

7/11/21
Task 4: Use various data types, List, Tuples and Dictionary in python programming key terms covered: Datatypes List, tuple, set, dict

4.1 List - Cafeteria Sales

In your college cafeteria, the sales (in units) of a new snack are recorded for 7 days, Monday to Sunday. Store these values in a list, then find the total and average sales, identify the best and worst sales days using `index()`

Aim: Record a cafeteria's snack sales for 7 days using a list; compute total and average sales, find the best/worst day, and count how many days crossed a target.

Algorithm:

- 1) Start
- 2) Create an empty list `sales = []`.
- 3) For 7 days, append integer sales to list using `append()`.
- 4) Compute `total = sum(sales)` and `avg = total/7`.
- 5) Find `max_val = max(sales)`, `min_val = min(sales)`.
- 6) find corresponding days with `index()` (add +1 to convert to day numbers).
- 7) Count days above a target using `count()` on a boolean re-map (`or`) with a loop.
- 8) Stop.

Program (uses `append()`, `index()`, `count()`):

LIST scenario

day8 = 7

~~sales = []~~

target = 500 # target sales for the day

for s in range(8):

sample_entries = int(input('Enter the seven days sales count'))

sales.append(sample_entries) # list.append()

2
To
9
1
1

Output:

[0, 20, 30]

[10, 30]

[30]

(0, 10) (0, 20) (0, 30) (0, 40) (0, 50) (0, 60) (0, 70) (0, 80) (0, 90)

(10, 20) (10, 30) (10, 40) (10, 50) (10, 60) (10, 70) (10, 80) (10, 90)

[10, 80] (10, 90)

The minimum : 8

(8, 10) (8, 20)

The maximum : 89

The sum : 8 : 156

The average : 26.0

TOTAL	
8	156
8	26.0
8	156
8	26.0

Counting of numbers not complete yet next three
rows will be added soon up to 1000 rows without

```
total = sum(sales)
avg = total / days
max_val = max(sales)
min_val = min(sales)
best_day = sales.index(max_val) + 1 # list index()
worst_day = sales.index(min_val) + 1
print("sales (Mon-Sun):", sales)
print("Total:", total)
print("Average", round(avg, 2))
print("Best day:", best_day, "with", max_val)
print("Worst day:", worst_day, "with", min_val)
```

4.2 Tuple-Lab timetable

Your department has a fixed daily lab schedule represented by a tuple of starting hours (24-hour format). Write a program to check if a given start time exists in the tuple, count how many times it appears using `count()`, find its first position using `index()` and display morning and after noon slots using slicing.

Aim

To manage and query an immutable daily lab slot schedule using a tuple, demonstrating membership check, `count()`, `index()` and slicing.

Algorithm

- 1) Start
- 2) Define slot as a fixed tuple of integers
- 3) Read query hour
- 4) Check existence with query in slots
- 5) Use `count()`; if positive, use `index()` to find the first position
- 6) Slice into morning and afternoon
- 7) Print results
- 8) Stop

Program

Tuple Scenario

```
slot = (9, 11, 14, 16, 14) # immutable daily schedule
```

query = 14

exists = (query in slot)

freq = slot.count(query) # tuple.count()

first_pos = slot.index(query + 1 if exists else "N/A") # tuple.index()

Output:

10, "hello", 3.14, "world")

(0

hello

3.14

world

(hello, 3.14)

(10, "hello", 3.14)

(0, "hello", 3.14)

(10, "hello", 3.14)

morning = slots[:2]

afternoon = slots[2:]

printf("All Lab slot", slots)

