計算物理概論

Introduction to Computational Physics (PHYS290000)

Lecture 3: Basic Python (part 2)

Instructor: 潘國全
kuochuan.pan@gapp.nthu.edu.tw

Last week

1. Basic Python Programming:
2. Python syntax
3. Variables
4. Collections
5. Control Flow
6. Functions

Today's plan

1. Warm-up
2. Exercise: Angry bird game
3. Formatted output
4. Read/Write files
5. Classes
6. Modules

Basic Python Programming

Warm-up

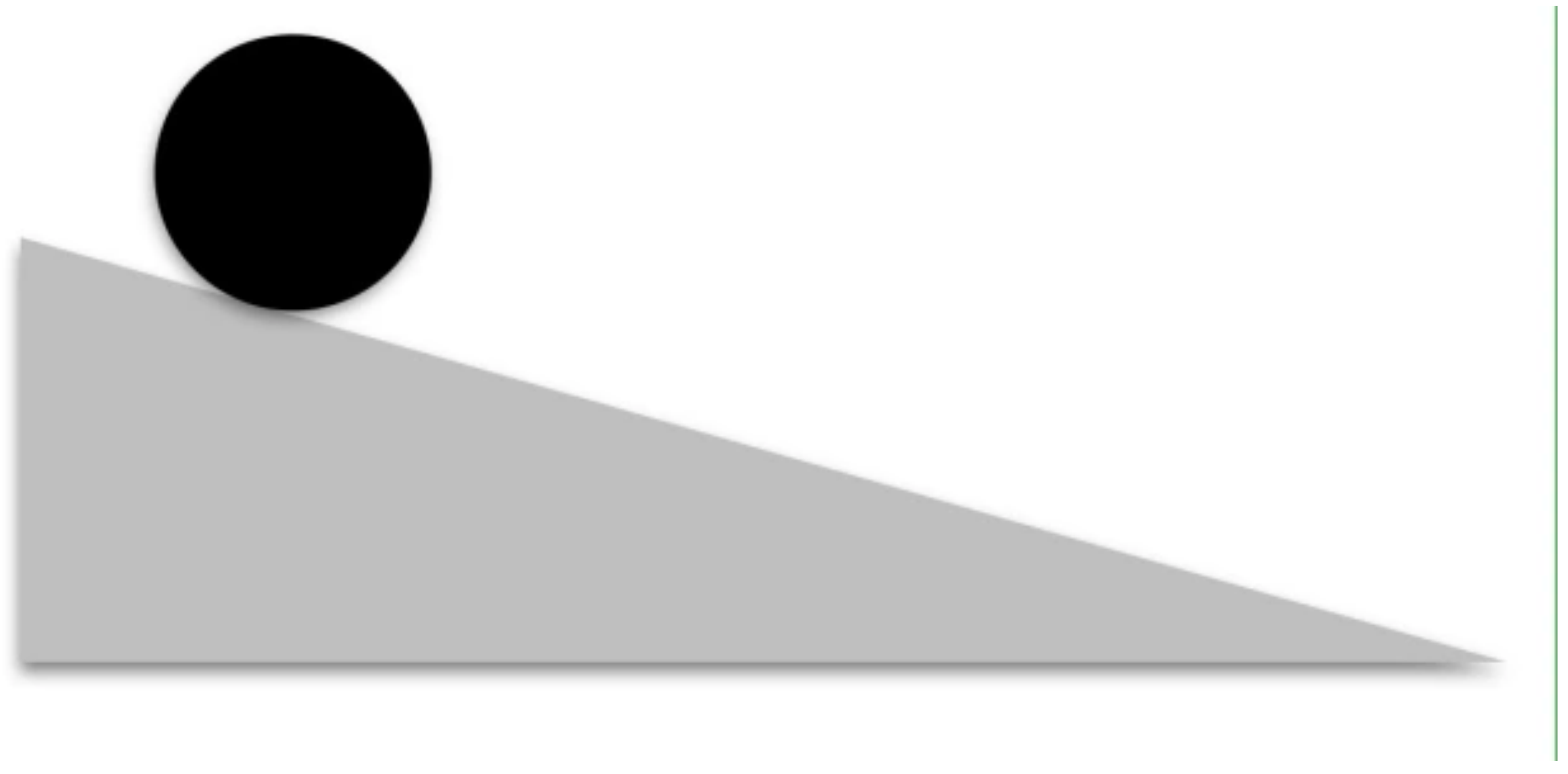
1. A ball (m = 1.5 kg) is released freely on a slope at height H, the height and velocity of the ball are measured in the right table.

