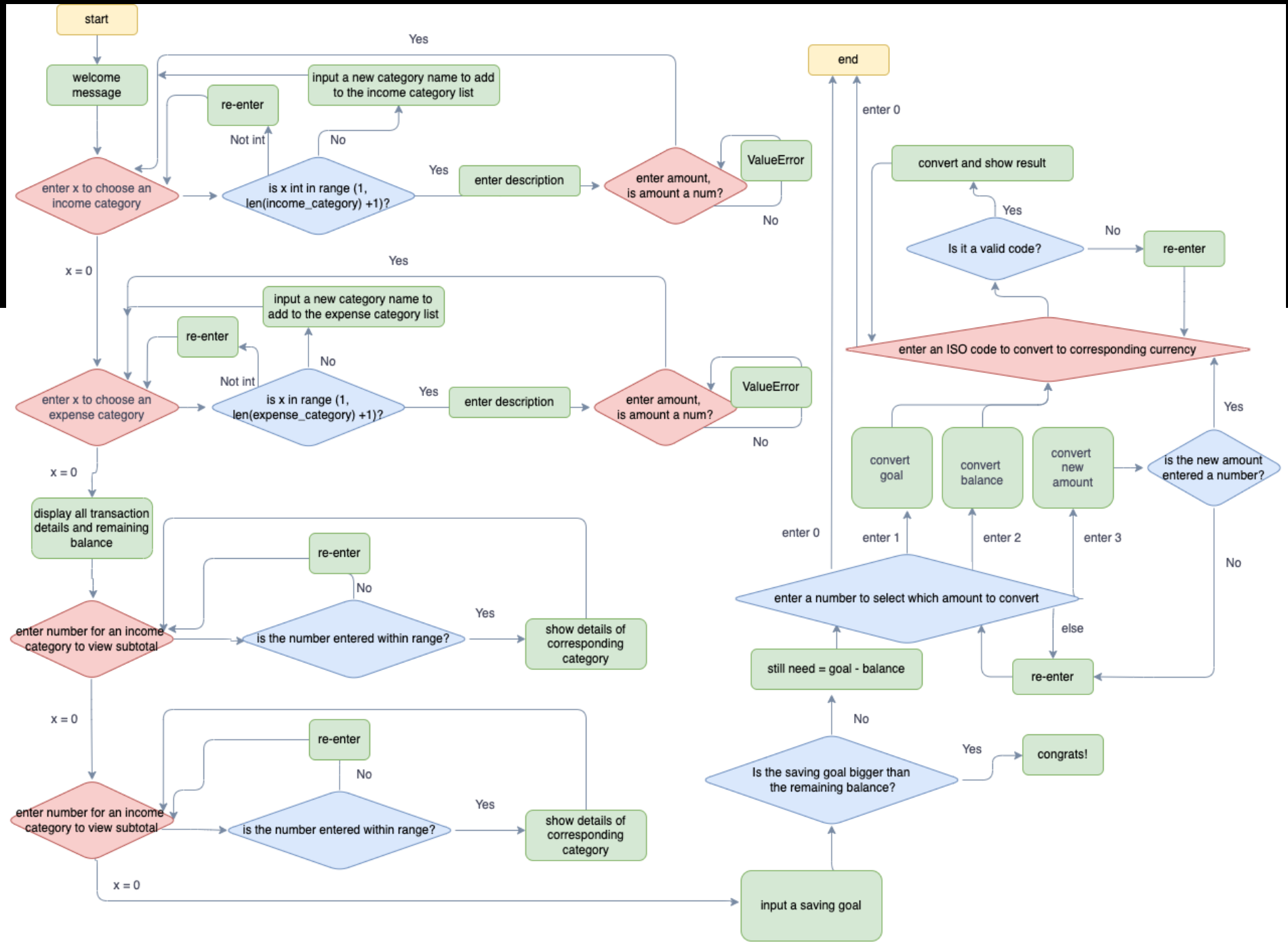
T1A3- TERMINAL APP

Dan Mo

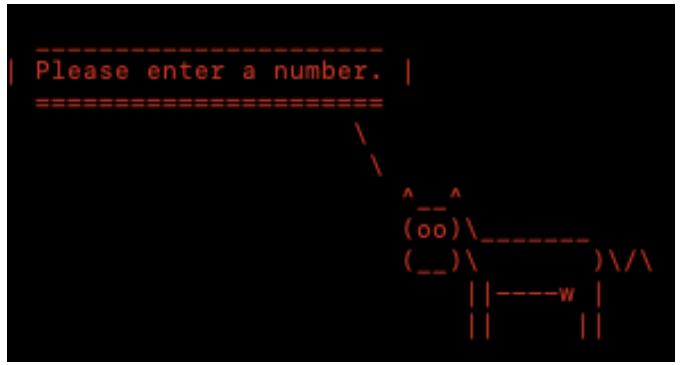
BUDGET CALCULATOR

1. Track incomes and expenses in different categories
2. Create new categories
3. Set up saving goal and see if the goal is achieved
4. Convert an amount of Australian Dollar into other currencies


start
welcome message
enter x to choose an income category
re-enter
Not int
input a new category name to add to the income category list
No
is x int in range (1, len(income_category) +1)?
Yes
enter description
enter amount, is amount a num?
ValueError
No
Yes
x = 0
enter x to choose an expense category
input a new category name to add to the expense category list
re-enter
Not int
No
is x in range (1, len(expense_category) +1)?
Yes
enter description
enter amount, is amount a num?
ValueError
No
Yes
x = 0
display all transaction details and remaining balance
enter number for an income category to view subtotal
re-enter
No
is the number entered within range?
Yes
show details of corresponding category
x = 0
enter number for an income category to view subtotal
re-enter
No
is the number entered within range?
Yes
show details of corresponding category
x = 0
input a saving goal
Is the saving goal bigger than the remaining balance?
Yes
congrats!
No
still need = goal - balance
enter a number to select which amount to convert
enter 0
enter 1
enter 2
enter 3
else
re-enter
convert goal
convert balance
convert new amount
is the new amount entered a number?
Yes
No
enter an ISO code to convert to corresponding currency
Is it a valid code?
No
re-enter
Yes
convert and show result
enter 0
end

1. TRACK TRANSACTIONS

```
2 Investment
3 Gifts
4 Lottery

Enter an integer number to add an income in one of the above categories.
Enter any integer other than 0 or the ones listed to customise a new category.
Enter 0 to move on when finish adding all income transactions. 1
Enter the income description: weekly wage
Enter the income amount: 1050
```



- In loops:
  - Display a list of income/expense categories
