Lecture 1: Introduction to Programming & Objects

Comp 102 (sec: A,B)

Fakhir Shaheen

Forman Christian University

Course Information

Instructor

- Fakhir Shaheen
  - fakhirshaheen@fccollege.edu.pk
  - linkedin.com/in/fakhirshaheen
- Office:
  - S-426 (E)
  - Hours: TR 9:15 am - 11:00 am
    W 9:00 am - 9:30 am

Grading Breakdown

Component	**Percentage**
Assignments *(5)*	15%
Labs *(13)*	10%
Quizzes *(6)*	15%
Mid	25%
Final	35%
Total	**100%**

* **NO** retakes for any missed quiz, lab or midterm.
* Assignments: 10% penalty per day late, up to 3 days. After that, assignment will not be accepted.
* Relative grading *(A and B separately)*

WARNING

You may miss **at most 5 lectures**.

A **6th absence** bars you from the **final exam** —

regardless of all other performance.

Attendance is taken from the **exit ticket** you hand in at the end of every lecture.
No name, no attendance.

WhatsApp Group

(only for announcements)

Discord Server

(complete course resources)

Course Contents

- **Building blocks** of programming
- **Organize** large software
- **Algorithms** to writing efficient software
- **Libraries** to write software quickly

Course Contents

- **Building blocks** of programming
  - variables, conditions, loops, functions etc
- **Organize** large software
  - Object oriented programming, Inheritance
- **Algorithms** to writing efficient software
  - Recursion, sorting, searching etc
- **Libraries** to write software quickly
  - Turtle, Tkinter, Data visualization, Software testing etc.

What is Programming?

What is Programming?

Big Idea

Programming is giving instructions to a computer to perform tasks.

What is Programming?

Question

Why do we need to give instructions to a computer?

Why can't we just solve the problem ourselves?

What is Programming?

Question

Why do we need to give instructions to a computer?
Why can't we just solve the problem ourselves?

- **Answer:**
  - Very high processing speed
  - Huge memory
  - High accuracy
  - Doesn't get tired

What is Programming?

Question

Why do we need to give instructions to a computer?
Why can't we just solve the problem ourselves?

- **Answer:**
  - Very high processing speed *billions of operations per second*
  - Huge memory *with high retrieval capability*
  - High accuracy 9.901×3.1415 *?*
  - Doesn't get tired *never ever !!*

Computers have limitations

- **Computers are stupid**
  - They do exactly what you tell them to do
- **Computers are not creative**
  - They can't think of new ways to solve problems



