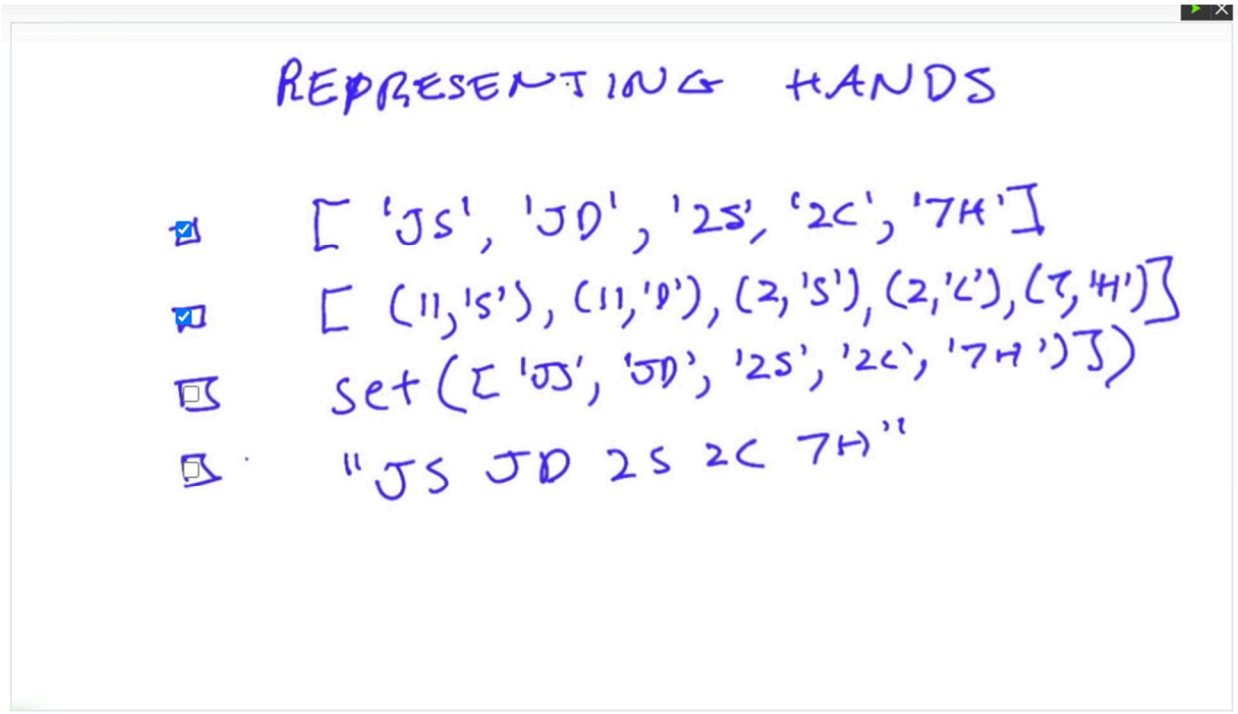
Representing Hands

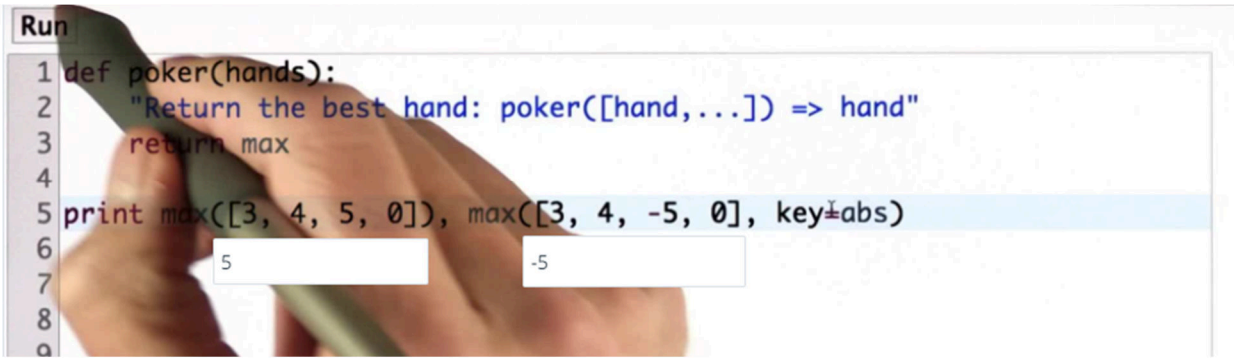


Poker Function

Run

```
def poker(hands):
    "Return the best hand: poker([hand,...]) => hand"
    return max
```