  - User choose an income/expense category
  - Input transaction details (description, amount)
- Enter 0 to break the loops and move on.
- Choose category and enter amount: raise ValueError

1. TRACK TRANSACTIONS

```
EXPENSES

1 Housing
2 Food
3 Transportation
4 Entertainment
5 Medical

Enter an integer number to add an expense in one of the above categories.
Enter any integer other than 0 or the ones listed to customise a new category.
Enter 0 to move on when finish adding all expense transactions. 0
See below your transaction details:

--------------------Transaction Summary--------------------
Salary              weekly wage                    1050.00
Investment          ETFs                             20.00
Housing             weekly rent                    -160.00
Food                aldi                            -65.00
Food                asian supermarket               -50.00
Food                Victorian market                -23.00
Transportation      myki topup                      -50.00

Balance: 722.0
Date: 2022-09-25
```

- View all transaction details and the remaining balance

```
EXPENSES

1 Housing
2 Food
3 Transportation
4 Entertainment
5 Medical

Enter integer for a category to view subtotal. Enter 0 to skip: 2
-------------Food-------------
aldi                    -65.00
asian supermarket       -50.00
Victorian market        -23.00

Subtotal: -138.0
```

```
 ____________________
| Input out of range. |
 ====================
                    \
                     \
                      ^__^
                      (oo)\_______
                      (__)\       )\/\
                          ||----w |
                          ||     ||
```

```
 ______________________
| Input must be an integer. |
 ======================
                      \
                       \
                        ^__^
                        (oo)\_______
                        (__)\       )\/\
                            ||----w |
                            ||     ||
```

