Python Advance

Lesson 2

Revisit

Data Collections

- List [ ] ordered, changeable, duplicates
- Tuple ( ) ordered, duplicates
- Set { } changeable
- Dictionary { : } ordered, changeable

Quiz – Data Types

https://pynative.com/python-variables-and-data-types-quiz/

Quick Function Review

Function do one job. 4 elements:

1. Define it with **def**
2. Give inputs (**parameters**)
3. Run the **task**
4. Return values

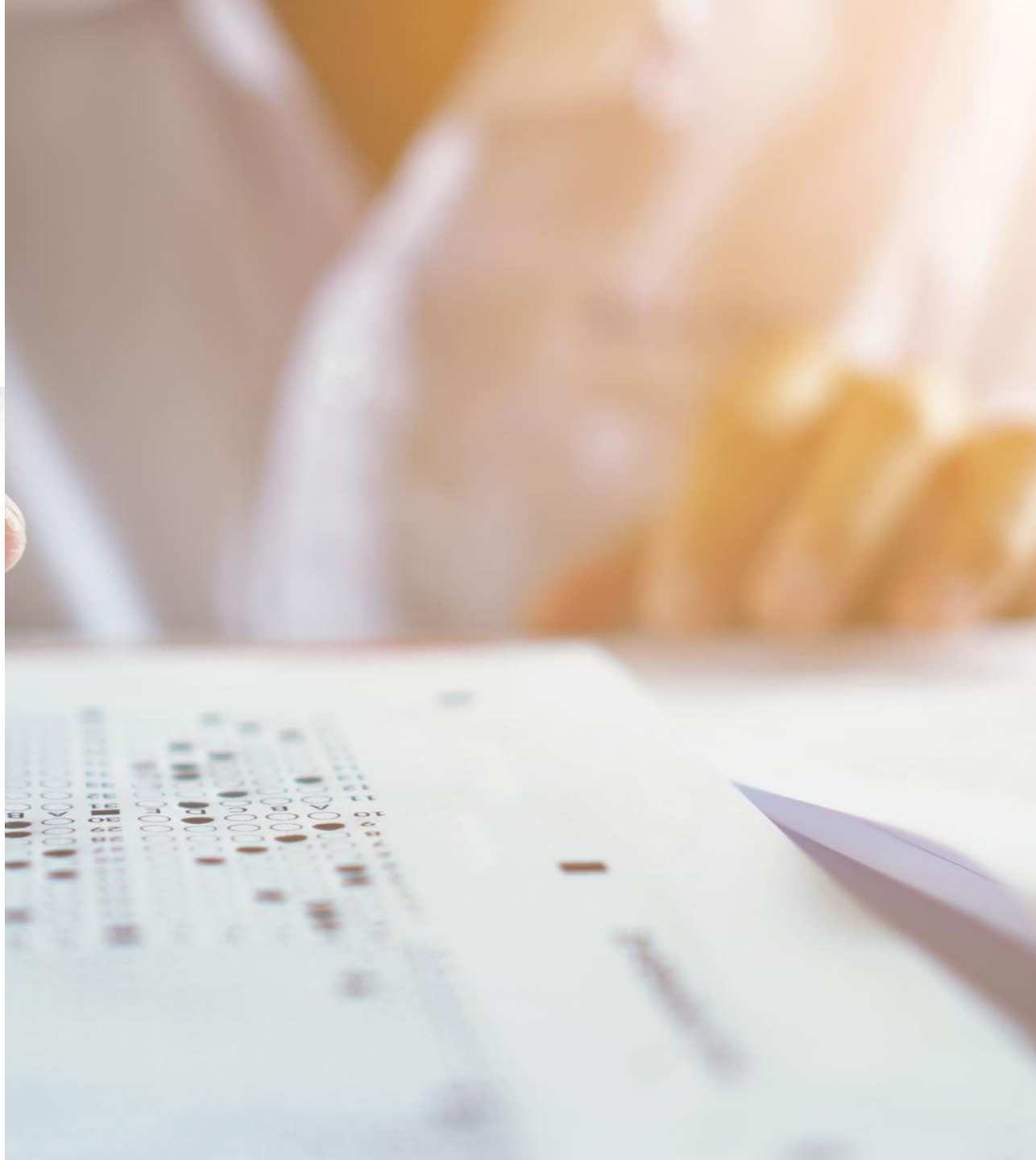
```
def sum_up(a, b):
    return a * b

print(sum_up(3, 5))
print(sum_up(2, 4))
print(sum_up(8, 12))
```

