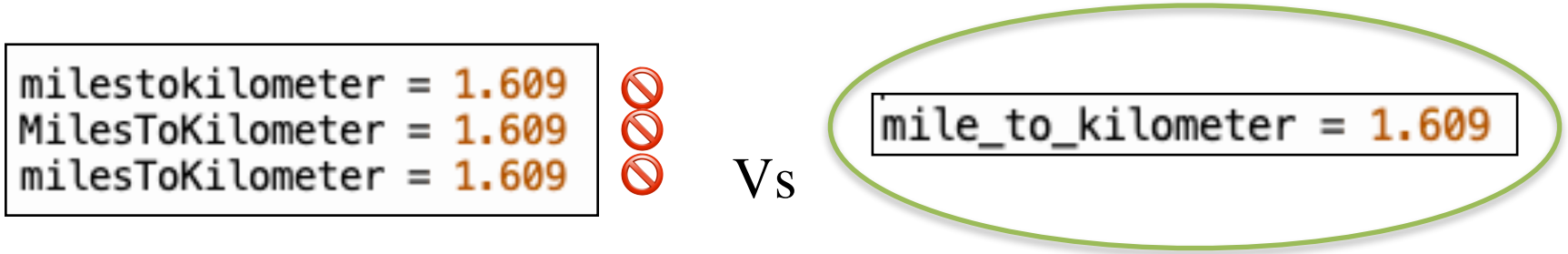
- `total_sale` ✅
- `totalSale` 🚫
- `totalsale` 🚫

Rules for Variable Naming

- Give meaningful variable name to make it easily readable/memorable

- Name should begin with a lowercase letter

- Use underscore to connect multiple words
  - do not use SPACE in between words



Demo

```
35 cash prize = 34
```

Shell

```
>>> %Run Variables2.py
 Traceback (most recent call last):
   File "/Users/000617123/Desktop/CS65code/Day02/Variables2.py", line 35
     cash prize = 34
          ^^^^^
 SyntaxError: invalid syntax
```

```
5 students = 42
6 print(Students)
```

Shell

```
>>> %Run Variables2.py
 Traceback (most recent call last):
   File "/Users/000617123/Desktop/CS65code/Day02/Variables2.py", line 6, in <module>
     print(Students)
 NameError: name 'Students' is not defined
```

Variable Naming Rules

Rules for naming variables in Python:

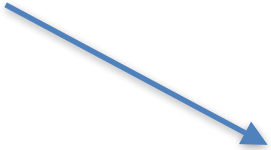
- Variable name cannot be a Python key word

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Demo

Rules for Variable Naming

- Names can only contain letter, numbers, and underscores
- First character must be a letter or an underscore
  - After that, you can use letter/numbers/underscore
- Cannot be a Python keyword



```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

- Cannot contain spaces
- Variable names are case sensitive
  - Uppercase and lowercase name will signify different variable

Exercise

- Which of followings are bad variable names and why?

```
import
First name
?my_variable
555_battery
X
FirstName
global
while
```

Rules for Variable Naming

- Names can only contain letter, numbers, and underscores
- First character must be a letter or an underscore
  - Then use letter/numbers/underscore
- Cannot be a Python keyword

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

- Cannot contain spaces
- Variable names are case sensitive
  - Uppercase and lowercase name will signify different variable

Roadmap

- Displaying a text
- Comments in Python
- Variables
- Assignment statements
- Expressions
- Receiving input from user
  - Counterpart of showing an user some outputs

Assignment Statements

Code with a = is called an *assignment statement*

You use an *assignment statement* to create a variable and make it reference a piece of data

Examples:

Assignment Statements

Code with a = is called an *assignment statement*

You use an *assignment statement* to create a variable and make it reference a piece of data

Examples:

```
age = 25
hourly_rate = 15.5
greeting = "Hello everyone"
```

Assignment Statements have an order

Whatever is on the right side of = will be evaluated and saved to the variable named on the left side.