Understanding Max



Using Max

```
def poker(hands):
    "Return the best hand: poker([hand,...]) => hand"
    return max(hands,key=hand_rank)
```

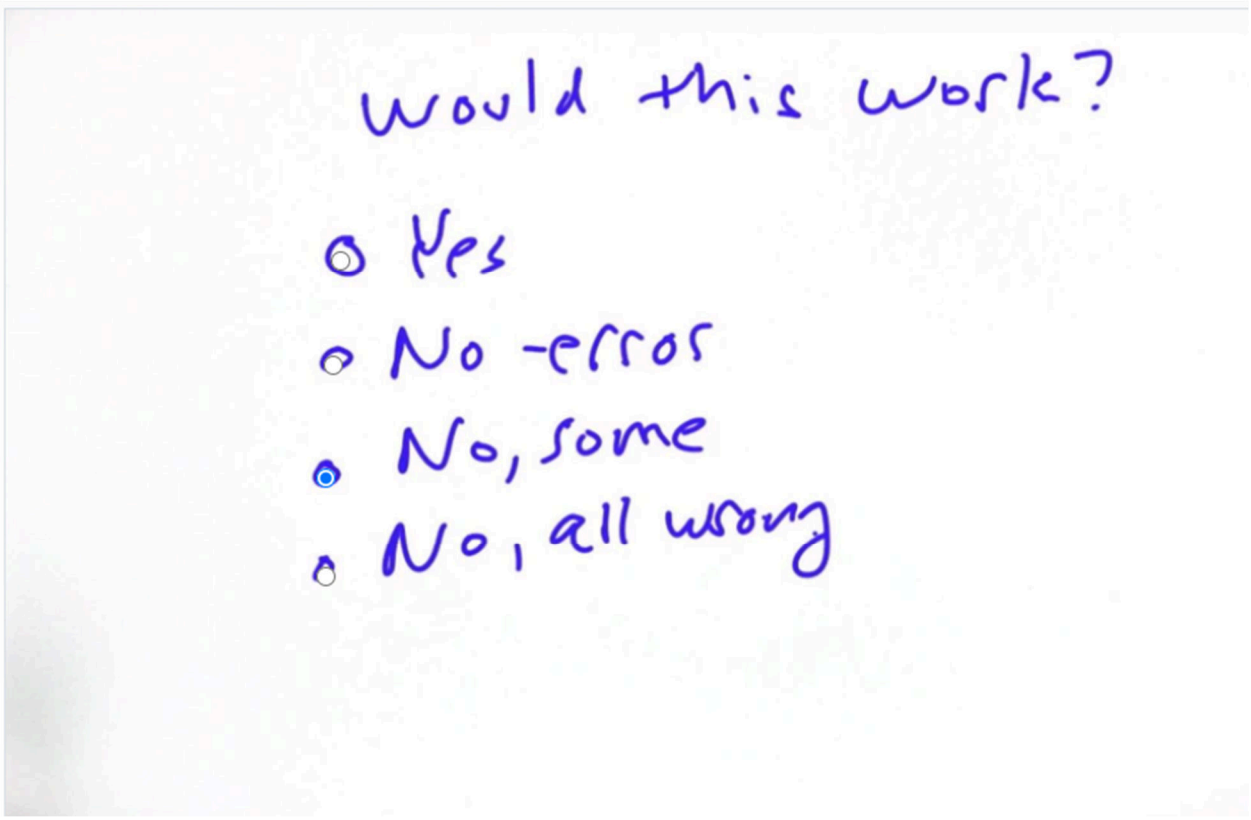
Testing

```
def test():
    "Test cases for the functions in poker program"
    sf = "6C 7C 8C 9C TC".split() # => ['6C', '7C', '8(
    fk = "9D 9H 9S 9C 7D".split()
    fh = "TD TC TH 7C 7D".split()
    assert poker([sf, fk, fh]) == sf
    assert poker([fk,fh])==fk
    assert poker([fh,fh])==fh
    return "test pass"
```