2. Create three lists to store the data of the ball's time, height, and velocity.

3. Loop all the data and compute the total energy of the ball (kinetic + potential); assume g=9.81 m/s/s

4. Store the total energy in another list

5. Use a loop to print out the value of total energy and compute the average.



Time [s],	Velocity [m/s],	Height [m]
0.004	0.013	2.004
0.215	0.679	1.987
0.417	1.306	1.988
0.605	1.920	1.862
0.813	2.638	1.684
1.003	3.236	1.573
1.209	3.897	1.327
1.411	4.516	1.070
1.602	5.010	0.768
1.805	5.803	0.416

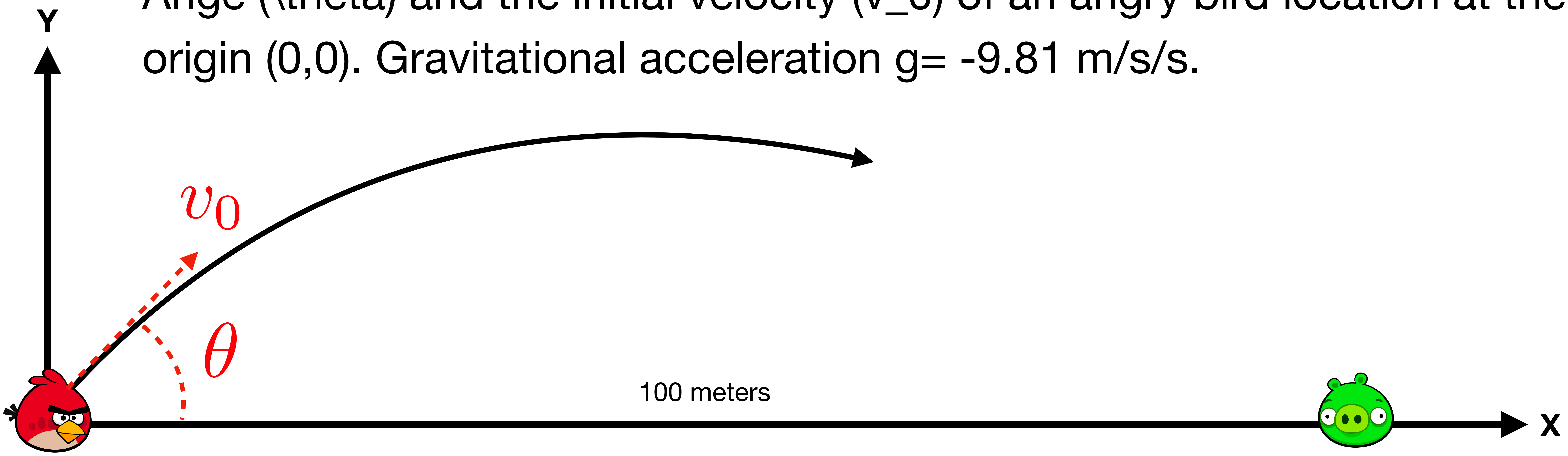
Warm-up

(See demo)