- Select categories to view category subtotal and details
- Choose category: raise ValueError, RangeError

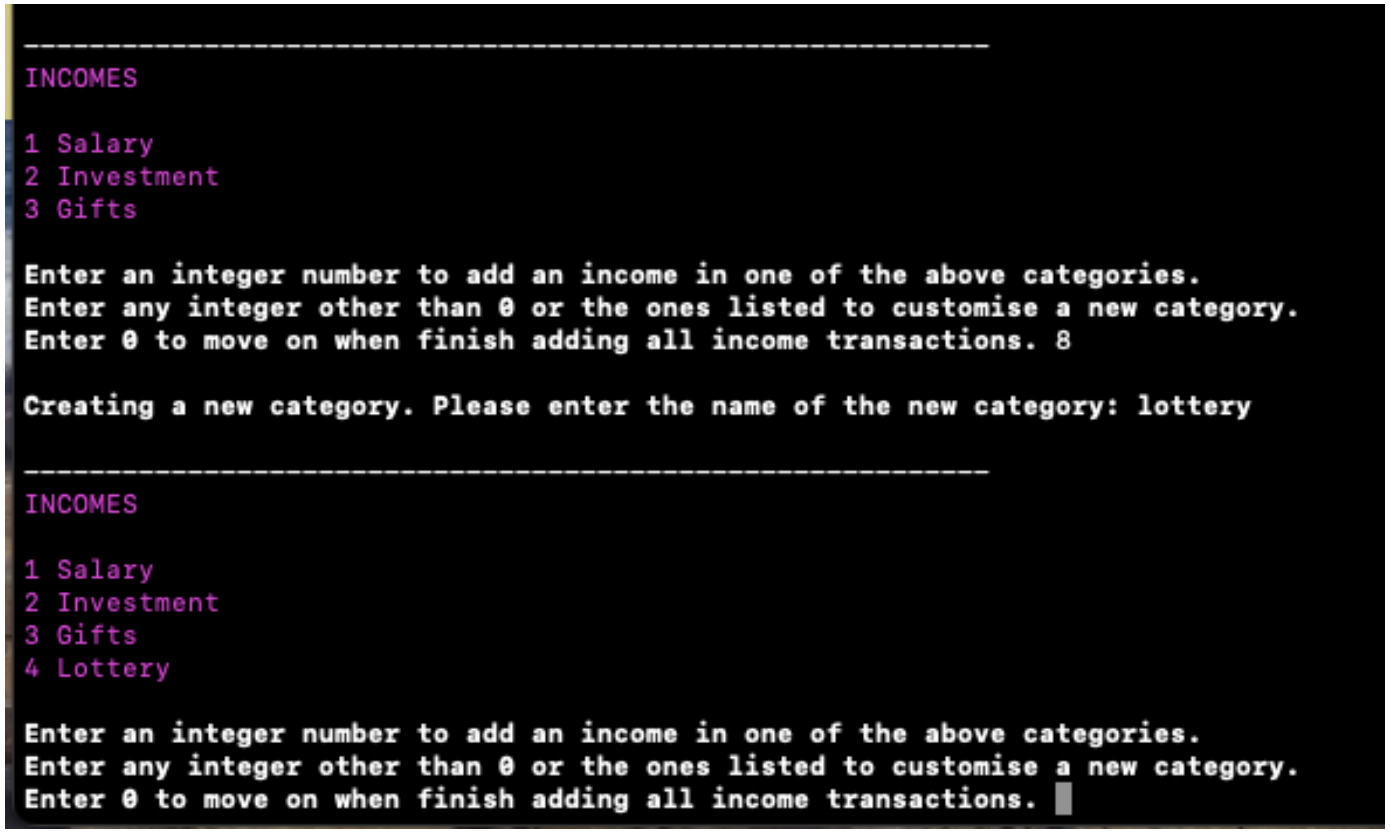
CODE

- Every transaction users enter is a dictionary: {‘category’: category, ‘description’: description, ‘amount’: amount} appended to a list of details.
- Deposit for incomes, amount positive; withdraw for expenses, amount negative.
- Print all the dictionary values in the list to display the details. Add up all the amounts for the remaining balance.

```
class Transactions:
    def __init__(self, category, description, amount):
        self.category = category
        self.description = description
        self.amount = amount
        self.details = details

    def deposit(self):
        self.details.append({'category': self.category,
        'description': self.description, 'amount': self.amount})

    def withdraw(self):
        self.details.append({'category': self.category,
        'description': self.description, 'amount': -self.