Programmatic Thinking

- **Commands Available:**
  - `forward()`

    move forward 1 step

- **Goal:**
  Get the cup



- **Commands Available:**
  - `forward()`

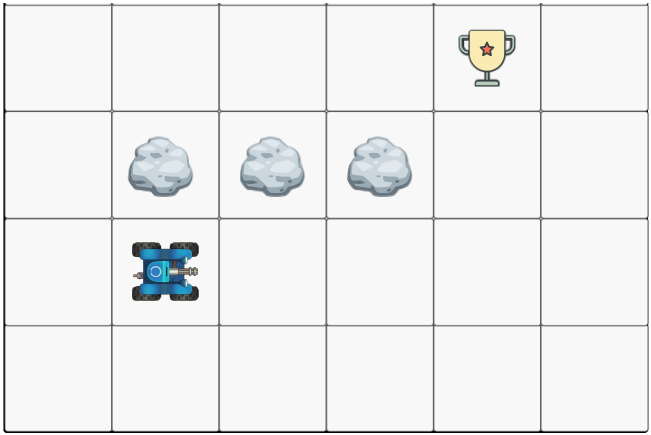
    move forward 1 step

- **Goal:**
  Get the cup

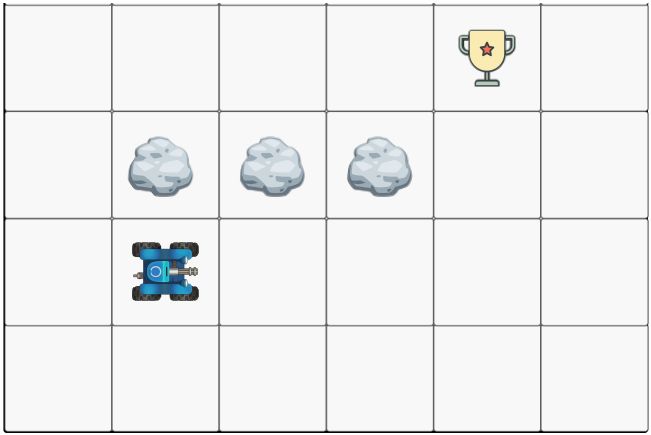
- **Solution:**
  1. `forward()`
  2. `forward()`
  3. `forward()`



- **Commands Available:**
  - `forward()`
    move forward 1 step
  - `left()`
    turn left 90°

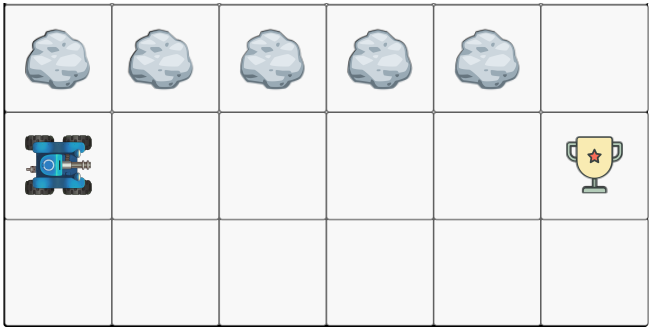


- **Commands Available:**
  - `forward()`
    move forward 1 step
  - `left()`
    turn left 90°
- **Solution:**
  1. `forward()`
  2. `forward()`
  3. `forward()`
  4. `left()`
  5. `forward()`
  6. `forward()`