Angry Bird Game

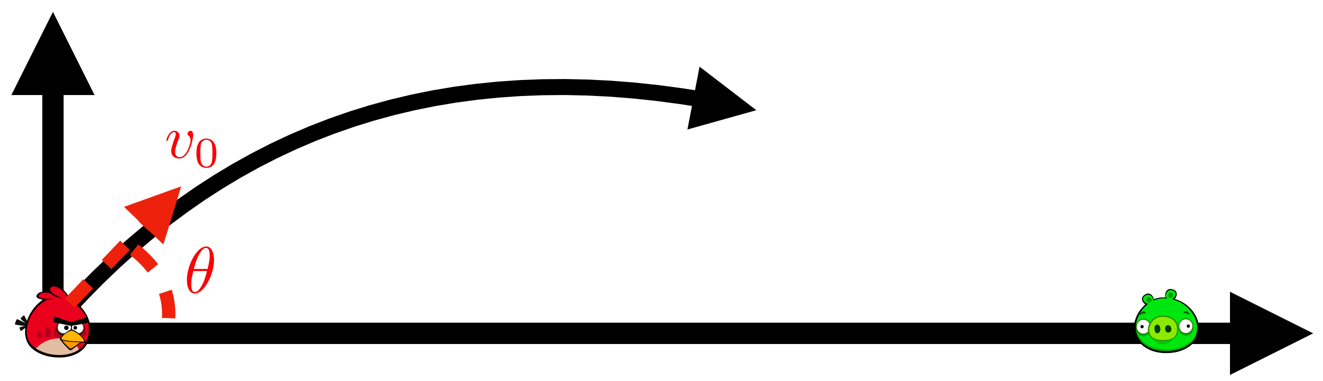
Exercise: Angry bird game

1. In this exercise, we will use what we have learned so far to design a simple text-based angry bird game.
2. Assuming no air-resistances, use analytical solution to describe the trajectory of an angry bird. Initial conditions include only the inclination Ange (\theta) and the initial velocity (v_0) of an angry bird location at the origin (0,0). Gravitational acceleration g= -9.81 m/s/s.



Exercise: Angry bird game

1. In this exercise, we will use what we have learned so far to design a simple text-based angry bird game.
2. Assuming no air-resistances, use analytical solution to describe the trajectory of an angry bird. Initial conditions include only the inclination Ange (\theta) and the initial velocity (v_0) of an angry bird location at the origin (0,0). Gravitational acceleration g= -9.81 m/s/s.
3. The target (the pig) is located at 100 m away from the bird (+100,0)
4. A hit is defined when the distance difference between the landing point and the pig is less than 1 m.



Exercise: Angry bird game

1. A “turn” include one “input phase” and one “output phase”.
2. During the input phase, the program should ask for user’s input.
    1. An input is a long string which separate variables by spaces.
    2. Inputs could be either a string with two values (v_0 and \theta; e.g. “10.0 45.0”) or “quit”. Use MKS unit system.
    3. If the player gives wrong input, the program should return a warning and ask to input again (move to the next turn).
3. During the output phase,
    1. If the input is “quit”, print a message and leave the program.
    2. If the input is a string (with two values), the program should calculate the trajectory and then check if the angry bird hits the pig
        1. If it hits, print a message to congrats the player and then ask if the player wants to play again or not.
        2. If not, print a message to describe how close was the trajectory and then move to the next turn.
4. The next turn just repeat the same flow.

Exercise: Angry bird game

python angry.py

```
(comphys)
~/codes/ComputationalPhysics/ComputationalPhysics/angry_bird/version1 (main*) » python angry.py

    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>
```

Exercise: Angry bird game

pan@vega:~/codes/ComputationalPhysics/ComputationalPhysics/angry_bird/version1

```
(comphys)
~/codes/ComputationalPhysics/ComputationalPhysics/angry_bird/version1 (main*) » python angry.py

    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>
        quit
>>>
>>>  Thanks for playing. Bye bye!
>>>
(comphys)
~/codes/ComputationalPhysics/ComputationalPhysics/angry_bird/version1 (main*) »
```

Exercise: Angry bird game

python angry.py

```
(comphys)
~/codes/ComputationalPhysics/ComputationalPhysics/angry_bird/version1 (main*) » python angry.py

    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>
        213a 214
>>>
>>>  Incorrect inputs. Please type again.
>>>

    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>
```

Exercise: Angry bird game

python angry.py

```
(comphys)
~/codes/ComputationalPhysics/ComputationalPhysics/angry_bird/version1 (main*) » python angry.py

    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>
       30 45
>>>
>>>  input values are 30.0 [ms] and 45.0 [degree]
>>>
>>>
>>>  The angry bird want too close! try increasing the angle or velocity. [dx = -8.256880733944953]
>>>

    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>
       35 47
>>>
>>>  input values are 35.0 [ms] and 47.0 [degree]
>>>
>>>
>>>  The angry bird went too far! try lowering the angle or velocity. [dx = 24.56839567464675]
>>>

    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>
```