Software Design

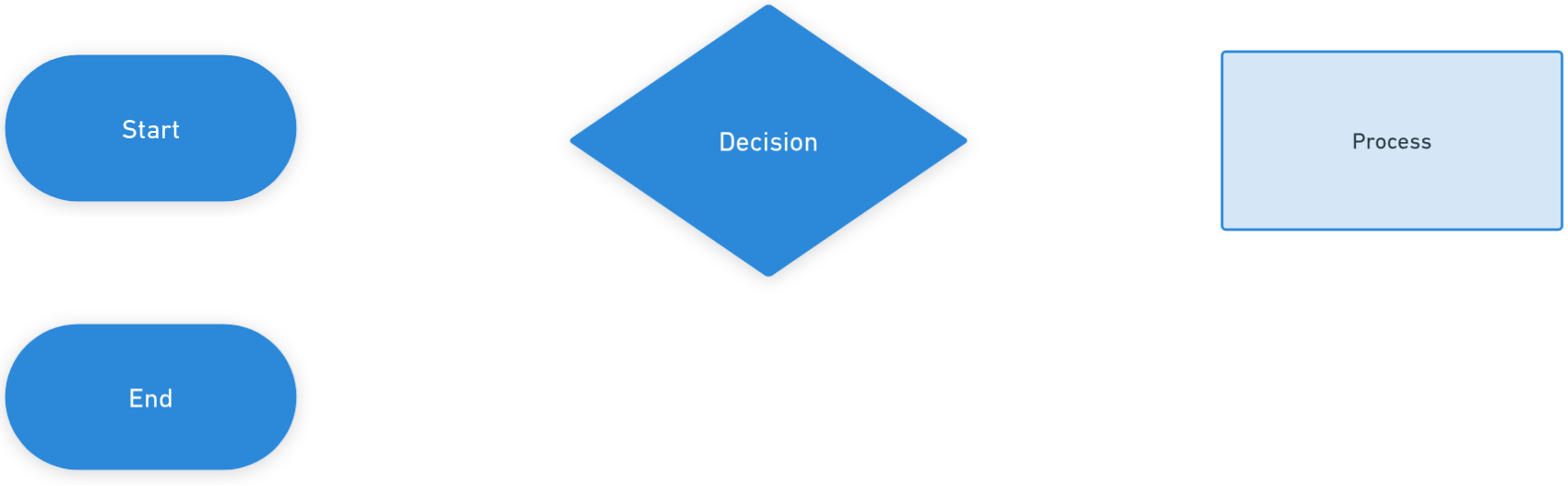
- Plan Before You Code - create a flow chart
- Break Problem into Small Pieces - separate functions
- Think About the User - show clear messages