- **Commands Available:**
  - `forward()` $\times 1$
  - `repeat n:` $\times 1$

    repeats commands n times

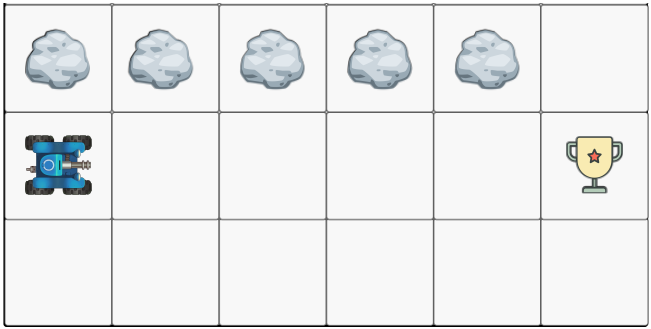


- **Commands Available:**
  - `forward()` ×1
  - `repeat n:` ×1

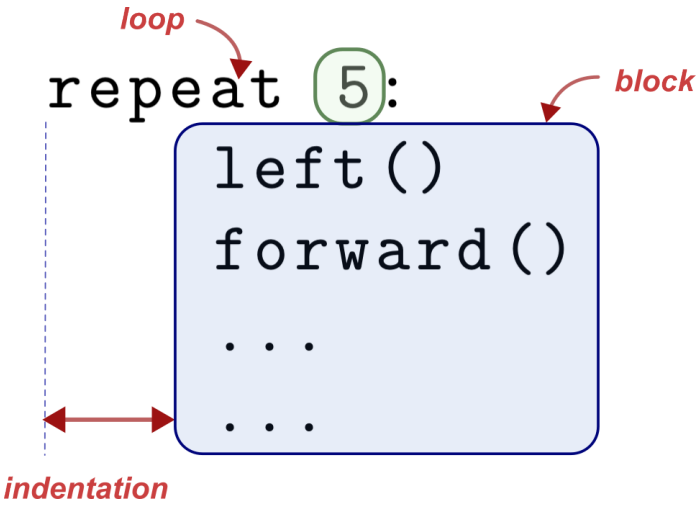
    repeats commands n times

- **Solution:**

```
1 repeat 5:
2     forward()
```



repeat command Syntax



You Try:

```
repeat 2:
    print(1)
    print(2)
print(3)
```

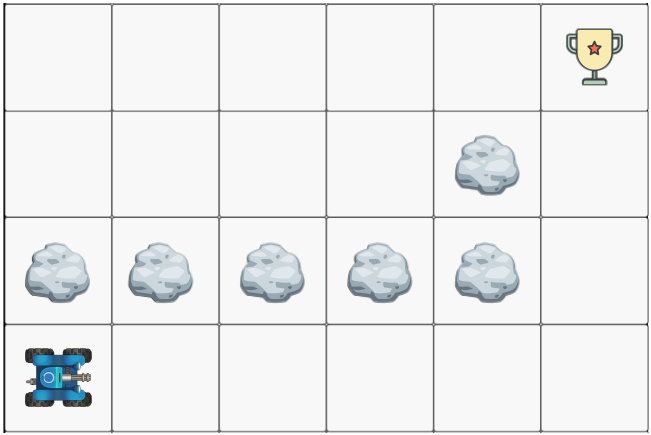
- **Output:**

You Try:

```
repeat 2:
    print(1)
    print(2)
print(3)
```

- **Output:**
- 1
- 2
- 1
- 2
- 3

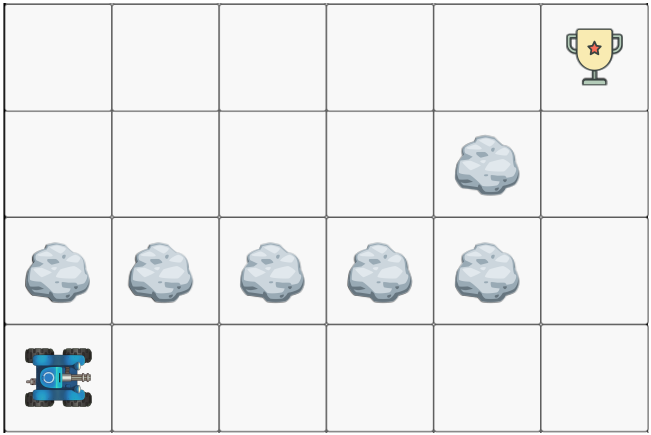
- **Commands Available:**
  - `forward()` ×2
  - `left()` ×1
  - `repeat n:` ×2



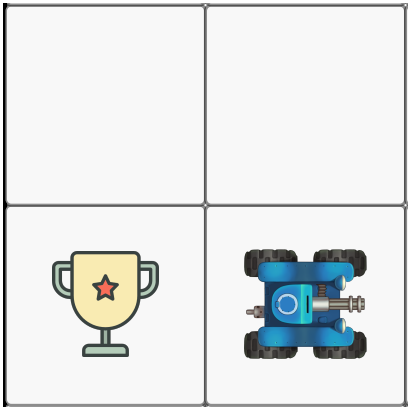
- **Commands Available:**
  - `forward()` ×2
  - `left()` ×1
  - `repeat n:` ×2

- **Solution:**

```
1 repeat 5:
2     forward()
3 left()
4 repeat 3:
5     forward()
```

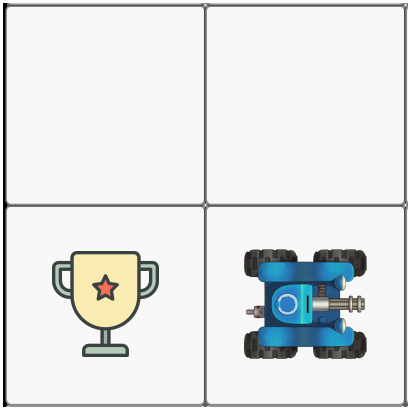


- **Commands Available:**
  - `forward()` $\times 1$
  - `left()` $\times 1$
  - `repeat n:` $\times 1$



- **Commands Available:**
  - `forward()` ×1
  - `left()` ×1
  - `repeat n:` ×1
- **Solution:**

```
1 repeat 3:
2     left()
3     forward()
```



Programming Languages

Question

What language does a computer understand?