```
students = 42
```

Assignment Statements have an order

Whatever is on the right side of = will be evaluated and saved to the variable named on the left side.

```
students = 42
```

You can't write it the other way around.

```
5 42 = students
6
```

Shell

```
>>> %Run Variables2.py
Traceback (most recent call last):
  File "/Users/000617123/Desktop/CS65code/Day02/Variables2.py", line 5
    42 = students
    ^^
SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?
```

Roadmap

- Displaying a text
- Comments in Python
- Variables
- Assignment statements
- Expressions
- Receiving input from user
  - Counterpart of showing an user some outputs

Expressions

An *expression* is something evaluated by the Python interpreter

```
num_students = 42
num_adults = 10
total_field_trip_lunches = num_students + num_adults
print("Total field trip lunches:", total_field_trip_lunches)
```

Expression

A fragment of Python code that calculates a new value called an expression

For example, you can convert miles into meters using the following expression:

```
num_of_miles = 10
miles_to_kilometer = 1.609
```

```
num_of_meter = num_of_miles*miles_to_kilometer*1000
```

Expressions

An *expression* is something evaluated by the Python interpreter

expression.py

```
expression.py
1  # author: Alimoor Reza
2  # description: introduction to expression
3
4  num_of_students = 42
5  num_of_adults = 10
6  total_field_trip_lunches = num_of_students + num_of_adults
7  print("Total field trip lunches ", total_field_trip_lunches)
```

When `num_of_students + num_of_adults` is evaluated, it results in an answer, so it is an *expression*

Expressions

An *expression* is something evaluated by the Python interpreter

expression.py

```
# author: Alimoor Reza
# description: introduction to expression

num_of_students = 42
num_of_adults = 10
total_field_trip_lunches = num_of_students + num_of_adults
print("Total field trip lunches ", total_field_trip_lunches)
```

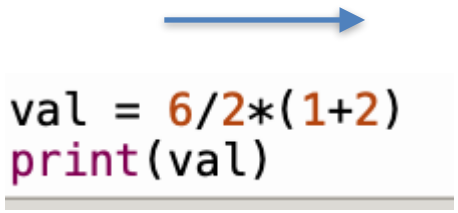
When **`num_of_students + num_of_adults`** is evaluated, it results in an answer, so it is an *expression*

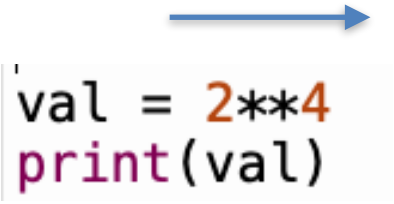
Expressions include operators, like +, and operands like `num_of_students` and `num_of_adults`

Arithmetic Operators

Python supports all of the arithmetic operators you know and love:

- ( ) parentheses
- ** exponents
- * multiplication
- / division
- + addition
- - subtraction



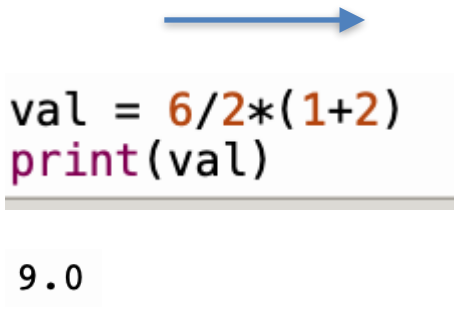


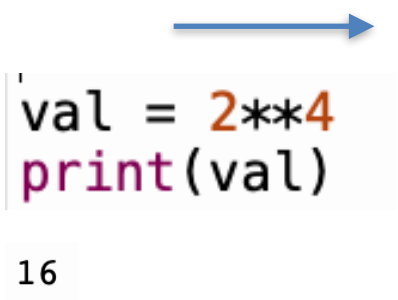
PEMDAS order of operations

Arithmetic Operators

Python supports all of the arithmetic operators you know and love:

- ( ) parentheses
- ** exponents
- * multiplication
- / division
- + addition
- - subtraction





PEMDAS order of operations

Arithmetic Operators

Python supports all of the arithmetic operators you know and love:

- ( ) parentheses
- ** exponents
- * multiplication
- / division
- + addition
- - subtraction

