

The background is a solid pink color. It is decorated with various hand-drawn geometric shapes in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a black circle in the bottom left, a white triangle in the bottom center, a dashed line in the bottom center, a black circle in the bottom center, a white circle in the bottom right, and a black circle in the bottom right.

# Welcome!

We'll get started shortly ...



# CS 49 Section


Week 5

Surajit A Bose





# Agenda


- Logistics and check-ins
  - Review of lecture concepts
    - Expressions
    - Constants
    - The **random** module and **math** library
  - [Coding Template](#)
  - Section Problem: [Mars Weight](#)
- 

The background is a solid pink color. It is decorated with various white line art elements: in the top left, there are two parallel diagonal lines, a dashed line, a zigzag line, and a plus sign; in the top right, there is a dashed line, a zigzag line, a triangle, and a plus sign; in the middle right, there is a curved line; in the bottom left, there is a thick white brushstroke and a plus sign; in the bottom center, there is a plus sign and a wavy line; in the bottom right, there is a semi-circle with vertical lines inside.

# Logistics



# Section Survey

- Please take the brief, anonymous [Section Survey](https://forms.gle/p4Qjd5235H2359vQ9)!
  - Two required multiple choice questions
  - Two optional open-ended questions
  - Will take about one minute
  - Will help me make section more useful for you
  - Not the same as the course-wide mid-quarter survey
  - Section survey URL: <https://forms.gle/p4Qjd5235H2359vQ9>
  - Deadline: 11a Friday, February 14
- 



# No meeting February 14


+

- Due to the Presidents' Day holiday, there will be no section next Friday
- Section slides **and solutions** for week 6 will be posted early next week
  - CiP solutions may not be available until Friday
- Feel free to reach out via Pronto or Canvas inbox with any questions regarding the week's material or assignments
- Enjoy your Valentine's Day!




# Quick Check-in

Let's quickly go round the room! Pick any one of the following to complete the sentence: As the weeks go by, this class is getting:

- easier, because ...
  - more difficult, because ....
  - more fun, because ...
  - more stressful, because ...
- 



# How to get hold of me / get help+

- The [class forum](#). Feel free not only to ask, but also to answer questions!
  - Surajit's office hours:
    - Fridays 12 noon–1p, directly after section
    - By appointment on [Zoom](#)
  - [Lane's office hours](#)
  - Canvas inbox or Pronto inbox for Lane
  - Canvas inbox (preferred) or Pronto for Surajit
  - [Sina's support section](#), Fridays 2p–3p on [Zoom](#)
  - Email [bozesurajit@fhda.edu](mailto:bozesurajit@fhda.edu), 24 hr turnaround
  - [Online](#) or [in-person](#) tutoring via the STEM center (Room 4213)
  - The section [GitHub repo](#) has lecture and section slides and solutions
- 





# Lecture Review: Expressions



+

- 

# Expressions

+



- The symbols **+**, **-**, **/**, etc. are the operators
- The terms operated upon (**x** and **y** in the previous slide) are the operands
- The evaluated result of the expression is typically stored in a variable (**z** in the previous slide) using the assignment operator **=**
- Keep in mind the difference between the two division operators:
  - **/** will always result in a float
  - **//** will always result in an integer, any remainder being discarded
  - **%** is the modulus operator for the remainder of integer division
- Given  $x = 8$  and  $y = 2$ , what is the value and type of these expressions?

**x \*\* y # ?**

**x / y # ?**

**x < y # ?**



# Expressions

+



- The symbols **+**, **-**, **/**, etc. are the operators
- The terms operated upon (**x** and **y** in the previous slide) are the operands
- The evaluated result of the expression is typically stored in a variable (**z** in the previous slide) using the assignment operator **=**
- Keep in mind the difference between the two division operators:
  - **/** will always result in a float
  - **//** will always result in an integer, any remainder being discarded
  - **%** is the modulus operator for the remainder of integer division
- Given  $x = 8$  and  $y = 2$ , what is the value and type of these expressions?

**x \*\* y # 64**

**x / y # 4.0**

**x < y # False**



# Expressions

- Watch out for floating point values! They are not stored precisely:

```
x = 1.9
```

```
y = 1
```

```
z = x - y
```

```
print(z)                # 0.8999999999999999
```

- Precision of float results is not reliable beyond the precision determinable by the inputs. Here, the value of **z** is not reliable past one decimal place.
- Can use **round(a, b)** where **a** is the value to round, **b** the number of decimal places:

```
print(round(z, 1))       # 0.9
```

# Expressions

- Operators have the following precedence:

( )	parentheses
**	exponentiation
-	unary negation
*, /, //, %	multiplication, division, integer division, modulus
+, -	addition, subtraction

- Operators with the same precedence (e.g., multiplication, division) are evaluated from left to right

# Expressions

- Compound operators: **+=**, **-=**, **\*=**, etc. combine the arithmetic and assignment operators in a single command
- Given initial values **x = 3** and **y = 2**, what would the following expressions evaluate to?
  - **x \*= y**      # **x = x \* y**
  - **x += 4**      # **x = x + 4**
  - **x /= y**      # **x = x / y**
  - **x %= y**      # **x = x % y**

