

Code in Place 2024

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



3. Practice Problem #1

"Mars Weight"



4. Practice Problem #2

"Planetary Weight"



Check-In

How is everyone doing?
Hopefully your second week of CIP went well!

- If you could have Karel know a 5th default command, what would it be?

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.
- Creating and assigning a new variable:
 - `variable_name = value or expression`

```
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
```

Variables:

Assignment (=) vs Comparison (==)



Spot the difference



`x = 64`

- Assigns the value 64 to a variable named `x`.
- Creates the variable if it didn't already exist.

`x == 64`

- Checks if a variable named `x` has the value 64.
- Returns either `true` or `false`.
- Used in `if` statements and `while` loops.

Variables: Python Naming Conventions



- 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`
- Use “snake case” for variable names.
 - **Do:** `user_height`
 - **Don't:** `userheight`, `userHeight`



Variables: Constants



- **Constants** are variables that you think should be a fixed value.
 - Constant names use capital SNAKE_CASE.
 - Examples:
 - `PI = 3.14159`
 - `MINUTES_PER_HOUR = 60`
 - `CAPITAL_OF_FRANCE = "Paris"`

Variables: 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]

Combining strings



- Different ways of **concatenating** a string:
 - Using plus sign (+) to combine strings.

```
print("Hello Chris Piech!")  
print("Hello " + "Chris " + "Piech!")  
print("Hello" + " " + "Chris" + " " + "Piech!")
```

- Using comma (,) to combine multiple arguments.
 - Each argument will be separated by a space.

```
print("Hello", "Chris", "Piech!")
```



- There are other ways, such as f-strings! We'll cover in the future!

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
```

- 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 remember 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.**
 - If the result is a number and you want to do calculations with it, remember to cast the result to an `int` or `float`.

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



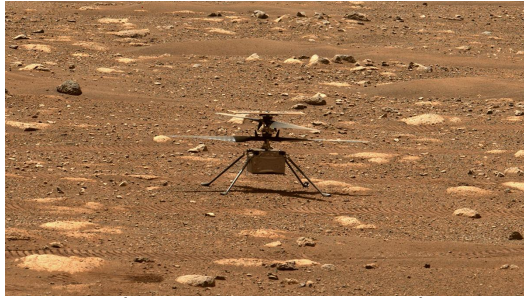
```
new_weight = user_weight + 5
```

```
# Error; can't add a string with an int
```

```
new_weight = int(user_weight) + 5
```

```
# This will work; adding two ints
```

Section Exercise: “Mars Weight Calculator”



(Image source: NASA)

Gravitational constant for Mars compared to Earth's:

- Mars: 37.8%



Milestone #1: Ask the user their weight on Earth. Output the equivalent weight on Mars (rounded to two decimal places)!

Input

Enter a weight on Earth: 120



Output

The equivalent weight on Mars: 45.36

How to Round a Number



- **Number Rounding:**

- `round(float, num_decimals)`

```
x = 3.1415926
rounded_x = round(x, 2)      # Rounds x to 2 decimal places
print(rounded_x)             # Prints: 3.14
```

```
y = 2.71828
print(round(y, 4))           # Prints: 2.7183
```

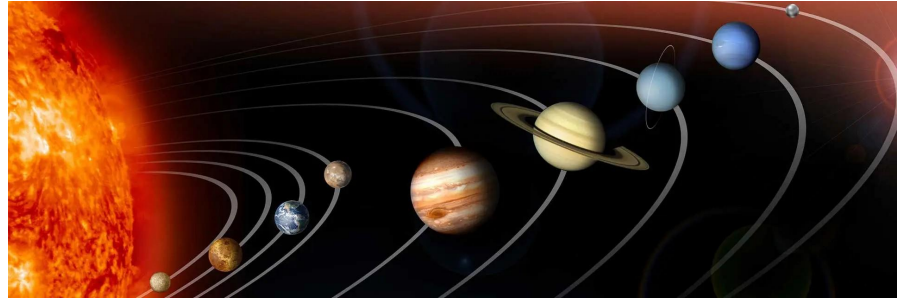

Section Exercise: “Planetary Weight Calculator”



Milestone #2: Make the calculator work for any planet in solar system.

Gravitational constants for each planet compared to Earth's:

- Mercury: 37.6%
- Venus: 88.9%
- Earth: 100.0%
- Mars: 37.8%
- Jupiter: 236.0%
- Saturn: 108.1%
- Uranus: 81.5%
- Neptune: 114.0%



(Image source: <https://science.nasa.gov/solar-system/>)



Input

Enter a weight on Earth: 150
Enter a planet: Jupiter



Output

The equivalent weight on Jupiter: 354.0