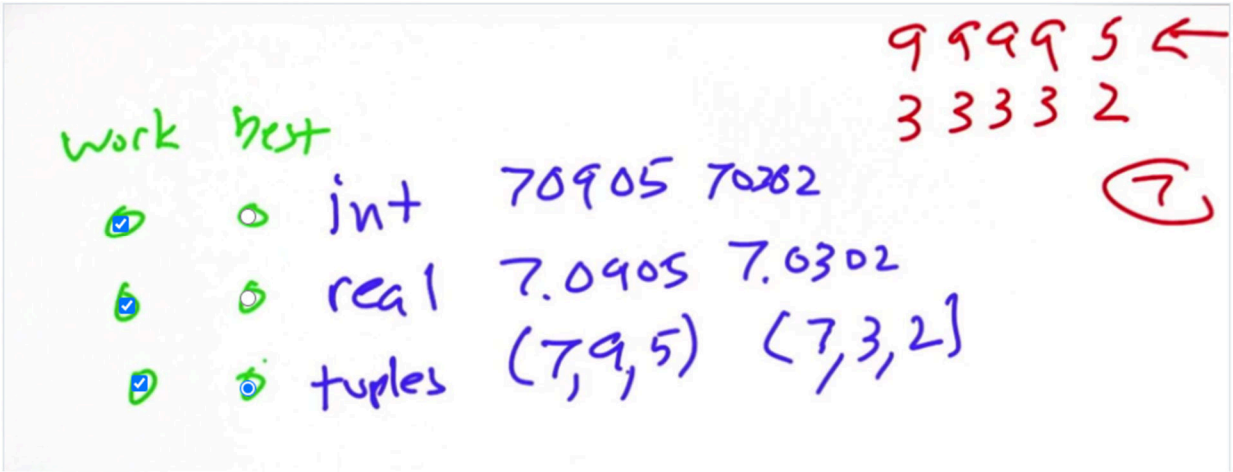
Extreme Values

```
def test():
    "Test cases for the functions in poker program"
    sf = "6C 7C 8C 9C TC".split()
    fk = "9D 9H 9S 9C 7D".split()
    fh = "TD TC TH 7C 7D".split()
    assert poker([sf, fk, fh]) == sf
    assert poker([fk, fh]) == fk
    assert poker([fh, fh]) == fh
    assert poker([sf]) == sf
    assert poker([sf]+99*[fh]) == sf
    return "test pass"
```

Hand Rank Attempt



Representing Rank



Testing Hand Rank

```
def test():
    "Test cases for the functions in poker program"
    sf = "6C 7C 8C 9C TC".split() # Straight Flush
    fk = "9D 9H 9S 9C 7D".split() # Four of a Kind
    fh = "TD TC TH 7C 7D".split() # Full House
    assert poker([sf, fk, fh]) == sf
    assert poker([fk, fh]) == fk
    assert poker([fh, fh]) == fh
    assert poker([sf]) == sf
    assert poker([sf] + 99*[fh]) == sf
    assert hand_rank(sf) == (8,10)
    assert hand_rank(fk) == (7,9,7)
    assert hand_rank(fh) == (6,10,7)
print test()
```

Writing Hand Rank

```
def hand_rank(hand):
    ranks = card_ranks(hand)
    if straight(ranks) and flush(hand):            # straight flush
        return (8, max(ranks))
    elif kind(4, ranks):                           # 4 of a kind
        return (7, kind(4, ranks), kind(1, ranks))
    elif kind(3, ranks) and kind(2, ranks):        # full house
        return (6, kind(3, ranks), kind(2, ranks))
    elif flush(hand):                              # flush
        return (5, ranks)
    elif straight(ranks):                          # straight
        return (4, max(ranks))
    elif kind(3, ranks):                           # 3 of a kind
        return (3, kind(3, ranks),ranks)
    elif two_pair(ranks):                          # 2 pair
        return (2, two_pair(ranks),ranks)
    elif kind(2, ranks):                           # kind
        return (1, kind(2, ranks),ranks)
    else:                                          # high card
        return (0, ranks)
```