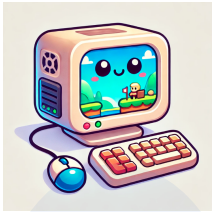
Computer understands **only** 1s and 0s

(a.k.a **machine language**)



Machine Commands	Actual Meaning
10110101000101	*Add 3 and 5*
1000001101	*Store the result at M0*
1110010100100011	*Multiply 5 and 7*
1001101	*Store the result at M1*
11111111000001	*Subtract M0 from M1*

High Level Languages





ASM

Machine language

Low level languages

C/C++
Rust
Go

Python
Javascript

High level languages

English

Natural languages

High Level Languages



does it understand?

NO !



ASM

Machine language

C/C++
Rust
Go

Python
Javascript

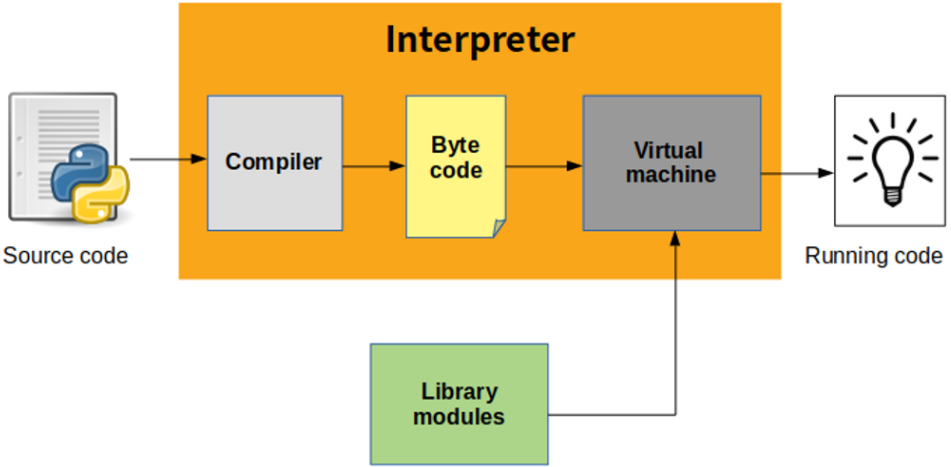
English

Natural languages

Low level languages

High level languages

Converting high level language to machine language



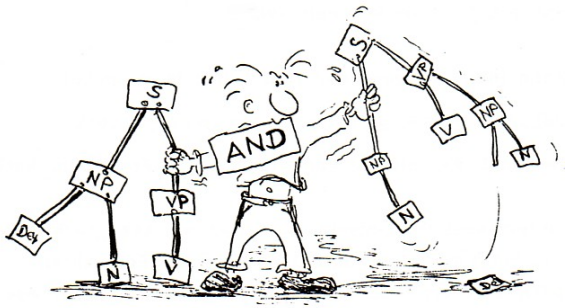
Language Fundamentals

Primitive constructs:

1. English: *words*
2. Programming Languages:
   numbers, strings, simple operators

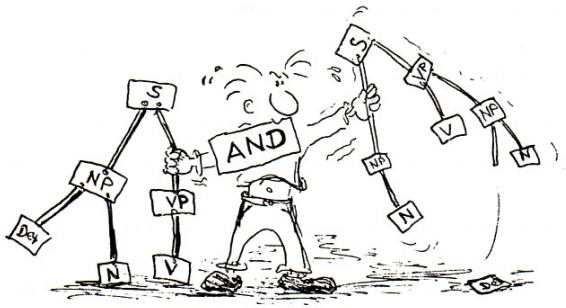
1. Syntax *(rules)*

- English:
  - "boy football cat" ✗
  - "boy plays football" ✔



1. Syntax *(rules)*

- English:
  - "boy football cat" ✗
  - "boy plays football" ✔
- Python:
  - `'hi'5` ✗
  - `'hi' * 5` ✔