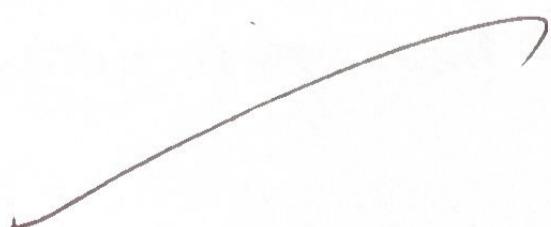
printf("Is %query:00 present?", exist)

printf("%query:00 occurs", freq, "time(s)"))

printf("first occurrence position-based", first-pos)

printf("Morning slots", morning)

printf("Afternoon slots", afternoon)



4.3 Dictionary - Bookstore Billing

A book store's price list is stored in a dictionary where keys are items names and values are price. Update the price of an item using update(), find the costliest item using max() on items(), remove an item from stock of item using pop() and display all keys, values, and items.

Aim: To manage a live price list and bill a customer using dictionary method and views.

Algorithm:

- 1) Start
- 2) Create an empty dictionary prices
- 3) Ask the user for the number of items in the price list
- 4) Repeat for each item:
 - 5) Get the item name
 - 6) Get the item price.
 - 7) Add the item and price to prices
 - 8) Ask the user for an item to update (or press Enter to skip)
 - 9) If the costliest item by checking each item's price and update it.
 - 10) Find the costliest item by checking each item's price
 - 11) Ask the user for an item to remove (or press Enter to skip)
 - 12) If given, remove that item from prices.
 - 13) Show all available items their price, the costliest item, and the removed item's price
 - 14) STOP.

Python Program:

```
prices = {}
```

```
n = int(input("Enter number of items in price list:"))
```

```
for i in range(n):
```

```
item = input("Enter item name")
price = float(input(f"Enter price of {item}:"))
prices[item] = price

# optional price revision
re_item = input("Enter item to update price list:")
rev_item = input(f"Enter item to update price or press Enter to skip: ")
if rev_item in prices:
    new_price = float(input(f"Enter new price for {rev_item}:"))
    prices.update({rev_item: new_price}) # dict updated

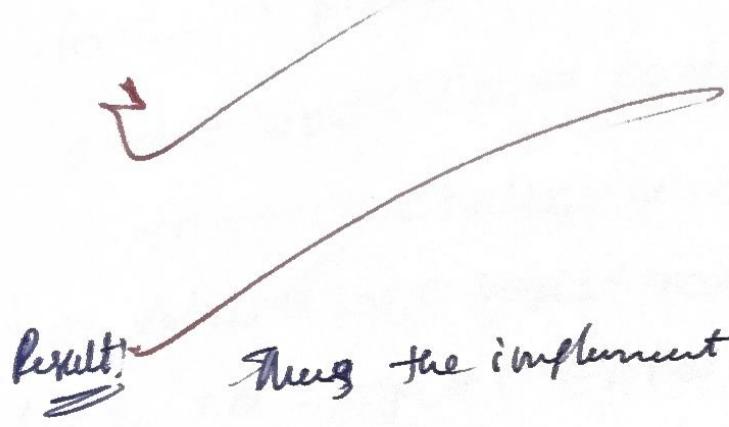
# Find costliest item
costliest_item = None
max_price = 0
for item, price in prices.items():
    if price > max_price:
        max_price = price
        costliest_item = item

## Remove out of stock item
remove_item = input("Enter an item to remove from price list (or press Enter to skip): ")
remove_price = None
if remove_item in prices:
    removed_price = prices.pop(remove_item, None)
    # dict.pop()
```

```
# display results
print("Available Items", list(prices.keys())) # dict.keys()
print("Prices", list(prices.values())) # dict.values()

if costlist_item:
    print("Costlist item", costlist_item, "at", max_price)

if remove_item:
    print("If removed", remove_item, "price", removed_price)
```



Result: Using the implement the dictionary BookStore Billing.

(U.S. Geological Survey Water-Supply Paper No. 1470, 1917)

Output:

```
{'name': 'Alice', 'age': 30, 'city': 'New York'}
```

Alice

{'name': 'James', 'age': 30, 'city': 'New York'}

{name}: 'James' {age}: 303

Canis lupus?