Testing Card Rank

```
def test():
    "Test cases for the functions in poker program"
    sf = "6C 7C 8C 9C TC".split() # Straight Flush
    fk = "9D 9H 9S 9C 7D".split() # Four of a Kind
    fh = "TD TC TH 7C 7D".split() # Full House
    assert card_ranks(sf)==[10,9,8,7,6]
    assert card_ranks(fk)==[9,9,9,9,7]
    assert card_ranks(fh)==[10,10,10,7,7]
    assert poker([sf, fk, fh]) == sf
    assert poker([fk, fh]) == fk
    assert poker([fh, fh]) == fh
    assert poker([sf]) == sf
    assert poker([sf] + 99*[fh]) == sf
    assert hand_rank(sf) == (8, 10)
    assert hand_rank(fk) == (7, 9, 7)
    assert hand_rank(fh) == (6, 10, 7)
    return 'tests pass'
```

Fixing Card Rank

```
def card_ranks(hand):
    "Return a list of the ranks, sorted with a higher first."
    ranks = ['--23456789TJQKA'.index(r) for r,s in hand]
    ranks.sort(reverse=True)
    return ranks

print card_ranks(['AC', '3D', '4S', 'KH']) #should output [14, 13, 4, 3]
```

Straight And Flush

```
def straight(ranks):
    "Return True if the ordered ranks form a 5-card straight."
    return (max(ranks)-min(ranks) == 4) and len(set(ranks))==5

def flush(hand):
    "Return True if all the cards have the same suit."
    suits = [s for r,s in hand]
    return len(set(suits))== 1
```

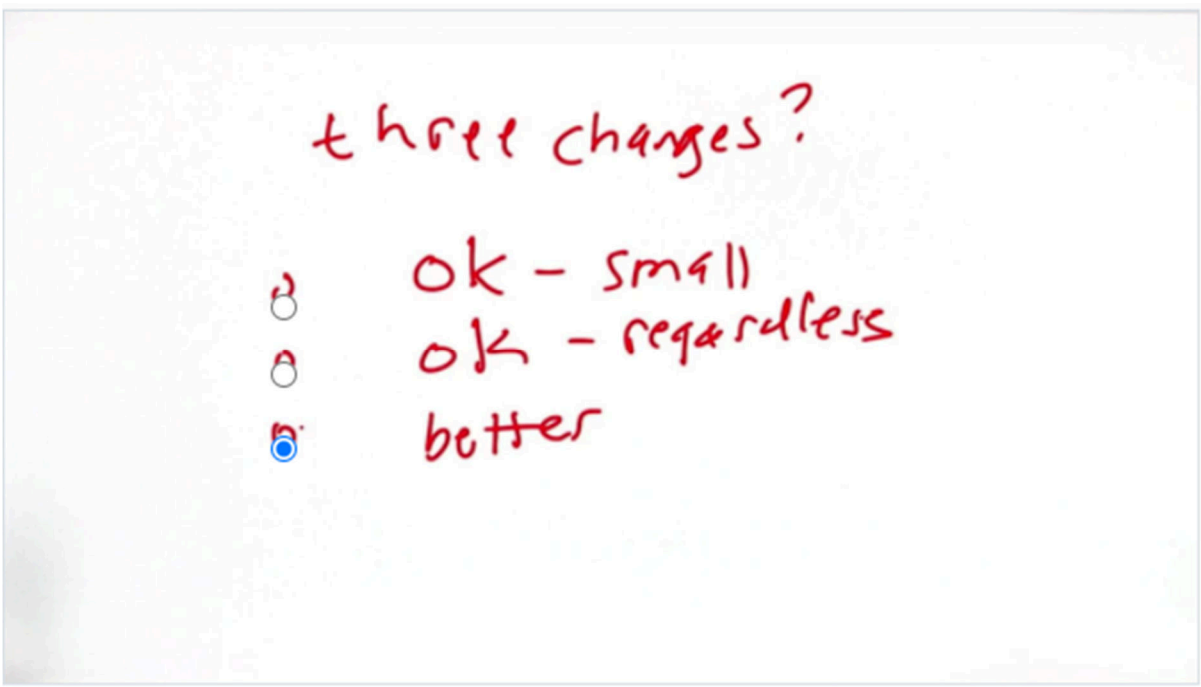
Kind function

```python
def kind(n, ranks):
    """Return the first rank that this hand has exactly n of.
    Return None if there is no n-of-a-kind in the hand."""
    for r in ranks:
        if ranks.count(r) == n: return r
    return None
```

