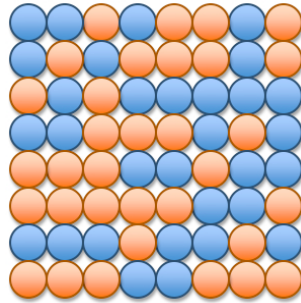




UNIT 9B

Randomness in Computation: Games with Random Numbers

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

1

Rolling a die



```
from random import randint  
def roll():  
    return randint(0,15110) % 6 + 1
```

OR

```
def roll():  
    return randint(1,6)
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

2

Another die



```
def roll():                                #wrong
    return randint(0,90)
def roll():                                #right
    return randint(0,9) * 10
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

3

Simulating a Deck of Cards

- A deck of cards is made up of 52 cards, where each card has a suit and a rank:
 - Suits: Spades (♠), Hearts (♥), Diamonds (♦), Clubs (♣)
 - Ranks: 2, 3, 4, 5, 6, 7, 8, 9, 10, J (Jack), Q (Queen), K (King), A (Ace)
- A standard deck of cards has 1 of each combination of suit and rank.



15110 Principles of Computing, Carnegie
Mellon University - CORTINA

4

A card deck in Python

Do not use
...
in your code!

```
def create_deck():
    deck = []
    ranklist = ["2","3","4", ... ,"K","A"]
    suitlist = ["clubs", ... , "spades"]
    for suit in range(0,4):
        for rank in range (0,13):
            card = []
            card.append(ranklist[rank])
            card.append(suitlist[suit])
            deck.append(card)
    return deck
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

5

Rank

```
def get_rank(card):
    ranklist = ["2", ... ,"A"]
    return
    ranklist.index(card[0])
```

Do not use
...
in your code!

card	rank
2	0
3	1
4	2
5	3
6	4
7	5
8	6
9	7
10	8
J	9
Q	10
K	11
A	12

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

6

Suit

Do not use
...
in your code!

```
def get_suit(card):
    suitlist = ["clubs", ... , "spades"]
    return
        suitlist.index(card[1])
```

card	suit
♣ (clubs)	0
♦ (diamonds)	1
♥ (hearts)	2
♠ (spades)	3

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

7

Picking a random card from a deck

```
from random import randint

def pick_card(deck):
    card_number = randint(0,51)
    return deck[card_number]
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

8

Dealing Random Cards

- Suppose we have a card game like Poker where we want to be dealt a "hand" of 5 random cards from the deck.
- What is potentially wrong with the following code?

```
hand = []  
for i in range(0,5):  
    hand.append(pick_card(deck))
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

9

Shuffling the Deck

- We should shuffle a deck and then create a hand from the first 5 cards in the deck.
- There are many ways to shuffle a deck of cards.
- One algorithm:
 - Exchange (swap) the first card with a random card.
 - Exchange the second card with a random card except the first card.
 - Exchange the third card with a random card except the first two cards.
 - ... Repeat until all cards have been swapped.

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

10

Building the Function

- For the first card (at index 0) in deck **d**, how do we generate a random index for a card to swap?

```
r = randint(0, len(d) - 1)
```

- How do we swap the first card with the randomly-selected card?

```
temp = d[0]
```

```
d[0] = d[r]
```

```
d[r] = temp
```

or we can use ***parallel assignment*** in Python...

```
d[0], d[r] = d[r], d[0]
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

11

Building the Function (cont' d)

- For the second card (at index 1) in deck **d**, how do we generate a random index for any card except the first card?

```
r = randint(1, len(d) - 1)
```

- How do we swap the first card with the randomly-selected card?

```
temp = d[1]
```

```
d[1] = d[r]
```

```
d[r] = temp
```

or we can use ***parallel assignment*** in Python...

```
d[1], d[r] = d[r], d[1]
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

12

Building the Function (cont' d)

- For the third card (at index 2) in deck **d**, how do we generate a random index for any card except the first two cards?

```
r = randint(2, len(d) - 1)
```

- How do we swap the first card with the randomly-selected card?

```
temp = d[2]
```

```
d[2] = d[r]
```

```
d[r] = temp
```

or we can use *parallel assignment* in Python...

```
d[2], d[r] = d[r], d[2]
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

13

In general...

- For the card at index **i** in deck **d**, how do we generate a random index for a card to swap?

```
r = randint(i, len(d) - 1)
```

- How do we swap the first card with the randomly-selected card?

```
temp = d[i]
```

```
d[i] = d[r]
```

```
d[r] = temp
```

or we can use *parallel assignment* in Python...

```
d[i], d[r] = d[r], d[i]
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

14

Shuffling the entire deck and dealing five cards...

```
def permute(deck):
    for i in range(0, len(deck)-1):
        r = randint(i, len(deck)-1)
        deck[i], deck[r] = deck[r], deck[i]
    return deck

>> hand = permute(deck)[0:5]
[ ['10', 'hearts'], ['10', 'spades'],
  ['J', 'spades'], ['4', 'clubs'], ['Q', 'spades'] ]
```

You can also use `random.shuffle(deck)` to permute!

Poker: Detecting a Flush

- In poker, a flush is a hand where all of the cards have the same suit.
- One possible algorithm:
 - If all of the cards have a suit of spades, return true.
 - If all of the cards have a suit of hearts, return true.
 - If all of the cards have a suit of diamonds, return true.
 - If all of the cards have a suit of clubs, return true.
 - If none of the above tests returns true, return false.

Poker: Detecting a Flush (cont' d)

```
def all_spades(hand):
    for i in range(0, len(hand)):
        if (hand[i].get_suit() != 3):
            return False
    return True
```

card	suit
♣ (clubs)	0
♦ (diamonds)	1
♥ (hearts)	2
♠ (spades)	3

`all_hearts`, `all_diamonds`
and `all_clubs` are written similarly.

Poker: Detecting a Flush (cont' d)

```
def flush(hand):
    if all_spades(hand):
        return True
    if all_hearts(hand):
        return True
    if all_diamonds(hand):
        return True
    if all_clubs(hand):
        return True
    return False
```

equivalent to:
`if all_spades(hand) == True:`

Poker: Detecting a Flush (Another way)

```
def flush2(hand)
    for j in range(0,4): # j is suit index
        count = 0        # reset for suit j
        for i in range(0,len(hand)):
            if (hand[i].get_suit() == j):
                count = count + 1
        if count == len(hand):
            return True    # all suits were j
    return False
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

19

Simple dice game

- A player has two die. On each roll, if the player does not roll “doubles” (same value on each die), then the player wins the sum of the die values. Otherwise, the player earns a “strike”. The game ends once the player has three strikes.
- Write a function that returns the amount the player wins in a simulated simple dice game.

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

20

Rolling a die

```
def roll():  
    return randint(1,6)
```

One round of the game

```
die1 = roll()  
die2 = roll()  
if die1 == die2:  
    strikes = strikes + 1  
else:  
    sum = sum + die1 + die2
```

Putting it together

```
def simple_game():
    strikes = 0
    sum = 0
    while (strikes < 3):
        die1 = roll()
        die2 = roll()
        if die1 == die2:
            strikes = strikes + 1
        else:
            sum = sum + die1 + die2
    return sum
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

23

What is the average winnings for 1000 players of this game?

```
>>> games = []
>>> for i in range(0,1000):
...     games.append(simple_game())
...
>>> games
[16, 60, 252, 131, 70, ... , 209, 70, 107]
>>> total = 0
>>> for score in games:
...     total = total + score
...
>> total/1000
=> 102.674
```

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

24