2. Semantics *(meaning)*

- English:
  - "Me are hungry" ✗
  - "I am hungry" ✔



2. Semantics *(meaning)*

- English:
  - "Me are hungry" ✗
  - "I am hungry" ✔
- Python:
  - `'hi' + 5` ✗
  - `'hi' + 'there'` ✔



2. Semantics *(meaning)*

English:

- Can have multiple meanings
  "The chicken is ready to eat."

Word	Semantic
pen	a writing tool
pen	a livestock's enclosure
pen	a portable enclosure for a baby
pen	a correctional institution
pen	a female swan

2. Semantics *(meaning)*

English:

- Can have multiple meanings
  "The chicken is ready to eat."

Programming languages:

- Have **exactly One** meaning.
  but it may not be the meaning you intended

Word	Semantic
pen	a writing tool
pen	a livestock's enclosure
pen	a portable enclosure for a baby
pen	a correctional institution
pen	a female swan

You Try:



Which of the following are syntax errors and which are semantic errors?

1. Add two apples and five oranges.
2. `2 5 +`
3. `3 / 'hi'`
4. `'hi' * 5`

You Try:



Which of the following are syntax errors and which are semantic errors?

1. Add two apples and five oranges. (semantic)
2. `2 5 +` (syntax)
3. `3 / 'hi'` (semantic)
4. `'hi' * 5` (no error)

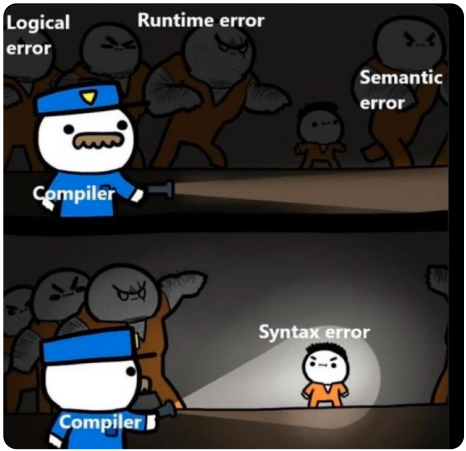
Where things go wrong:-

Syntax Errors:

- Very common, caught early

Semantics Errors:

- Some languages check early, some don't



Where things go wrong:-

Syntax Errors:

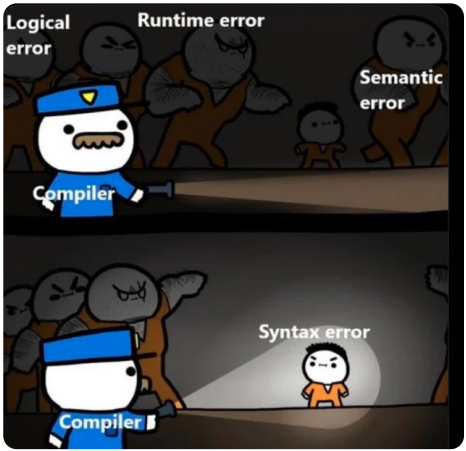
- Very common, caught early

Semantics Errors:

- Some languages check early, some don't

No Errors: *(but different meaning than intended)*

- Program crashes, stops running
- Program runs forever
- Program gives an answer, but it's wrong!



