REPETITION AND SELECTION

Example: Solving a Quadratic Eqt.

- Remember what we learned in high school…

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Let's try to implement it in Python

solve_qe1.py - /Volumes/GoogleDrive/My Drive/Courses/YSC2221/Lectur

```python
from math import sqrt

def solve_qe(a,b,c):
    delta = b**2 - 4*a*c
    ans1 = (-b + sqrt(delta))/(2*a)
    ans2 = (-b - sqrt(delta))/(2*a)
    print("The two solutions are " + str(ans1)
          + " and " + str(ans2))
```

```
>>> solve_qe(1,5,6)
The two solutions are -2.0 and -3.0
>>> solve_qe(1,4,4)
The two solutions are -2.0 and -2.0
>>>
```

However…

```
>>> solve_qe(1,-5,6)
The two solutions are 3.0 and 2.0
>>> solve_qe(1,1,8)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    solve_qe(1,1,8)
  File "C:\Users\dcschl\Google Drive\Courses\YSC22
21\Lectures\solve_qe1.py", line 5, in solve_qe
    ans1 = (-b + sqrt(delta))/(2*a)
ValueError: math domain error
```

- Why?

solve_qe1.py - /Volumes/GoogleDrive/My Drive/Courses/YSC2221/Lectur

```
from math import sqrt

def solve_qe(a,b,c):
    delta = b**2 - 4*a*c
    ans1 = (-b + sqrt(delta))/(2*a)
    ans2 = (-b - sqrt(delta))/(2*a)
    print("The two solutions are " + str(ans1)
          + " and " + str(ans2))
```

delta = 25-24 = 1 > 0

```
>>> solve_qe(1,-5,6)
The two solutions are 3.0 and 2.0
>>> solve_qe(1,1,8)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    solve_qe(1,1,8)
  File "C:\Users\dcschl\Google Drive\Courses\YSC22
21\Lectures\solve_qe1.py", line 5, in solve_qe
```

delta = 1 – 32 = -31 < 0

Example: Solving a Quadratic Eqt.

- Remember what we learned in high school…

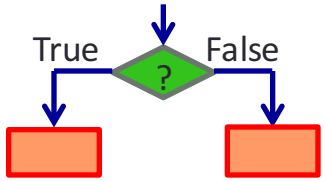
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- If delta < 0
  - The equation has no real solution

- So we cannot call sqrt() if delta is negative

Making Choices

Control Structure: Selection




```
If (a condition is true)
  Do A
Else
  Do B
```

Can be MORE THAN one single instruction

- For example:

```
If (I have $1000000000000000)
  Buy a car
  Eat a lot of buffets
  Go travel
  Quit NUS!
Else
  Be good and study
```

Condition

```
def solve_qe(a,b,c):
    delta = b**2 - 4*a*c
    if delta >= 0:
        ans1 = (-b + sqrt(delta))/(2*a)
        ans2 = (-b - sqrt(delta))/(2*a)
        print("The two solutions are " + str(ans1)
              + " and " + str(ans2))
    else:
        print("The equation has no real root")
```

condition

If the condition is True

```
def solve_qe(a,b,c):
    delta = b**2 - 4*a*c
    if delta >= 0:
        ans1 = (-b + sqrt(delta))/(2*a)
        ans2 = (-b - sqrt(delta))/(2*a)
        print("The two solutions are " + str(ans1)
              + " and " + str(ans2))
    else:
        print("The equation has no real root")
```

If the condition is False

```
def solve_qe(a,b,c):
    delta = b**2 - 4*a*c
    if delta >= 0:
        ans1 = (-b + sqrt(delta))/(2*a)
        ans2 = (-b - sqrt(delta))/(2*a)
        print("The two solutions are " + str(ans1)
              + " and " + str(ans2))
    else:
        print("The equation has no real root")
```

Conditional

Syntax

```
if <expr>:
    statement(s)
```

Example

```
>>> my_money = 1000
>>> if my_money > 0:
        print('Good')
'Good'
```

indentation

Conditional

Syntax

```
if <expr>:
    statement(s)
```

indentation

Example

```
>>> my_money = 1000
>>> if my_money > 0:
        print('Good')
        print('Good')
        print('Good')

'Good'
'Good'
'Good'
```

Conditional

Syntax

```
if <expr>:
    statement(s)
else:
    statement(s)
```

Example

```
>>> my_account = 1000
>>> if my_account > 0:
        print('rich')
else:
        print('broke')
'rich'
```

Conditional (Nested)

Syntax

```
if <expr>:
  if <expr>:
    statement(s)
```

Example

```
a = 4
if a < 10:
    if a < 1:
        print('Here')
```

Print nothing

Conditional

Syntax

```
if <expr>:
    statement(s)
else:
    statement(s)
```

Example

```
>>> my_account = 1000
>>> if my_account < 0:
        print('poor')
else:
    if my_account > 1:
        print(‘v rich')

v rich
```