KE 4-age

```
dict_items([('name', 'James'), ('age', 30)])
```

Monteiro's bird

37069 = maz-f

On 8th Dec.

162 May

1981-08-26 10:00 AM - 10:30 AM

1913-34
in progress

With thanks

1984-1985

$\left(\frac{2+3x}{2} \right)^{\frac{1}{3}}$

• 100

W. H. C.

Waco, Texas Higgin

9 (9) 100

1968-1970

house

None

1982. 10024.

1983.6.1

W. C. Ying

W. H. C.

4.4 Set - Tech For participation

Two events, AI Hackathon and Robotics challenge have participants IDs stored in two sets. Add a late registrant to AI Hackathon, remove own ID from participant from Robotics using discard(), then find participants in both events (intersection()), only in one (difference()), the total unique participants (union()).

Get AI Hackathon participants

```
ai_hackathon = set()
```

```
n1 = int(input("Enter number of participants in  
AI Hackathon:"))
```

```
for _ in range(n1):  
    pid = input("Enter participant ID:")  
    ai_hackathon.add(pid)
```

Get Robotics Challenge participants

```
robotics_challenge = set()
```

```
n2 = int(input("Enter number of participants in  
Robotics challenge:"))
```

```
for _ in range(n2):  
    pid = input("Enter participant ID:")  
    robotics_challenge.add(pid)
```

Add a late registrant

```
late_id = input("Enter late registrant participant  
ID from Robotics challenge (or press Enter  
to skip):") AI Hackathon
```

If remove_id:

```
robotics_challenge.discard(remove_id) # set.  
discard()
```

adpat)

```
Print("In AI-hackathon : ", ai_hackathon)
Print("Robotics Challenge : ", robotics_challenge)
Print("Both events : ", both)
Print("Only AI : ", only_ai)
Print("Only Robotics : ", only_robotics)
Print("Total unique participants : ", unique_all)
```

(1) ~~reactions~~ ~~for~~ ~~synthesis~~ IA

:(14) ~~www.123456.com~~

Championships in 1979, 1980

(big b b b - mato)

1988-10
Anoectochilus longiligulus

Oct 22 = 10 miles S of Hodges

Wills - 24780 02

W: spread the eyes -

($\leq n$) \max_{\sigma \in S} \sigma

: Although there was no
such a thing as a

(big) $\theta = \frac{1}{2} \pi$ $\Rightarrow \theta = 90^\circ$

Hospital for Jaundice
Hosp. for Jaundice

~~Strewn~~ ~~Scattered~~ ~~Scattered~~ ~~Scattered~~ ~~Scattered~~

~~HNO₃ + 2 NaOH → NaNO₃ + H₂O~~

~~Chlorophyll a + b~~

~~1900~~ 1901-02
High School Dr. C. H. Ladd

Observe the following products of combustion.

Peselt

```

if late_id:
    ai_hackathon.add(late_id) # set add()

# Remove a with drawn participant
remove_id = input("Enter with drawn participant ID  
from robotics challenge (or press  
Enter to skip!)")

; + remove_id:
    robotics_challenge.discard(remove_id) # remove(id) & discard()

# Set operations
both = ai_hackathon.intersection(robotics_challenge)
only_ai = ai_hackathon.difference(robotics_challenge)
only_robotics = robotics_challenge.difference(ai_hackathon)
unique_all = ai_hackathon.union(robotics_challenge)

# Output
print("In AI Hackathon:", ai_hackathon)
print("Robotics challenge:", robotics_challenge)
print("Both events:", both)
print("Only AI:", only_ai)
print("Only Robotics:", only_robotics)
print("Total unique participants:", len(unique_all))

*-----*

```

VEL TECH	
EX NO.	
PERFORMANCE (5)	4
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	15
SIGN WITH DATE	10

Result: Thus implement the
Set tech fair participation