        amount})
```

```
def show_details():
    print(Style.BRIGHT + 'See below your transaction details:')
    print(Style.RESET_ALL)

    title = "-"*20 + "Transaction Summary" + "-"*20 + "\n"
    items = ""
    total = 0
    date_printed = date.today()
    for item in details:
        items += f"{item['category'][0:20]:20}" + f"{item
        ['description']:30}" + f"{item['amount']:>9.2f}" + "\n"
        total += item['amount']

    output = title + items + '\nBalance: ' + str(total) +
    '\nDate: ' + str(date_printed) + '\n'
    print(Fore.CYAN + output)
    print(Fore.RESET)
```

2. CUSTOMIZE NEW CATEGORY



- If users enter an integer not listed, the app creates a new income/expense category
- The new category is added to the list with a corresponding number to select it

CODE

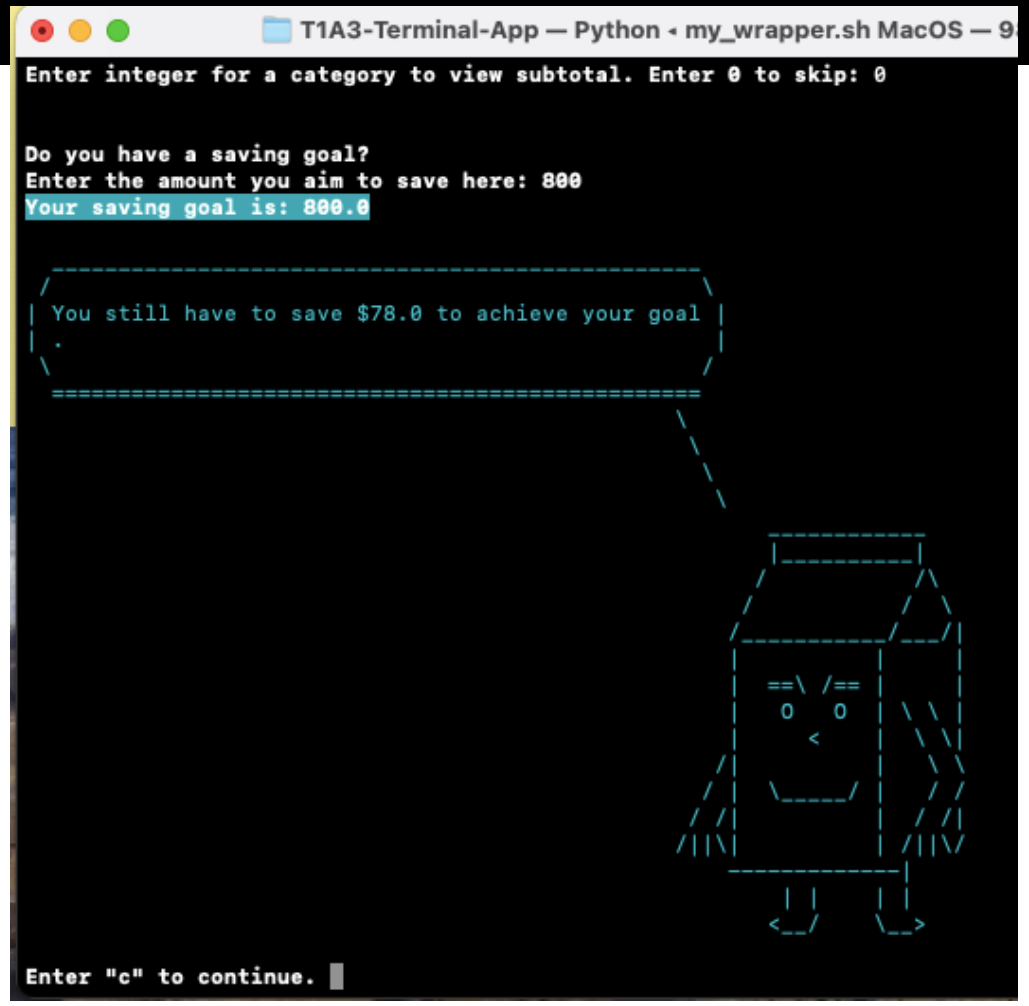
- The if and elif statements controls how the app responds to user input in the process of adding transactions details.