Clumsy

Conditional

Syntax

```
if <expr>:
    statement(s)
elif <expr>:
    statements(s)
else:
    statement(s)
```

Example

```
>>> a = -3
>>> if a > 0:
        print('yes')
elif a == 0:
        print('no')
else:
        print('huh')
'huh'
```

Conditional

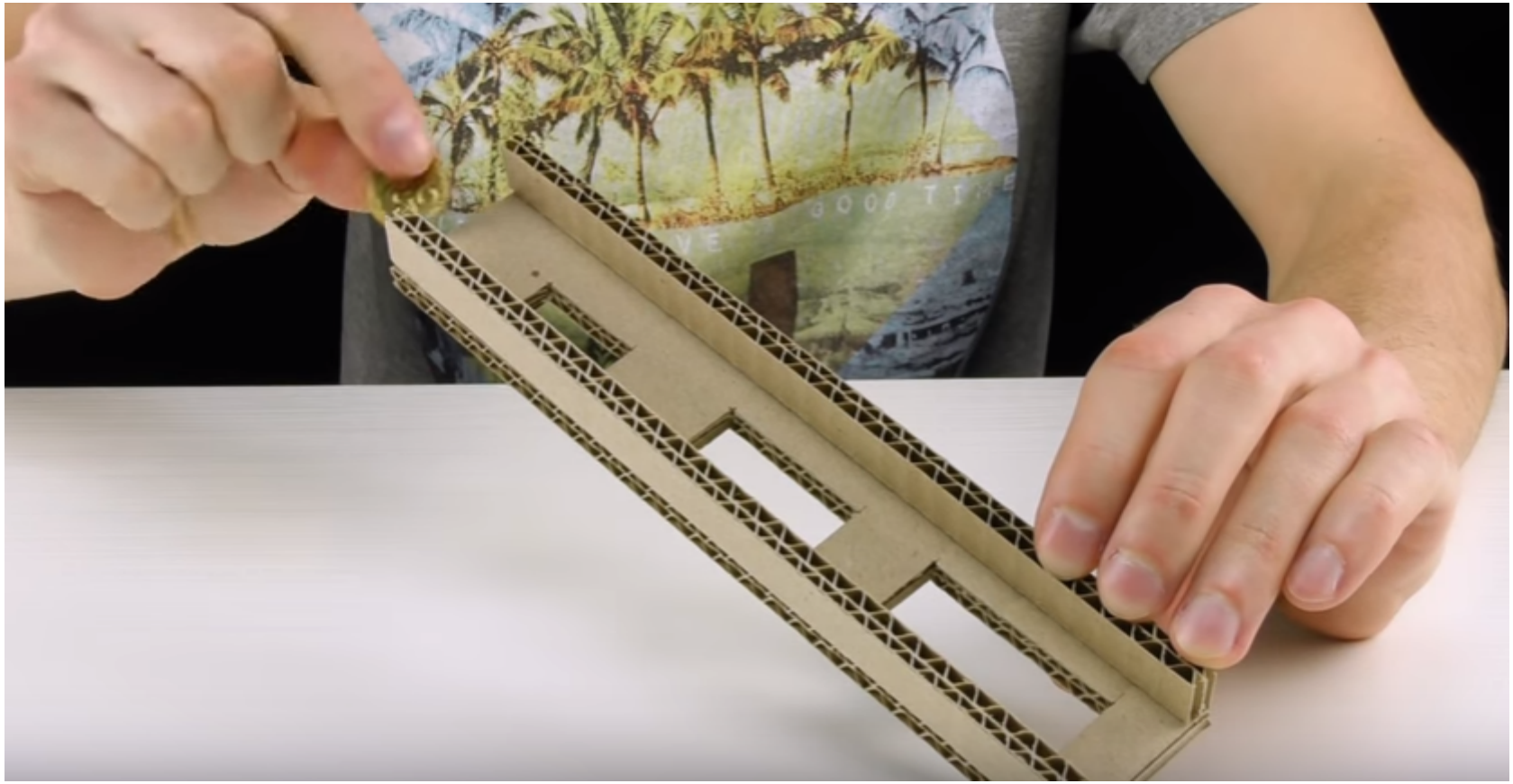
Syntax

```
if <expr>:
    statement(s)
elif <expr>:
    statements(s)
elif <expr>:
    statements(s)
else:
    statement(s)
```