Exercise: Angry bird game

```
    Please type two values for the angry bird's initial velocity [m/s] and inclindation angle [deg].
    (example:  30.0 45.0)

    or type "quit" to quit the game

>>>


>>>
>>>  input values are       [ms] and       [degree]
>>>
>>>
>>>   ---  Bang! You win!  ---
>>>
Press any key to continue...
```

Exercise: Angry bird game

(See demo)

Hints: Angry bird game

1. Running the game is in a while loop

```
finished = False

while not finished:

    # repeat turns
    finished = run_one_turn()

return
```

2. The “try; except” statement can be used to check if the input is valid or not.

```
try:
    velocity = float(velocity)
    angle    = float(angle)
    values   = [velocity, angle]
    return OK, values
except:
    pass
```

Formatted output

Formatted output

1. So far, we only learned the basic usage of the “print” statement

2. More usages: formatted string (f””)

```python
name   = "Albert Einstein"
gender = "male"
age    = 45
iq     = 200

if gender=="male":
    pronoun = "his"
elif gender=="female":
    pronoun = "her"
else:
    pronoun = "hir"

print(f"{name} is {age} years old. {pronoun} IQ is {iq}.")
```

[3] ✓ 0.0s

```
Albert Einstein is 45 years old. his IQ is 200.
```

Formatted output

```python
mass     = 1.235
velocity = 1.478
print(f"debugging: {mass=}, {velocity=}")
```

✓ 0.0s

```
debugging: mass=1.235 velocity=1.478
```

```python
import math
print("pi is",math.pi)
print(f"pi is {math.pi}")
print("pi is {}".format(math.pi))
print("pi is {:.3f}".format(math.pi)) # 3 digits
print(f"pi is {math.pi:.3f}")         # 3 digits
print("pi is {:.3e}".format(math.pi)) # scientific
```

[15] ✓ 0.0s

```
...
pi is 3.141592653589793
pi is 3.141592653589793
pi is 3.141592653589793
pi is 3.142
pi is 3.142
pi is 3.142e+00
```

Formatted output

```
data = [
    ["Einstein", 2/3],
    ["Maxwell", math.pi],
    ["Newton", math.e]
]
for item in data:
    # <  left alignment
    # >  right alignment
    # ^  center alignment
    print("{:<10} | {:>12.6f}".format(item[0],item[1]))
```

✓ 0.0s

```
Einstein   |     0.666667
Maxwell    |     3.141593
Newton     |     2.718282
```

Read/Write Files

Write Files

```python
f = open('file1.txt',mode='w',encoding="utf-8")
# mode (default='r'):
# 'w' : write (existing file will be overwritted)
# 'a' : appending to an existing file
# 'r' : only read a file
# 'r+': both read and write
# 'rb': read in bianry format
# 'wb': write in binary foramt
f.write("This the first line.\n")
f.write("This the second line. ")
f.write("Still in the second line.")
f.close()
```

✓ 0.0s

```
This the first line.
This the second line. Still in the second line.
```

Read Files

```
f = open('file1.txt',mode='r')
msg = f.read()
print(msg)
f.close()
```

✓ 0.0s

```
This the first line.
This the second line. Still in the second line.
```

```
f = open('file1.txt',mode='r')
for line in f:
    print(line)
```

✓ 0.0s

```
This the first line.

This the second line. Still in the second line.
```

Note: for scientific IO, there are many other packages to use

Exercise:

1. Modify your angry bird game
2. When calculating the trajectory, record the time and position of the trajectory in every time step dt = 0.1 sec into a file named “trajectory.txt”.
3. Make sure your values in the trajectory file are all aligned.

Miniexam 3 & Homework 1

1. Check google classroom

2. Write your solutions in .py files (do not use Jupyter notebook)

3. Please write comments and explanation of your answers

4. Your code should always have if __name__=='__main__':

5. Your code must able to executed by python your_code.py

6. Answer will be show up with the "print()" function.