Code in Place 2023

Stanford CS106A

Section - Week 3

Programming with the Python Console

David Tsai, Code in Place

Today's Agenda

1. Check-In

How are we all doing?

2. Concepts Review

Console Programming,
Expressions, Control Flow

3. Practice Problem #1

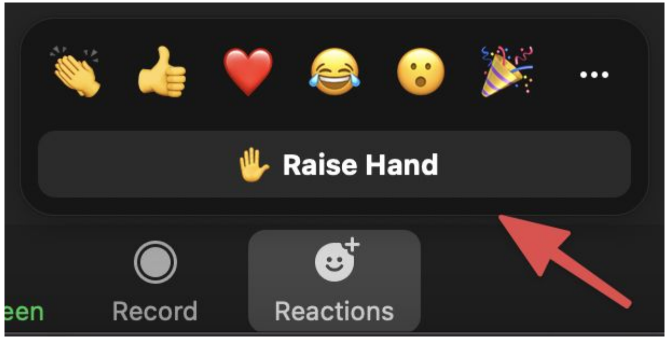
"Mars Weight"

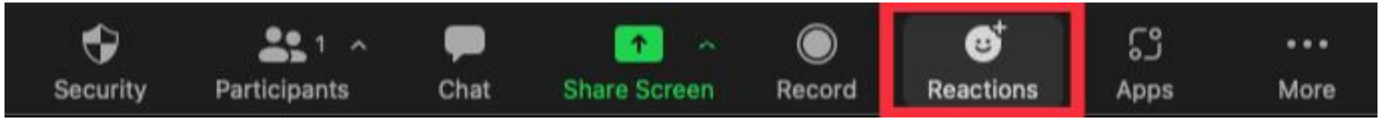
4. Practice Problem #2

"Planetary Weight"

Zoom Reactions

- 👍 **Thumbs Up:** If you understand.
- ✋ **Raise Hand:** If you have a question (or just speak in the mic).





Before We Start

How are you all doing?
Hopefully your second week of CIP went well!

- If you could have Karel know a 5th default command, what would it be?
- Is there anything blocking you from completing Assignment 1 or 2?
- Any major questions?

Concepts Review

Intro to Console Programming

Welcome to real-life Python world!
No longer restricted to Karel's world with only 4 commands!

Basic commands for today's Exercise:

`print()`

- Prints text or value to console.
- Example:
  - `print("Hello, world!")`
  - `print(42)`

`input()`

- Requests the user to type in an input, which can be stored as a string.
- Example:
  - `user_height = input("Please enter your height: ")`
  - `print(user_height)`

Variables

- A **variable** is a place to store information in a program.

- You create a new variable by **assigning** a **value** to a **name**.
    - You use the equal sign (=) to assign a value to a variable.
    - `variable_name = value`

```
x = 10                  # Assign the value "10" to the variable named "x"
x = 5                   # The value of "x" is now 5
x = 5 + 7               # The value of "x" is now 12
```

- Variable assignment (=) is **NOT THE SAME** as "equals" in math.

```
total = 5               # Assign the value of "5" to the variable named "total"
total = total + 1       # The value of "total" is now 6
```

Data Types

- Each variable needs to know what **Type** of information it's carrying.

- **Some Types in Python:**
  - `int`: integer value (no decimal point)
    - `-2, -1, 0, 1, 2, 3, 4`
  - `float`: real number value (has decimal point)
    - `2.0, -0.39, 3.14159`
  - `string`: text characters (surrounded by single/double quotes)
    - `“Hello CIP!”, ‘Hello CIP!’, “10”, ‘10’`
  - `bool`: Boolean logical values (True or False)
    - `True, False`

Type Casting (aka Converting)

- You can **cast** (aka convert) a variable from one Type to another.

- Python has several built-in functions for type casting. Here are a few you might find helpful:

```
x = int(y)            # y is cast to an int
x = float(y)          # y is cast to a float
x = str(y)            # y is cast to a string
```

- Examples:

```
user_input = int("75")     # user_input: 75  [Type: int]

height = float("5.3")      # height: 5.3     [Type: float]

total = str(42.9)          # total: "42.9"   [Type: str]
```

Variables: Naming Convention and Constants

- Variable name must:
  - Start with a letter or an underscore (`_`)
  - Contain only letters, digits, or underscores
  - Cannot be one of the "built-in" Python commands (e.g. `for`)
- Variable names are **case sensitive**
  - `User_height` is not the same as `user_height`
- Python style: Use "snake case" for variable names.
  - **Do:** `user_height`
  - **Don't:** `userheight, userHeight`

- **Constants** are variables that you think should be a fixed value.
  - Python style: Use "snake case" with all capital letters.
  - Examples:
    - `PI = 3.14159`
    - `MINUTES_PER_HOUR = 60`

Be mindful of Types when using `print()`

- `print(`*`argument`*`)`: The argument can be any Type.

```
print(“42”) # string                   print(42.0) # float
print(42)   # int                      print(True) # bool
```

- Different ways of **concatenating** a string:

```
print(“Hello Chris!”)
print(“Hello ” + “Chris!”)             # Note the space after Hello
print(“Hello” + “ ” + “Chris!”)
print(“Hello”, “Chris!”)               # Arguments will separated by a space
```