Create a Flowchart

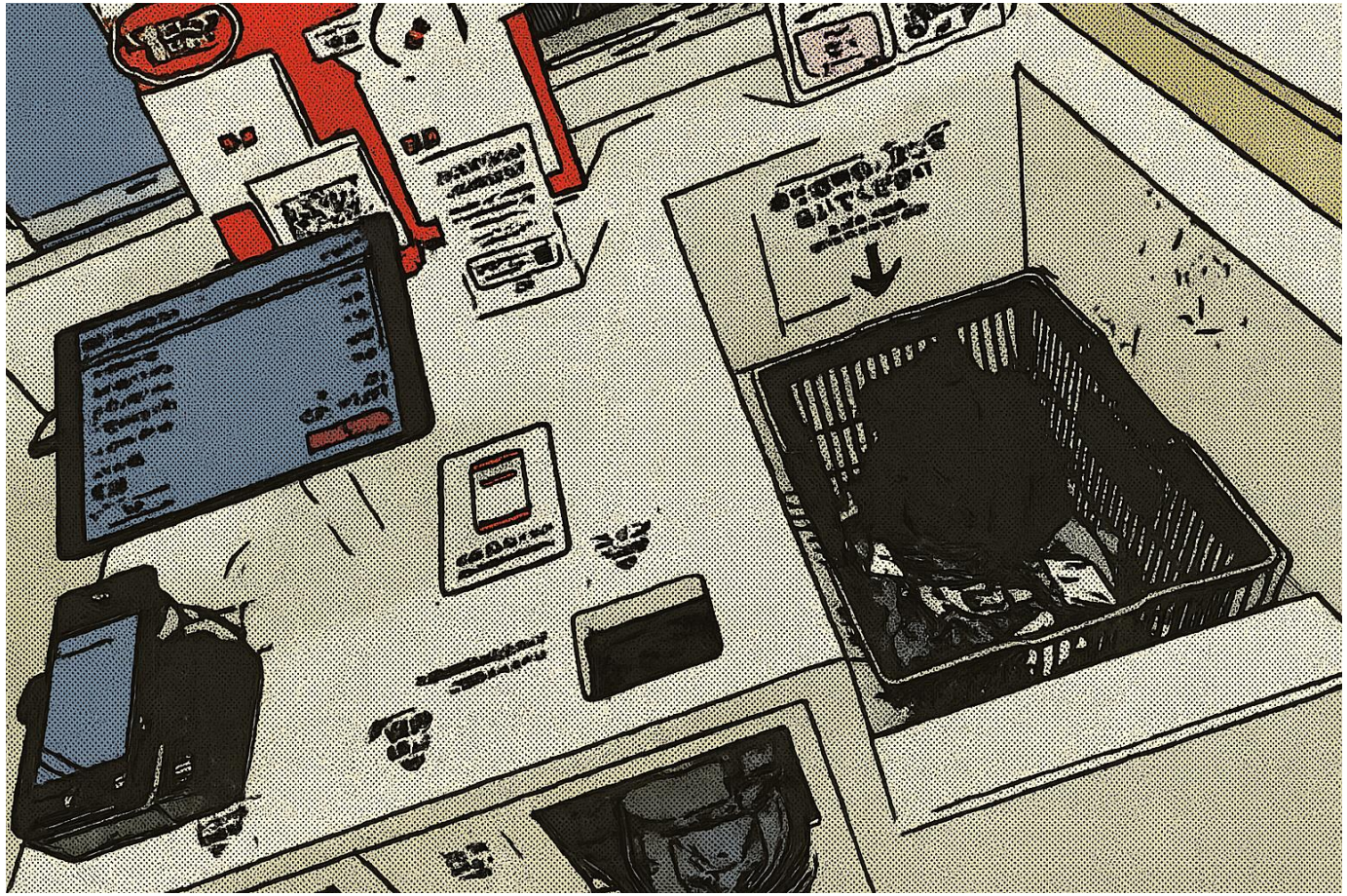
- DECISION: Do I have homework due? If yes, complete homework.
- DECISION: Are parents home? If no, call parents.
- Ask permission
- DECISION: Did they say yes? If no, stay home
- DECISION: Did my friend say yes? If no, stay home
- Go Out



Flow Chart – Basic Shaps



Design a Kiosk Checkout Program



Grocery Store Kiosk Checkout Process

- Start: User begins checkout.
- Process: Scan item (enter item name and price).
- Decision: Is the item valid?
- Process: Find the price and add item to list.
- Decision: More items to scan?
- Process: Calculate final total.
- Decision: Payment method (cash or card)?
- Process: Process payment.
- Decision: Is payment sufficient?
- Process: Issue receipt.
- End: Checkout complete.

Homework

- Use Whimsical
- Draw a flowchart for:
- **Grocery Store Kiosk Checkout Process**

Checkout – 1/2

```
# Start - checkout
import random
items = []
print("Welcome to Checkout System =====")

# Scan Items
while True:
    item_name = input("Enter item name: ")
    if item_name.strip() != "":
        # Find the price and add item to list
        item_price = round(random.uniform(0.50, 10.00), 2)
        items.append({"name": item_name, "price": item_price})
        print(f"Item: ${item_name} @ ${item_price:.2f}")
    else:
        # Exit while loop for just hitting Enter key
        break
```