Can be many

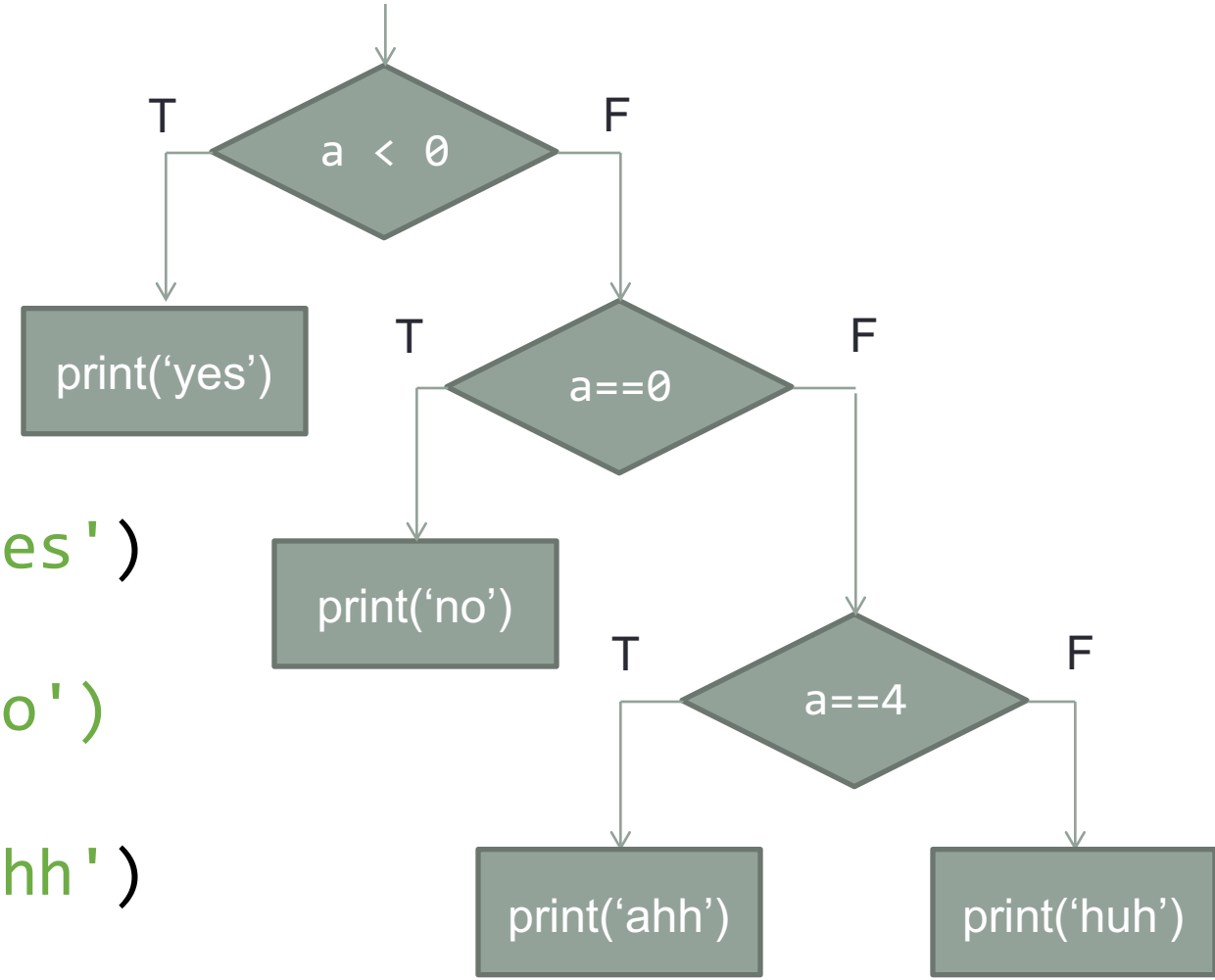
Example

```
>>> a = 4
>>> if a > 0:
        print('yes')
elif a == 0:
        print('no')
elif a == 4:
        print('ahh')
else:
        print('huh')
'yes'
```



e.g.

```
>>> a = 4
>>> if a < 0:
        print('yes')
elif a == 0:
        print('no')
elif a == 4:
        print('ahh')
else:
        print('huh')
'ahh'
```



Homework: Figure out ALL conditions

```
def solve_qe(a,b,c):
    delta = b**2 - 4*a*c
    if delta >= 0:
        ans1 = (-b + sqrt(delta))/(2*a)
        ans2 = (-b - sqrt(delta))/(2*a)
        print("The two solutions are " + str(ans1)
              + " and " + str(ans2))
    else:
        print("The equation has no real root")
```

condition

Repetition


I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.