- You **can’t mix-and-match** Types for the argument.

```
print(“My age is ” + 42)               # Error; you can’t mix a string with an int
print(“My age is ” + str(42))          # This will work; argument is entirely of type string
```

- You can print variables, but be careful of the above rule!

```
student_name = “Chris”                 # Type: string
student_age = 25                       # Type: int
print(“My name is ” + student_name + “ and I am ” + str(student_age))
```

Be mindful of Types when using `input()`

- `input(`*`argument`*`)`: **Will return a result of Type string.**

```
user_weight = input("Enter your weight (kg): ")

# Let's say the user types into the console the number 68 and pressed Enter.
      # user_weight will be assigned the value "68".
      # user_weight will be of Type string!  Even though they typed in a number.

new_weight = user_weight + 5            # Error; can't add a string with an int
new_weight = int(user_weight) + 5       # This will work; adding two ints
```

Expressions & Arithmetic Operators

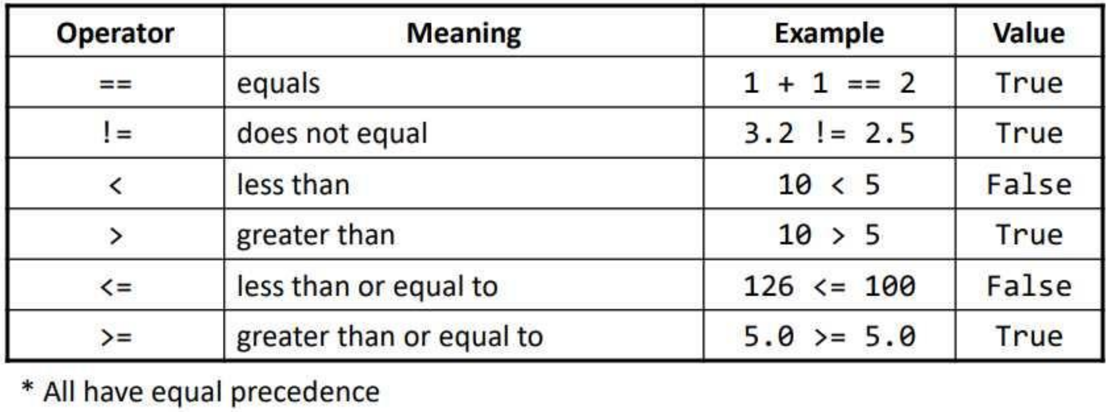
- When assigning variables, any **expression** on the right sight of the equal sign is calculated before being assigned to the variable.
- **Operations on numerical types** (`int` and `float`)
  - `+, -, *, /, //, %, **, -`
- **Precedence of operations** (similar to "PEMDAS" in Algebra):

`()`	"parentheses"	**highest precedence**
`**`	"exponentiation"	↓
`-`	"negation"	
`*, /, //, %`		
`+, -`		**lowest precedence**

- Example:

```
x = (1 + 3 * 5 / 2) * (-3)    # Python will calculate the right-side expression.
x = (1 + 15 / 2) * (-3)
x = (1 + 7.5) * (-3)
x = (8.5) * (-3)
x = -25.5                     # float -25.5 assigned to "x".
```

Comparison Operators



Operator	Meaning	Example	Value
==	equals	`1 + 1 == 2`	`True`
!=	does not equal	`3.2 != 2.5`	`True`
<	less than	`10 < 5`	`False`
>	greater than	`10 > 5`	`True`
<=	less than or equal to	`126 <= 100`	`False`
>=	greater than or equal to	`5.0 >= 5.0`	`True`

* All have equal precedence

WARNING: Notice the difference between variable **assignment** vs **equality** comparison.

- `x = 65        # Assigns the int value 65 to the variable x.`
- `x == 65       # Checks if the value of x is equal to 65.`

Control Flow

- Everything from Karel is still in play!
- `for` loops
- `while` loops
- booleans (`True` or `False`, used in conditional statements)
- `if, else, elif`

```
user_number = int(input(“Enter a number: ))      # Variable assignment
if user_number == 0:                             # Equality comparison
      print(“Your number is 0!”)
elif user_number > 0:
      print(“Your number is positive!”)
else:
      print(“Your number is negative!”)
```

Section Exercise: “Mars Weight Calculator”

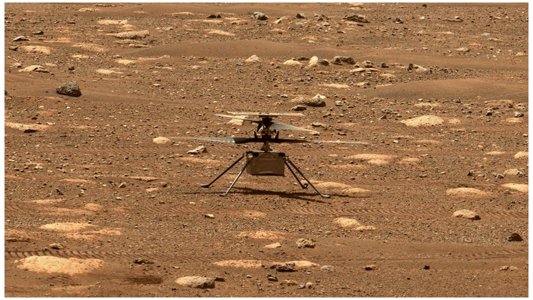
Milestone #1: Ask the user their weight on Earth. Output the equivalent weight on Mars!

Input

```
Enter a weight on Earth: 120
```

Output

```
The equivalent weight on Mars: 45.36
```



Milestone #2: Make the calculator work for <u>any</u> other planet in solar system.