Two Pair Function

```python
def two_pair(ranks):
    """If there are two pair, return the two ranks as a
    tuple: (highest, lowest); otherwise return None."""
    pair=kind(2,ranks)
    lowpair=kind(2,list(reversed(ranks)))
    if pair and lowpair != pair:
        return (pair, lowpair)
    else:
        return None
```

Making Changes

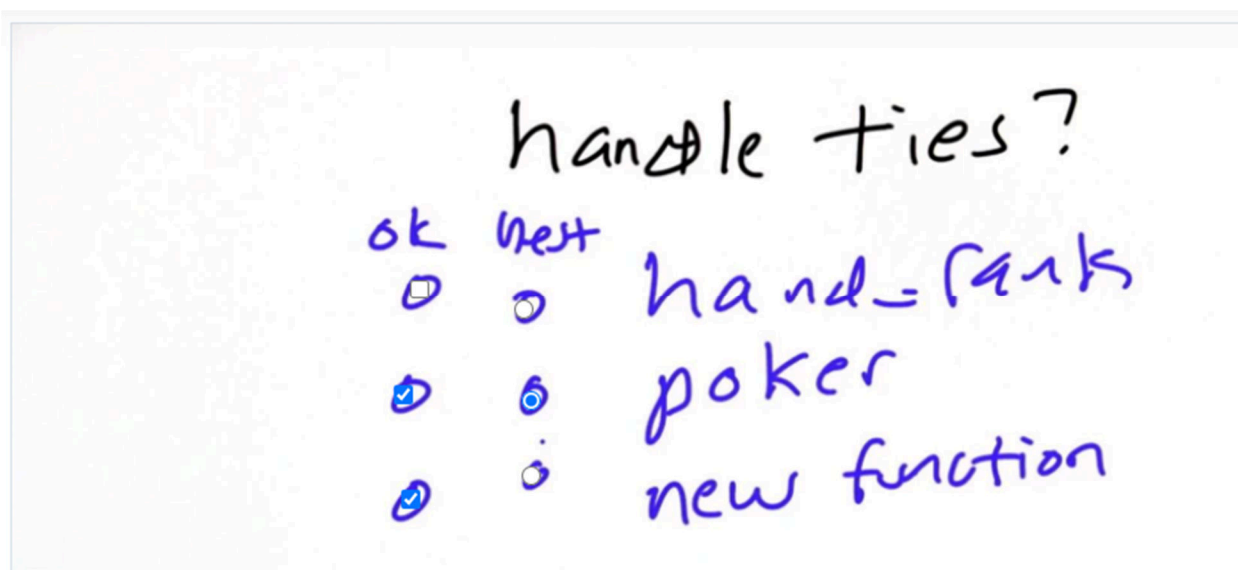


What to Change

Ace Low Straight

```
def card_ranks(hand):
    "Return a list of the ranks, sorted with higher first."
    ranks = ['--23456789TJQKA'.index(r) for r, s in hand]
    ranks.sort(reverse = True)
    return [5,4,3,2,1] if (ranks==[14,5,4,3,2]) else ranks
```

Handling Ties



Allmax

```python
def allmax(iterable, key=None):
    "Return a list of all items equal to the max of the iterable."
    result, maxval =[], None
    key = key or (lambda x: x)
    for x in iterable:
        xval = key(x)
    if not result or xval > maxval:
        result, maxval = [x], xval
    elif xval == maxval:
        result.append(x)
    return result
```

Deal

```python
def deal(numhands, n=5, deck=mydeck):
    random.shuffle(deck)
    return [deck[n*i:n*(i+1)] for i in range(numhands)]
```

Hand Frequencies