Checkout – 2/2

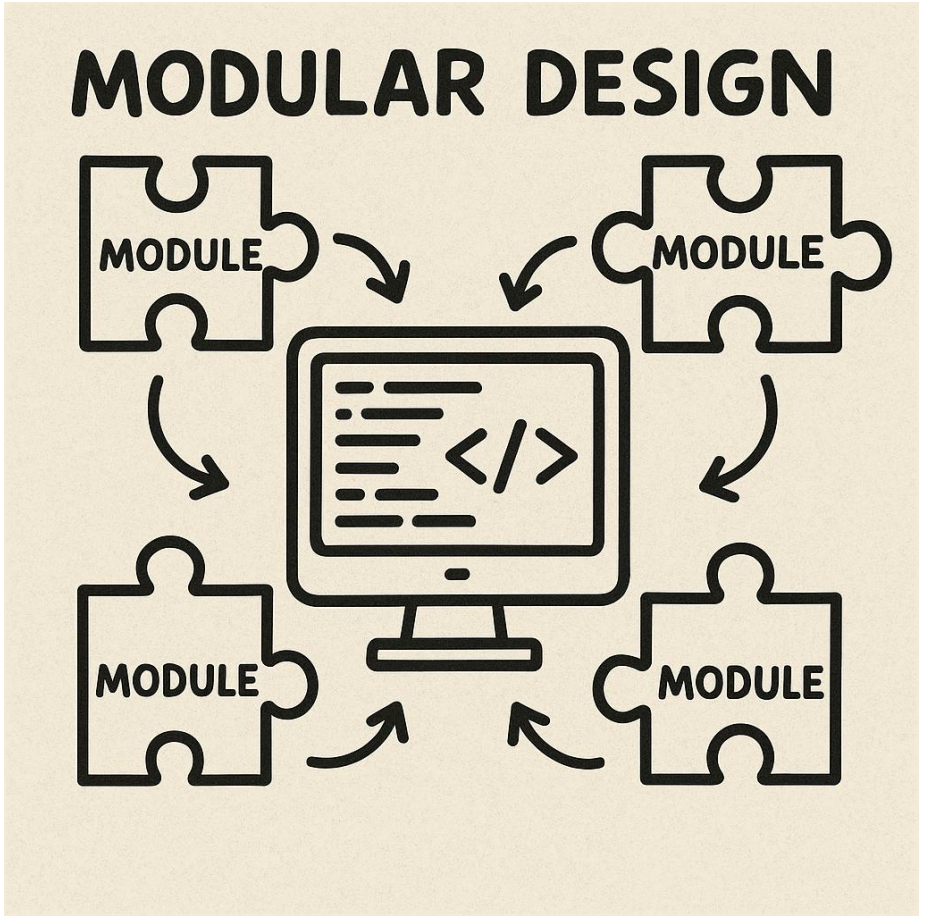
```
# Process if user has input more than zero item(s)
if len(items) > 0:
    # Calculate final total
    total = 0
    for item in items:
        total += item["price"]
    print(f"Total: ${total:.2f}")
    # Payment method
    payment_method = input("Payment method (cash/card): ")
    if payment_method not in ["cash", "card"]:
        print("Error: Invalid payment method!")
    else:
        # Process Payment
        amount_paid = float(input("Enter payment amount: "))
        # Is payment sufficient?
        if amount_paid >= total:
            print(f"Payment successful by {payment_method}! Change: ${amount_paid - total:.2f}")
            # Issue receipt
            print("Receipt:")
            for item in items:
                    print(f"{item['name']}: ${item['price']:.2f}")
            print(f"Total: ${total:.2f}")
            print("Checkout complete!")
        else:
            print("Error: Insufficient payment!")
```

Output

```
Welcome to Checkout System =====
Enter item name: Apple
Item: $Apple @ $4.03
Enter item name: Banana
Item: $Banana @ $5.25
Enter item name:
Total: $9.28
Payment method (cash/card): cash
Enter payment amount: 10
Payment successful by cash! Change: $0.72
Receipt:
Apple: $4.03
Banana: $5.25
Total: $9.28
Checkout complete!
```

Modular Design

- Better code organization
- Reusability
- Easier testing
- Easier debugging and maintenance



Turn Messy Code into Modular Code

- The checkout code is messy
- Let's break it into functions:
  - **`scan_item()`**
  - **`calculate_total()`**
  - **`Get_payment_method()`**
  - **`process_payment()`**
  - **`print_receipt()`**
- Use **`main()`** function to execute these functions – system starting point

Modularized Code

- Copy the scanning code into **`scan_item()`**, total code into **`calculate_total()`**, etc.
- Test each function!