# Expressions

- Compound operators: `+=`, `-=`, `*=`, etc. combine the arithmetic and assignment operators in a single command
- Given initial values `x = 3` and `y = 2`, what would the following expressions sequentially evaluate to?
  - `x *= y`      # `x = x * y`, `x = 6`
  - `x += 4`      # `x = x + 4`, `x = 10`
  - `x /= y`      # `x = x / y`, `x = 5.0`
  - `x %= y`      # `x = x % y`, `x = 1.0`
- Notice that the types of the results depend variously on the operands, the operators, or the results themselves
- Remember that types can be cast to a different type



# Any Questions?




# Lecture Review: Constants






# Constants

- In Python, a constant is a variable whose value does not change during the execution of the program
  - By convention, constants are named in **UPPER\_SNAKE\_CASE**
  - Why use constants?
    - To avoid "magic numbers"
    - To allow easy updates
    - To follow the principle of programming for the general case
  - Unlike most other programming languages, Python does not enforce constants; they are a convention
- 



# Constants

- Suppose I have a program that calculates various resources needed for running an office building based on its square footage. E.g., how many smoke detectors it needs, what the budget should be for HVAC, etc.
  - What constants would I need so I can reuse the program for any building?
- 

# Constants

- Suppose I have a program that calculates various resources needed for running an office building based on its square footage. E.g., how many smoke detectors it needs, what the budget should be for HVAC, etc.
- What constants would I need so I can reuse the program for any building?
  - **SMOKE\_DETECTOR\_FACTOR**  
*# e.g. 1 per 2000 sq.ft. or 0.0005*
  - **HVAC\_FACTOR**  
*# how many dollars to budget per sq.ft.*

The background is a solid orange color. It is decorated with various hand-drawn geometric shapes in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a large black circle in the bottom right, a white circle in the bottom right, a black plus sign in the bottom left, a white circle in the bottom center, a white triangle in the bottom center, and a black plus sign in the bottom center.

**Any Questions?**



# Lecture Review: random and math





# The random module and math library

- A module is a python file (with the extension .py) that contains code that can be reused in a different program.
- The **random** module allows generation of pseudo-random numbers
- A library, loosely speaking, is a collection of many modules
- The **math** library allows mathematical operations such as calculating square roots
- To use such external modules or libraries, your program needs an **import** statement such as **import math** or **import random**
- We've seen such a statement: **from karel.stanfordkarel import \***








# The random module and math library

```
from math import sqrt
import random
```

```
def main():
    for i in range(4):
        my_num = random.randint(200, 1000)
        my_sqrt = sqrt(my_num)
        print(f'Number is: {my_num}, square root is: {my_sqrt}')
```

```
if __name__ == '__main__':
    main()
```






# The random module and math library

```
from random import randint  
import math
```

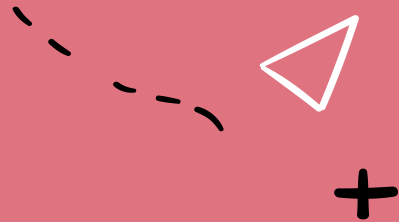
```
def main():  
    for i in range(4):  
        my_num = randint(200, 1000)  
        my_sqrt = math.sqrt(my_num)  
        print(f'Number is: {my_num}, square root is: {my_sqrt}')
```

```
if __name__ == '__main__':  
    main()
```



The background is a solid orange color. It is decorated with various hand-drawn geometric shapes in white and black. These include: a dashed line in the top left; a white triangle in the top center; a black zigzag line in the top right; a white circle in the top right; two parallel black lines in the top right; a white triangle in the top right; a large black circle in the bottom right; a white circle in the bottom right; a black plus sign in the bottom left; a white circle in the bottom center; a white triangle in the bottom center; and a black plus sign in the bottom center.

**Any Questions?**



Where do all these go?



# Flashback to Week 1 ...



# The Anatomy of a Program

- File name, comment, your name, date
- Import statement
- **main()** module with further comments. Indentation is important!
- Guard clause and invocation of **main()**

```
"""
main.py

Karel moves a beeper from the bottom row to the top row

Programmer: Surajit A Bose, Date: 2025.02.03
"""

from karel.stanfordkarel import *

def main():
    """Move Karel and beeper to top row

    Preconditions:
    - Karel is in the bottom left corner, facing east
    - There is a beeper immediately in front of Karel

    Postconditions:
    - Karel is in the top right corner, facing east
    - Beeper has been moved and is immediately behind Karel
    """
    pass # Delete this line and write your code here! :)

# There is no need to edit code beyond this point
if __name__ == '__main__':
    main()
```



# The structure of a Python program

- Python programs have a typical order:
  - Comment with filename, program overview, and programmer name
  - import statements
  - constants
  - `main()` function
  - helper functions
  - guard clause and invocation of `main()`
- A template for your use is [here](#)





# Section problem: Mars Weights

<https://codeinplace.stanford.edu/cs49-w24/ide/a/marsweight>







# Mars Weights

+



Due to the weaker gravity on Mars, an Earthling's weight on Mars is 37.8% of their weight on Earth. Write a Python program that prompts an Earthling to enter their weight on Earth and prints their calculated weight on Mars. The output should be rounded to two decimal places when necessary. Example:

**Enter a weight on Earth:** *120*

**The equivalent weight on Mars:** *45.36*

- What constant should we use?
- As what type should the input from the user be cast?

Let's get to work!



The background is a solid pink color. It is decorated with various hand-drawn geometric shapes in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a black circle in the bottom right, a white circle in the bottom right, a black plus sign in the bottom left, a white circle in the bottom left, a white triangle in the bottom left, and a black line in the bottom left.

# That's all, folks!

Next up: Control Flow!