```
val = 6/2*(1+2)+5
print(val)
```

PEMDAS order of operations

Let’s try it

Exercise

1. Open Thonny
2. Create a new File
3. Save it as “Day02Exercise.py”

You may want to consider putting this into its own folder inside your CS65 folder

1. Create a program that uses a variable and outputs a computation

   Examples include:

   - Amount of sugar, flower, butter for cookies
   - Body Mass Index (BMI) calculator
   - English to metric conversions
   - Yield on investments
   - Weight on moon formula

Feel free to work with others!

Goals:

- Get comfortable with Thonny
- Make mistakes (and fix them)
- Experiment
- Have fun!

Here are some example exercises!

- Can you compute the area of a rectangle?
  - Length of the two sides will be given in variables

- Can you compute the area of a circle?
  - Radius of the circle is given
  - Value of Pi is 3.14159

Roadmap

- Displaying a text
- Comments in Python
- Variables
- Assignment statements
- Expressions
- Receiving input from user
  - Opposite of displaying an user some outputs

User Input

Python provides a function called `input` that gets input from the keyboard.

When this function is called, the program *stops and waits* for the user to type something.

When the user presses Return or Enter, the program resumes and input returns what the user typed as a string

User Input

Python provides a function called `input` that gets input from the keyboard.

When this function is called, the program *stops and waits* for the user to type something.

When the user presses Return or Enter, the program resumes and input returns what the user typed as a string

input_and_output.py

input_and_output.py

```
1  # author: Alimoor Reza
2  # description: receiving input from the user and displaying output
3
4  first_name = input("Please enter your first name")
5  last_name = input("Please enter your last name")
6  print("Hello ", first_name, " ", last_name)
```

User Input – Strings

By default, the value of variable that results from an input statement is a String.

Strings are very useful, but they are limited in that they only hold a sequence of characters.

- As such, you cannot perform mathematical operations on strings or string variables.
- if I want the result of the input function to give me a value that I can use in a calculation, I need to convert the string to a numeric format.

Numerical Formats

Python has two different forms of numbers:

- `int` – short for integer. These are whole numbers that are not a fraction. Examples are 0, 4, 24, -12
- `float` – floats are numbers with a decimal point. float is short for floating-point numbers. Examples include 1.3, -34.5, 5.0

This will cause an error… why?

input_error.py

```
1  # author: Alimoor Reza
2  # description: there is a mistake in the program
3
4  num_cookies = input("How many cookies you would like to make?")
5  ratio = num_cookies/48
6  print(ratio)
7
```

input_rrror.py

This will cause an error… why?

input_error.py

```
1  # author: Alimoor Reza
2  # description: there is a mistake in the program
3
4  num_cookies = input("How many cookies you would like to make?")
5  ratio = num_cookies/48
6  print(ratio)
7
```

Shell

```
Traceback (most recent call last):
  File "/Users/reza/Class_and_Research/drake_university/drake_teaching/qingdao_university/cs65_fall25/cs65_codes/lecture02/input_error.py", line 5, in <module>
    ratio = num_cookies/48
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

input_rrror.py

Solution: Use integer or float types for calculations

If you expect the user to type an `integer` or `float`, you can try to convert the return value to int using the `int()` or `float()` function:

Use integer or float types for calculations

If you expect the user to type an `integer` or `float`, you can try to convert the return value to int using the `int()` or `float()` function:

input_error_fixed.py

```
1  # author: Alimoor Reza
2  # description: user input with an integer
3
4  num_cookies = int(input("How many cookies you would like to make?"))
5  ratio = num_cookies/48
6  sugar = ratio*1.5
7  print("you will need ", sugar, " cups of sugar")
8
9
```