Introduction to Objects

1. Objects *(every object has:)*

1. a **type**
2. set of **attributes**, a.k.a data that it holds
3. set of **operations** that we can perform on it

1. Objects *(every object has:)*

1. a **type**
2. set of **attributes**, a.k.a data that it holds
3. set of **operations** that we can perform on it

- for example:
  1. 30
     - ⋆ is a number *(type)*
     - ⋆ we can add, subtract, multiply, divide, etc. *(operations)*

1. Objects *(every object has:)*

1. a **type**
2. set of **attributes**, a.k.a data that it holds
3. set of **operations** that we can perform on it

- for example:
  1. 30
     - is a number *(type)*
     - we can add, subtract, multiply, divide, etc. *(operations)*
  2. `'Hello'`
     - is a sequence of characters a.k.a `string` *(type)*
     - we can add, multiply but can't divide, etc. *(operations)*

Objects

1. **Scalar** *one single value*

2. **Non-Scalar** *a container of other objects*

Objects

1. **Scalar** *one single value*
   - Python treats it as **one whole thing**
   - It does **not** contain other objects inside it
   - Numbers: `8.3`, `27` Truth values: `True`, `False`
2. **Non-Scalar** *a container of other objects*

Objects

1. **Scalar** *one single value*
   - Python treats it as **one whole thing**
   - It does **not** contain other objects inside it
   - Numbers: `8.3`, `27` Truth values: `True`, `False`

2. **Non-Scalar** *a container of other objects*
   - It **holds other objects** inside it
   - Python can hand you those objects **one at a time**
   - Strings: `'Hello'` holds `'H'`, `'e'`, `'l'`, `'l'`, `'o'`
   - Lists, Dictionaries etc

“But 27 has a 2 and a 7 in it!”

“But 27 has a 2 and a 7 in it!”

The **same number**, written three ways:

```
   27
11011   (binary)
XXVII   (Roman)
```

Different digits, **one** number.

The digits are how we **write** it, not objects stored **inside** it.

“But 27 has a 2 and a 7 in it!”

The **same number**, written three ways:

```
   27
11011   (binary)
XXVII   (Roman)
```

Different digits, **one** number.

The digits are how we **write** it, not objects stored **inside** it.

Ask Python for the first piece:

```
>>> '27'[0]
'2'
>>> 27[0]
TypeError
```

`'27'` is a string. It holds the characters `'2'` and `'7'`. **Non-scalar.**

27 is a number. One quantity, nothing inside to hand out. **Scalar.**

You Try

Which of the following are scalar and which are non-scalar objects?

1. `3.5`
2. `27`
3. `'27'`
4. `'Hello'`
5. `True`
6. `[1, 2, 3]`

You Try

Which of the following are scalar and which are non-scalar objects?

1. `3.5` **scalar**
2. `27` **scalar**
3. `'27'` **non-scalar**
4. `'Hello'` **non-scalar**
5. `True` **scalar**
6. `[1, 2, 3]` **non-scalar**

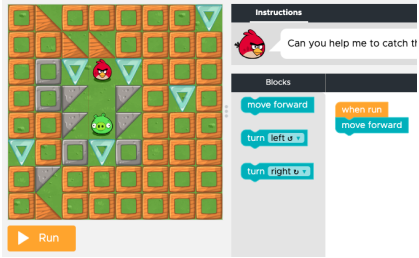
Big Idea

Everything in Python is an **Object**.

Summary

Resources

- **Block based games:**
  - The Maze
  - The Artist
  - The Farmer
- **Code based games:** *(Steam)*
  - The Farmer Was Replaced
- **Visual Programming:**
  - Craftomation 101: Programming & Craft





Summary

- **Programming** is giving instructions to a computer to perform tasks.
- Computers are stupid and not creative.
- **Programmatic thinking** is breaking down a problem into smaller steps.
- **Commands** are used to instruct the computer.
- **Machine language** vs **High level languages**
- **Syntax** and **Semantics**
- **Objects**: Everything in Python is an object
- **Scalar** vs **Non-scalar** objects

Summary

Questions?

Exit Ticket

Today's question *(2 minutes)*

Label each as a **syntax** error or a **semantic** error:

(a) `7 2 *`

(b) `'hi' - 4`

COMP 102 — Exit Ticket #1

Name: _____

Roll #: _____ Date: _____

Answer:

Hand it to me on your way out.
No name, no attendance.