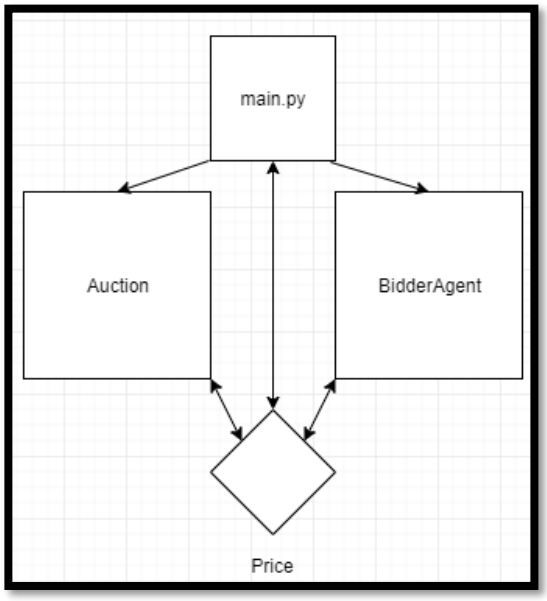
UNIVERSITY OF ABERDEEN

# COMBINED-AUCTION

- Document detailing the development of prototype solution for combined auctions.

## SYSTEM DESIGN:




The original intention for the design of this prototype was to have 2 objects and 1 file that calls their functions. The objects would be each agent; bidder, auctioneer. These objects would have various functions that would be called from the main python file.

The program was written in pure python with the goal of simplicity and only used minimalist libraries such as random, decimal and date-time.

Due to challenges in development a 4th object was added to store item price that could be accessed by all objects.

## MAIN.PY:

This file is simply the controller of the program that the user calls to execute the program. It lists in order the functions that should flow the user through the program and allow the agents to communicate and automate the auction. The main file imported all other objects; agents and the price variable class.

```
# Welcome the user
e.introduction()

# Set starting price!
e.setPrice()

# Generate ticket
e.cast_ticketInfo()

# Ask user to confirm they are happy with the parameters
# If yes, begin auction, else, exit program
answer = raw_input("Being the auction?: [y/n]: ")
print (type(answer))
if not answer or answer[0].lower() != 'y':
    print("You did not indicate approval")
    exit(1)
else:
    a.setBudget()
    print ("Agent budget set to:", Budget)
    e.englishAuction()
```

UNIVERSITY OF ABERDEEN

# BIDDERAGENT.PY:

This class was designed to act as the bidding agent. The class was intended to be a shell, to list the class creating multiple agents to bid in the auction all with the same design. Each agent would have 2 main functions: Set Budget, the agent randomly generates a budget to bid with based on the initial price of the ticket. This budget is a random range, so each agent has a different budget. And finally, the bid function, each agent would only bid if the current asking price was equal to or less than their budget. The agent with the highest budget would win.

```
def setBudget(self):
    global Budget
    Budget = v.initialPrice * random.randint(155, 389)/100

def bid(self):
    global amountOfBids
    global currentBid
    global Budget
    if Budget <= v.initialPrice:
        currentBid = 1
        if currentBid == 1:
            amountOfBids += 1
        print (" ")
        print ("Agent: I shall bid this round!")
        print (" ")
    else:
        currentBid = 0
        if currentBid == 0:
            amountOfBids -=1
        ("is out!")
```

```
def englishAuction(self):
    global auctionRound
    print ("<English Auction>")
    print ("Auctioneer: OPEN-CRY! ticket ID: ", ticket_id, "starting price at ", v.initialPrice)
    # While there is no winner for the auction
    while currentBid == 1:
        if auctionWinner == False:
            # New round!
            auctionRound += 1
            # Announce round and allow bids!
            print ("Auctioneer: Round ", auctionRound,"do I have", v.initialPrice,"?")
            # Agent msakes bid if within budget
            b.bid()
            # Initate round timer
            run = auctionWinner
            mins = 0
            # Only run if the user types in "start"
            if run == False:
                # Loop until we reach 4 seconds running
                while mins !=4:
                    print (">>>>>>>>>>>>>>>>>>>>", mins)
                    # Sleep for a minute
                    time.sleep(1)
                    # Increment the minute total
                    mins += 1
            if currentBid == 1:
                increasePrice + v.initialPrice
            #ea.checkForWinner()
        else:
            print ("The auction has concluded!")
```

# AUCTION.PY

Acting as the auctioneer, the auctioneer was designed to have 3 key functions. English Auction, Dutch Auction and Timer. Both auction functions would mimic each of their respected styles of auction and would both lye in the timer auction. The timer auction acted as its own ticket, designed to sell a ticket with 5 ticks (days) on the 3rd ticket the auction would switch to Dutch if no winner was found.

## Bugs & Shortcomings:

Due to my lack of skill and understanding in coding I struggled to complete the prototype. The main challenge I faced was accessing and storing data in variables across classes. Despite this I attempted to stay true to my original design as possible, allowing the bidder to communicate to the auctioneer through the bid variable, however, I was unable to write a timer function in time for both auction functions. Though I made both auction functions and was unable to do final testing. My hope is the basic structure of the program is still apparent even in its buggy design.