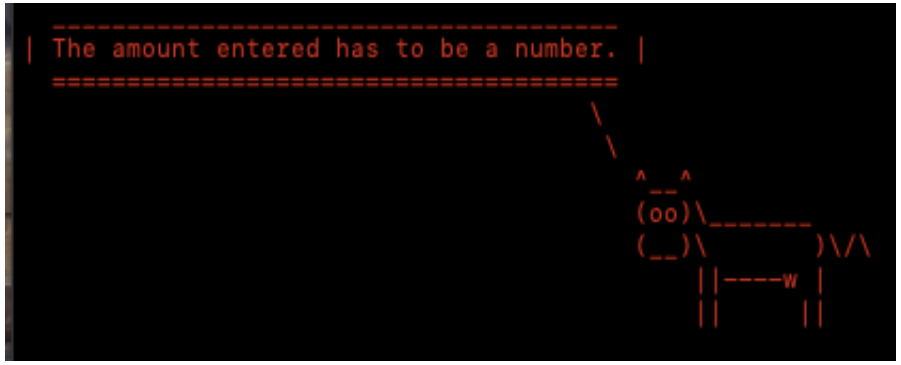
- Income & expense categories are stored in lists. When the input integer is out of the range of the category list, the method appends the new category name to the list

```
def new_category(self):
    new = input(Style.BRIGHT + '\nCreating a new category.
    Please enter the name of the new category: ')
    print(Style.RESET_ALL)
    return self.category_list.append(new.capitalize())
```

```
try:
    x = instruction(section)

    if x in range (1, len(income_or_expense)+1):
        category = income_or_expense[x-1]
        description = input(f'Enter the {section}
        description: ')

        while True:
            try:
                amount = float(input(f'Enter the
                {section} amount: '))
                break
            except ValueError:
                print(Fore.RED)
                cowsay.cow('Please enter a number.')
                print(Fore.RESET)

        deposit_or_withdraw(Transactions(category,
        description, amount))

    elif x == 0:
        break

    elif x < 0 or x >= len(income_or_expense)+1:
        display.new_category()

except ValueError:
    print(Fore.RED)
    cowsay.cow('Please enter an integer. ')
    print(Fore.RESET)
```

3. SET UP SAVING GOAL & COMPARE





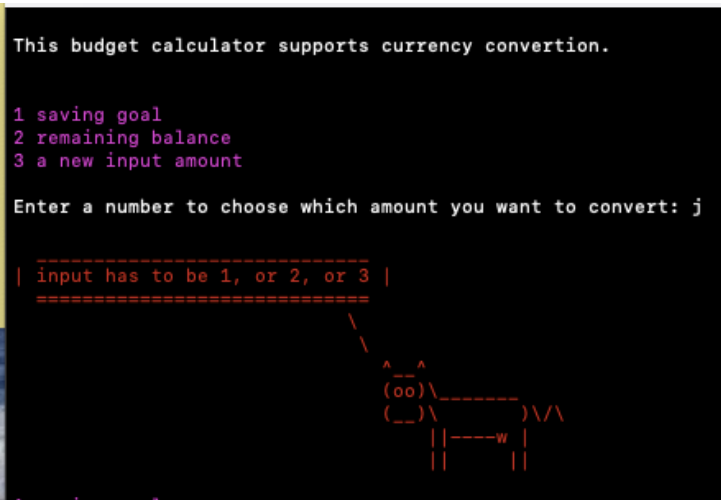
- User input an amount to set a saving goal
- Enter saving goal: raise ValueError
- App compares the saving goal and the remaining balance to see if the saving goal is achieved