Flipping a coin

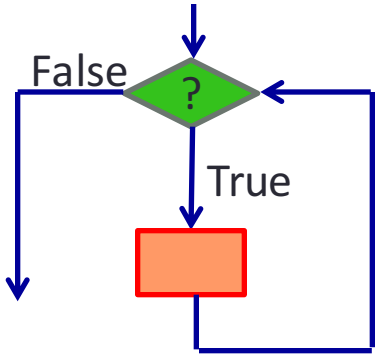
- A coin is “fair” if the probably of getting a head is equal to a tail
  - = 0.5
- How to test a coin is fair?
- Flip 1000 times!

```python
import random

def flipCoins():
    print('I will flip a coin 1000 times. ')
    print('Guess how many times it will come up heads. ')
    flips = 0
    heads = 0
    while flips < 1000:
        if random.randint(0, 1) == 1:
            heads = heads + 1
        flips = flips + 1
```

Randomly generate either 0 or 1

Control Structure: Repetition



- While (a condition)
  - Do something

- For example

```
While (I am hungry)
  Eat a bun
```

- Again, can be more than one single instruction

```
While(I have money in bank)
  Take money out from bank
  Eat an expensive meal
  While(I have money in my wallet)
    Go Shopping
```

Repetition (Infinite)

```
while True:
    print('Ah...')
```

```
a = 0
while > 0:
    a = a + 1
    print(a)
```

Repetition

Syntax

```
while <expr>:
    statement(s)
```

Example

indentation

```
>>> a = 0
>>> while a < 5:
        a = a + 1
        print(a)
1
2
3
4
5
```

Factorial

- The factorial n! is defined by

$$n! = 1 \times 2 \times 3 \times \cdots \times n$$

- Write a function for factorial?

```
def factorial(n):
    ans = 1
    i = 1
    while i <= n:
        ans = ans * i
        i = i + 1
    print(ans)
```

```
>>> factorial(3)
6
>>> factorial(6)
720
>>>
```

Repetition (nested)

inentation

Syntax

```
while <expr>:
    while <expr>:
        statement(s)
```

```
#
##
###
#
##
###
#
##
###
#
##
###
#
##
###
```

Example

```
def nestedWhile():
    i = 0
    while i < 5:
        i += 1
        j = 0
        while j < 3:
            j += 1
            print ('#' * j)
```

Repetition, a Very Common Pattern

9 out of 10 times you will do

```
>>> a = 0
>>> while a < N:
        a = a + 1
        do something
```

For loop

```
for i in range(0,N):
    do something
```

Another Version of Flipping Coins

```
def flipCoins():
    print('I will flip a coin 1000 times. ')
    print('Guess how many times it will come up heads. ')

    heads = 0
    for flip in range(0,1000):
        if random.randint(0, 1) == 1:
            heads = heads + 1
```

Another Repetition Flow Control: “For”

Syntax

```
for i in range(n,m):
    statement(s)
```

Example

```
for i in range(0,5):
    print(i)
```

```
0
1
2
3
4
```

Exclusive

Another Repetition Flow Control: “For”

Example

```
for i in range(0,5):
      print(i)
```

```
0
1
2
3
4
```

Interpreted as

```
i=0
print(i)
i=1
print(i)
i=2
print(i)
i=3
print(i)
i=4
print(i)
```

Factorial again

From 1 to n
(Exclusive)

```
>>> def factorial(n):
      ans = 1
      for i in range(1,n+1):
            ans *= i
      return ans

>>> factorial(5)
120
>>>
```

Let’s play a game

```
>>> guessANum()
I have a number in mind between 0 and 99
Guess a number: 50
Too big
Guess a number: 25
Too big
Guess a number: 12
Too big
Guess a number: 6
Too small
Guess a number: 9
Too big
Guess a number: 7
Bingo!!!
>>>
```

guessANum.py

```
import random

def guessANum():
    secret = random.randint(0,99)    # 0 <= secret <= 99
    guess = -1
    print('I have a number in mind between 0 and 99')
    while guess != secret:
        guess = int(input('Guess a number: '))
        if guess == secret:
            print('Bingo!!! You got it! ')
        elif guess < secret:
            print('Your number is too small')
        else:
            print('Your number is too big')

guessANum()
```

Repeat until the condition is False

guessANum.py

```
import random

def guessANum():
    secret = random.randint(0,99)    # 0 <= secret <= 99
    guess = -1
    print('I have a number in mind between 0 and 99')
    while guess != secret:
        guess = int(input('Guess a number: '))
        if guess == secret:
            print('Bingo!!! The answer is ' + str(secret)))
        elif guess < secret:
            print('Your number is too small')
        else:
            print('Your number is too big')

guessANum()
```

Repeat until the condition is False

Can you spot the difference?

Example 1

```
def foo():
  if True:
    if False:
        print(1)
    else:
        print(2)
```

Example 2

```
def foo():
  if True:
    if False:
      print(1)
  else:
    print(2)
```

Tips

- A “while” or “if” block starts with a colon “:”
- Remember
  - When there is a colon, there are indentations
  - When there are indentations, before these there is a colon
- The inclusive/exclusive range is a pain