Shell

```
>>> %Run input_error_fixed.py

  How many cookies you would like to make?4
  you will need  0.125  cups of sugar
```

Summary: Getting Input from Users

- Built-in function in Python ***input("….")***
  - Step 1: displays the prompt to the user
  - Step 2: waits for user to type in something
  - Step 3: returns the typed content when user hits enter
  - Step 4: this value is stored if assigned to a variable

```
rect_a = input("enter the length of rectangle side a: ")
print(rect_a)
```

Demo

What to work on

Lab #1 due before next class on Tuesday, September 16th (before class)

Textbook Reading

https://www.py4e.com/html3/02-variables

Class summary

- Main takeaway from this lecture:
  - You can write your Python code in Thonny (IDE)
  - Comments in Python
  - Variables and expressions
  - Receiving input from user
    - Opposite of showing some outputs

Lab Activity

- Additionally— all the materials will be available (as links) in the schedule

- To do: Finish the instructions in Lab1

CS 65: Introduction to Computer Science (Fall 25)
Instructor: **Dr. Md Alimoor Reza**
Assistant Professor of Computer Science
Department of Mathematics and Computer Science
Drake University

Classroom: Boyi#508@Qingdao University
Meeting Time : Tuesday 14:00 pm - 15:50 pm (Week#1-Week#9), Thursday 10:10 am - 12:00 pm (Week#1-Week#17)
Office Hours: Tuesday: 08:00 am-9:50 am and and Thursday: 2:00 pm - 3:50 pm or by appointment

MOOC Portal: Chaoxing Fanya (Xue Xi Tong)

Section#1 Schedule
A tentative schedule is provided below (content may be adjusted as we progress).

Date	Topic	Reading	Items due
week 1 (Tue: Sep 09)	Introduction to Computer Science Lecture 1 slide **Lab 0 (Windows user)** **Lab 0 (Mac user)**	Reading: A Byte of Python (Why Python & its advantage)	
week 1 (Thu: Sep 11)	Displaying text Assignment statements Variables, expression Lecture 2 slide **Lab 1 (released on Sep 11)**	Reading: Chapter 2: Python for Everyone – Variables, expressions, and statements	
week 2 (Tue: Sep 16)	Functions (part 1) Lecture 3 slide	Reading:	Lab 1 (due by September 16th)

Summary

- To do: Finish the instructions in Lab1
- To do: Finish the provided book chapter reading

CS 65: Introduction to Computer Science (Fall 25)
Instructor: **Dr. Md Alimoor Reza**
Assistant Professor of Computer Science
Department of Mathematics and Computer Science
Drake University

Classroom: Boyi#508@Qingdao University
Meeting Time : Tuesday 14:00 pm - 15:50 pm (Week#1-Week#9), Thursday 10:10 am - 12:00 pm (Week#1-Week#17)
Office Hours: Tuesday: 08:00 am-9:50 am and and Thursday: 2:00 pm - 3:50 pm or by appointment

MOOC Portal: Chaoxing Fanya (Xue Xi Tong)

Section#1 Schedule
A tentative schedule is provided below (content may be adjusted as we progress).

Date	Topic	Reading	Items due
week 1 (Tue: Sep 09)	Introduction to Computer Science Lecture 1 slide **Lab 0 (Windows user)** **Lab 0 (Mac user)**	Reading: A Byte of Python (Why Python & its advantage)	
week 1 (Thu: Sep 11)	Displaying text Assignment statements Variables, expression Lecture 2 slide **Lab 1 (released on Sep 11)**	Reading: Chapter 2: Python for Everyone – Variables, expressions, and statements	
week 2 (Tue: Sep 16)	Functions (part 1) Lecture 3 slide	Reading:	Lab 1 (due by September 16th)