

# Programming: Writing a Program

---

**CPSC 217: Introduction to Computer Science for Multidisciplinary Studies I**  
**Winter 2023**

Jonathan Hudson, Ph.D.  
Instructor  
Department of Computer Science  
University of Calgary

*January 9, 2023*

*Copyright © 2023*



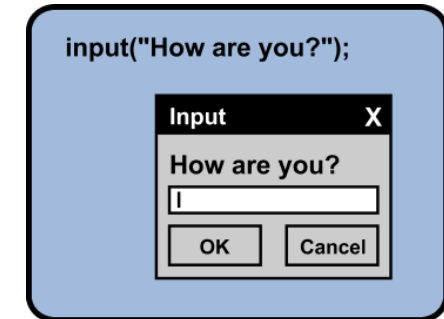
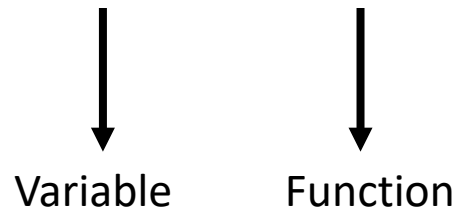
**UNIVERSITY OF  
CALGARY**

# Input

---



- A built-in function to get an input from the user.
- `sName = input("Please enter your name: ")`



- The obtained value will be stored in the variable in **string** form.
- You need to convert the value type if necessary.

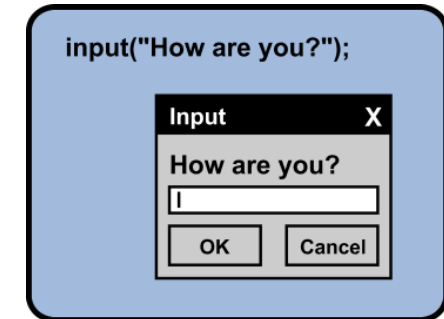
# Input

---

```
#Changing type (this is always a string)  
sInput = input("Enter a number:")
```

```
#Change to float  
fInput = float(sInput)  
print(type(fInput))
```

```
#Change to integer  
iInput = int(sInput)  
print(type(iInput))
```



# Meaningful sentences

- Provide meaningful sentences to communicate questions and results to the user.

```
sFahrenheit = input("Please enter the degree in fahrenheit : ")
```

```
#Program
```

```
print ("The degree in celsius is: ", fCelsius)
```



```
sFahrenheit = input("degree")  
print (fCelsius)
```



The programmer knows they need to do to complete the above program.  
The user using program knows what to enter and what is output.