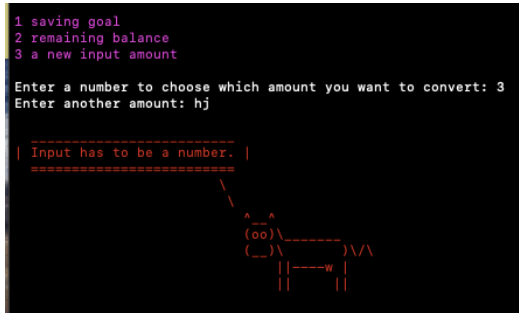
CODE

- Prompt user to input the amount and print the goal.
- If-else statement to control the output message.

```
class Comparison:
    def __init__(self, remain):
        self.remain = remain
        self.goal = 0

    def input_goal(self):
        print(Fore.RESET)
        self.goal = float(input(Style.BRIGHT + '\nDo you have a
        saving goal?\nEnter the amount you aim to save here: '))
        print(Back.CYAN + f'Your saving goal is: {self.goal}' +
        Back.RESET + Style.RESET_ALL)
        return self.goal

    def compare_goal(self):
        print(Fore.CYAN)
        if self.remain >= self.goal:
            outcome = cowsay.milk('Congratulation, you\'ve
            achieved your saving goal!')
        else:
            outcome = cowsay.milk(f'You still have to save ${self.
            goal-self.remain} to achieve your goal.')
        print(Fore.RESET)
        return outcome
```

4. CONVERT CURRENCY





- Users select which amount they want to convert (balance, goal, a new input amount)
- Select amount: raise InputError
- Enter new amount: raise ValueError
- User enter the ISO currency code to convert the amount
- Input code: raise CodeError

CODE

```
def receive(choice_1, choice_2):
    print(Fore.MAGENTA)
    for count, items in enumerate(option):
        print(count+1, items)
    print(Fore.RESET)
    user_input = input('Enter a number to choose which amount you
    want to convert: ')
    if user_input == '1':
        user_choice = choice_1
    elif user_input == '2':
        user_choice = choice_2
    elif user_input == '3':
        user_choice = float(input('Enter another amount: '))
    else:
        print(Fore.RED)
        raise InputError(cowsay.get_output_string('cow', 'input
        has to be 1, or 2, or 3'))
    return user_choice
```

- The if and elif statements controls the value assign to ‘user_choice’, which is passed to the next function and determines the amount of the conversion.

```
def exchange(user_choice):
    for key, value in currency_types.items():
        print(Fore.MAGENTA + f'The code for {value} is {key}')
        print(Fore.RESET)

    while True:
        select_currency = input(Style.BRIGHT + '\nEnter a
        currency code to convert Australian Dollar into another
        currency (See examples of some codes above).\nEnter 0 to
        exit and terminate the app. ')
        print(Style.RESET_ALL)

        if select_currency in list(currency_types.keys()):
            converted = convert(base='AUD', amount=user_choice,
            to=[select_currency])
            for key, value in converted.items():
                print(Style.BRIGHT)
                print(Back.CYAN + f'\n{user_choice} AUD is equal
                to {round(value, 1)} {key}' + Back.RESET)
            print(Style.RESET_ALL)

        elif select_currency == '0':
            break
        else:
            print(Fore.RED)
            raise CodeError(cowsay.get_output_string('cow',
            'Invalid code, please try again.'))
```

- Use convert(base, amount, to) function imported from the currency converter package.

DEMO THE APP

REVIEW

- Stressful but rewarding process
  - Challenging: deciding classes, attributes, methods; putting classes and functions in different modules
  - Deepened understanding in OOP
  - Got more familiar with importing packages and using different functions
  - Learnt how to write bash script