# Errors

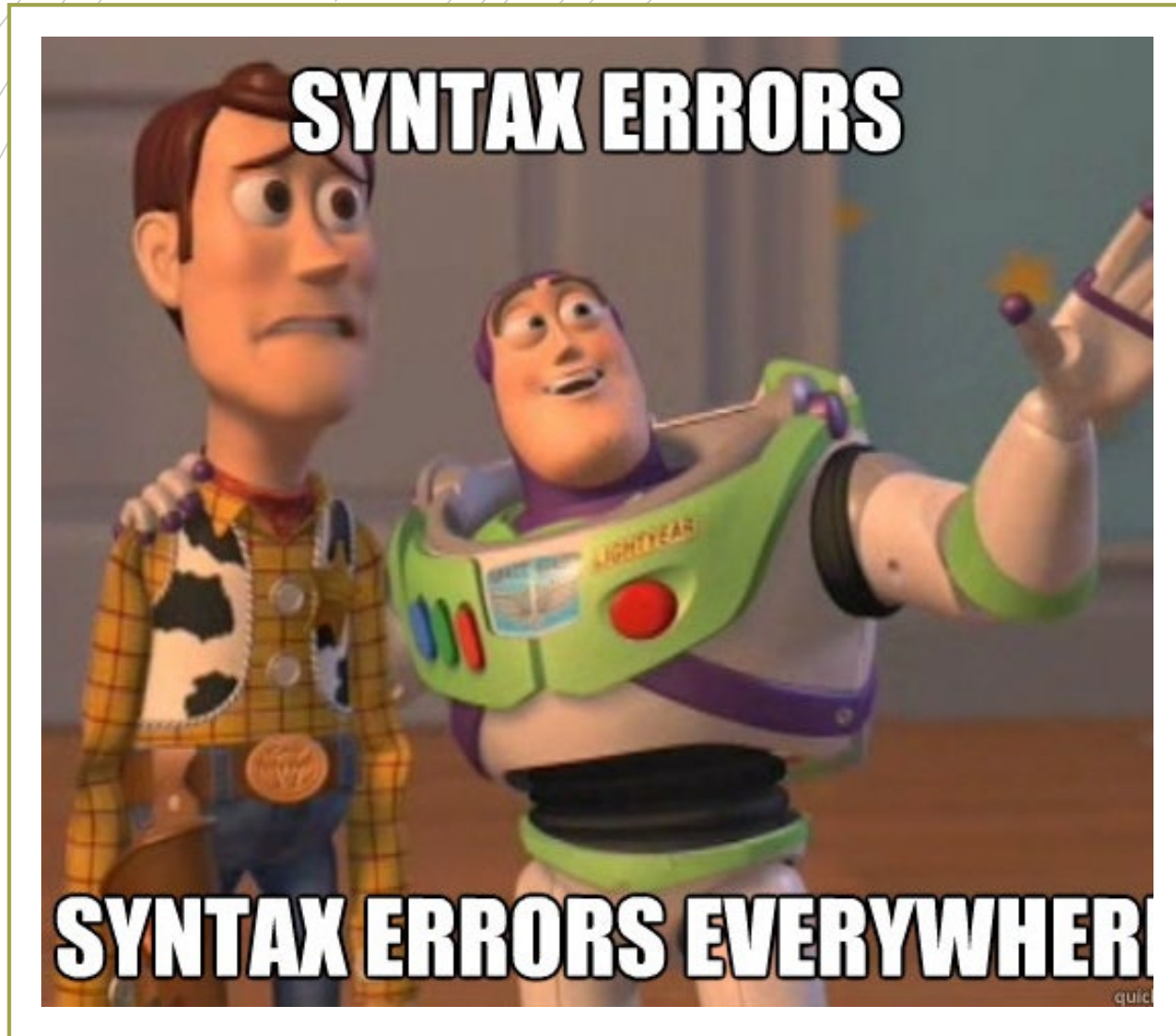
---



# Types of Errors

---

- Three categories
  1. Syntax
  2. Runtime
  3. Logic



## Syntax errors

- **Syntax** refers to the structure of a program and the rules about that structure.
- Easiest to find (**your program will not begin to execute (after compiling)**)
- Displays an error message and stops.
- The compiler/interpreter finds syntax errors
- Example:

```
name = "Jim  
print(77
```

# Runtime errors

---



**Does not appear until  
the program is run  
(compiling is complete)  
(harder to find)**



**Also called exceptions**



**Example:**

```
callMe = "Maybe"  
print(callme)
```



# Semantic/Logic Errors

Program will run successfully

It does not produce the correct answer:

- It does what you told it to do
- **The hardest to find!**
- Program runs to completion, but result is wrong.

Result of a wrong algorithm

Identifying is tricky!

Example:

- $x = 3$
- $y = 4$
- $\text{average} = x + y / 2$
- `print(average)`

# Remember

---

- Getting errors in your code does not make you a bad programmer.
- Errorless code is the goal
- You need to debug your code
- Add comments to your code
- Grading is generally setup to give you credit for the parts that are complete and correct (as long as they run and can be verified)

# Explore First Program

---

## What Does this Program Do?

```
x = float(input())  
y = (x - 32) * 5/9  
print(y)
```

- What's wrong with this program?

# Fahrenheit to Celsius

---

- Examples on course website/python notebook

# Comments

- Notes and explanations in natural language
- Intended only for the human reader → Completely ignored by the interpreter
- # token starts a comment
  
- **Commenting is necessary!**

```
#Calculating Fahrenheit to Celsius
OFFSET = 32
RATIO_CHANGE = 5/9

#Obtaining Fahrenheit degree
sFahrenheit = input ("Please enter the degree in Fahrenheit :")

#Converting the input to float
fFahrenheit = float(sFahrenheit)

#Calculating the Celsius
fCelsius = (fFahrenheit - OFFSET) * RATIO_CHANGE

#Printing the result
print("The degree in Celsius is: ", fCelsius)
```

# Importing

---

# Import

---

- **import <Module Name>**
- Import the packages that you need only!
- Example:  

```
import math  
import turtle
```



# Math Functions

---

- Many additional math functions are available
- Located in the math library
  - Import the math library
    - Precede the name of the function with **math**.

- Examples:

```
import math
```

```
math.sqrt(x)
```

```
math.floor(x)
```

```
math.ceil(x)
```

```
math.cos(x)
```

# Area of a circle

---

# Data Types

---

# Data Types

---

Some operations are only well defined for certain types

- $1 + 2$
- "Hello" + " World"
- $1 + \text{"Hello"}$
- $2 + \text{"4"}$
- $1 / 3$
- $2.0 / 4$
- "Hello"\*3

# Data Types

---

Some operations are only well defined for certain types

- $1 + 2$  → GOOD
- "Hello" + " World" → GOOD
- $1 + \text{"Hello"}$  → TYPE ERROR (int + string)
- $2 + \text{"4"}$  → TYPE ERROR (int + string)
- $1 / 3$  → GOOD
- $2.0 / 4$  → GOOD
- "Hello"\*3 → GOOD

# Type Conversions (Casting)

---

Python permits you to convert from one type to another

`"1.0" / "3.0"` → Makes no sense to python

`float("1.0") / float("3.0")` → But this does

`float("asdf")` → However this doesn't

Other type conversions: int, bool, str

# Type Conversions (Casting)

---

Python permits you to convert from one type to another

<code>int("1")</code>	→ 1 (string to int)
<code>str(1)</code>	→ "1" (int to string)
<code>float("1.0")</code>	→ 1.0 (string to float)
<code>bool("True")</code>	→ True (string to boolean)

# Formatting

---



# Formatting

---

Sometimes print doesn't display things the way we would like

```
print(1 / 3.0) gives 0.3333333333333333
```

What if we want 0.33?

# Formatting

---

Basic python format method can be used to format floats

Parts

A string

**“pi is %.2f”**

Format **float** to **2** decimals

**%.2f**

Format symbol for strings

**%**

The value(s) to format

**(math.pi)**

Example:

```
import math
```

```
print(“pi is %.2f when rounded to 2 decimal places” % (math.pi))
```

# Formatting

---

## types

f – float

g – scientific notation

s – string

d – integer

[width].[precision][type]

width – total characters in final result ("" is default ) (add 0 in front to pad 0's)

precision – how many decimal points

Ex. 05.3f

float, pad with 0s if shorter than 5 to get width of 5, but only after showing precision of 3

# Turtle - Drawing

---

# Alex the turtle

---

- Example in course website/python notebook

# Onward to ... information and data.

---

Jonathan Hudson  
[jwhudson@ucalgary.ca](mailto:jwhudson@ucalgary.ca)  
<https://pages.cpsc.ucalgary.ca/~jwhudson/>



UNIVERSITY OF  
CALGARY