1

Collection = single "variable" use to store multiple values
List = [] ordered and changeable, Duplicates ok.
Set = { } unordered and immutable, but Add/Remove ok. No duplicates
Tuple = ( ) ordered and unchangeable Duplicates ok. Faster

```
Fruit = "apple"          Collection

Fruits = ["apple", "orange", banana", "coconut"] List

print(Fruits[0]) // apple

print(Fruits[0:3]) // ['apple', 'orange', 'banana']

print(Fruits[:2]) // ['apple', 'banana'

print(Fruits[::-1]) // ['coconut', 'banana', orange', 'apple']

For fruit in fruits:          // apple
    print(fruit)                 orange
                                 banana
                                 coconut

print(dir(Fruits)) → list
print(help(Fruits))

print(len(fruits))            // 4

print("apple" in Fruits) // True

Fruits[0] = "Pineapple"
For fruit in Fruits:          // Pineapple
    print(fruit)                 orange
                                 banana
                                 coconut
```

Fruits.append("Pineapple") هيزود عنصر جديد في الاخر
Print(Fruits) // ['apple', 'orange', 'banana', Coconut', 'Pineaple']

Fruits.remove("apple")
Print(Fruits) // ['orange', 'banana', 'coconut']

Fruits.insert(0,"Pineapple") هيزود في الليست الى اول عنصر
Print(Fruits) // ['Pineapple', 'apple', 'orange', 'banana', 'coconut']

Fruits.sort() هترتبهم ابجديا
Print(Fruits) // ['apple', 'banana', 'Coconut', 'orange']

Fruits.reverse() هتطبع من الاخر الاول
Print(Fruits) // ['coconut', 'banana', 'orange', 'apple']

Fruits.clear() هتمسح كل حاجه
Print(Fruits) // []

Print(Fruits.index("apple")) // 0 ترجع

Print(Fruits.count("banana")) // 1 هيعد كام مره موجود

set

```
Fruits = {"apple", "orange", "banana", "Coconut"}

print(dir(fruits))
print(help(fruits))
print(len(fruits)) // 4

print("Pineapple" in Fruits) // False

print(Fruits[0]) // TypeError → unordered

Fruits.add("pineapple") // {'orange', 'apple', 'Pineapple'
```

طبعا مش هنطبعوا مترتبين عشان ال set

```
Fruits.remove("apple")
print(Fruits) // {'Coconut, 'orange', 'banana'}

Fruits.pop()
```

هتحذف اي واحده بمزاجها

```
Fruits.clear()
print(fruits) // set()
```

واحده بس الكوكنت

```
Fruits = {"apple", "orange", "banana", "Coconut", "Coconut"}
print(Fruits) // {"apple", "orange", "banana", "Coconut"}
```

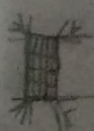
No duplicates — طبعا عشان هي

Tuple

```
Fruits = ("apple", "orange", "banana", "coconut", "coconut")
print(Fruits)
// ('apple', 'orange', 'banana', 'coconut', 'coconut')
                    ordered , Duplicates
print(len(Fruits)) // 5
print("pineapple" in fruits) // False
print(Fruits.index("apple")) // 0
print(Fruits.count("coconut")) // 2
for Fruit in fruits // apple
    print(fruit)      orange
                      banana
                      Coconut
                      Coconut
```

Shopping cart program

```
Foods = []
Prices = []
TOTAL = 0

while True:
    Food = print("Enter a Food to buy (q to quit): ")
    if Food.lower() == "q":
        break
    else:
        Price = float(input(f"Enter The Price of a {food}: $
        Foods.append(Food)
        Prices.append(Price)

Print("---Your Cart---")

For Food in Foods
    print(Foods, end=" ")

for price in Prices
    TOTAL += Price
print()
Print(f"Your total is: ${TOTAL}")
```



2D Lists

```
Fruits = ["apple", "orange", "banana", "coconut"]
Vegetables = ["Celery", "carrots", "Potatoes"]
Meats = ["chicken", "Fish", "Turkey"]
```
1D

```
groceries = [ Fruits, Vegetables, Meats ]
```
2D

```
Print (groceries)
//[["apple", "orange", "banana", "coconut"], ["celery", "carrots", ...]]

print (groceries[0]) //['apple', 'orange', 'banana', 'coconut']
                row 0
print (groceries[0][0]) // apple
                row 0 column 0

For Collection in groceries:
    print(Collection) // Fruits List
                      Vegetables List
                      Meats List

For Collection in groceries: -> Row
    For Food in Collection: -> Column
        print(Food, end = " ") // apple orange banana...
    print()                    // Celery carrots Potatoes
                               // chicken Fish Turkey
```

2D KeyPad For Phone

4 row
3 colum

```
num_pad = ((1, 2, 3), (4,5,6), (7,8,9), ("*", 0, "#"))

For row in num_pad
    For column in row:
        print(column, end=" ")
    print()
```

Quiz Game

```
questions = ("How many elements are in the Periodic Table?: ",
             "Which animal lays the largest eggs?: ",
             "What is the most abundant gas in Earth's atmosphere?: ",
             "How many bones are in human body?: ",
             "Which Planet in the Solar system is the hottest?: ")
options = (("A. 116", "B. 117", "C. 118", "D. 119"),
           ("A. Whale", "B. Crocodile", "C. Elephant", "D. ostrich"),
           ("A. Nitrogen", "B. oxygen", "C. Carbon Dioxide", "D. Hydrogen"),
           ("A. Mercury", "B. Venus", "C. Earth", "D. Mars"))
answers = ("C", "D", "A", "A", "B")
guesses = []
score = 0
question_num = 0
For question in questions:
    print("---------")
    print(question)
    For option in options[question_num]:
        print(option)
```

```
guess = input("Enter (A, B, C, D): ").upper()
guesses.append(guess)

if guess == answers[question_num]:
    score += 1
    print("Correct")
else
    print("Incorrect")
    print(f"{answers[question_num]} is the correct.")

question_num += 1

print("------------")
print("  Results  ")
print("------------")
print("answers: ", end=" ")
for answer in answers:
    print(answer, end=" ")
print()

print("guesses: ", end=" ")
for guess in guesses:
    print(guess, end=" ")
print()

score = int(score / len(questions) * 100)
print(f"your score: {score}")
```

Dictionary = a collection of [Key: value] pairs
ordered and changeable. No duplicates

```
Capitals = {"USA": "Washington D.C.", "India": "New Delhi",
            "China": "Beijing", "Russia": "Moscow"}
Print(dir(Capitals))
Print(help(Capitals))

Print(Capitals.get("USA")) // Washington D.C.
Print(Capitals.get("Japan")) // None

If Capitals.get("Japan"):
    Print("That capital exists")
else:
    Print("That capital doesn't exist")

Capitals.update({"Germany": "Berlin"})

Capitals.pop("china") →

Capitals.popitem() →
Capitals.clear() // {}
Keys = Capitals.keys()
Print(keys) // dict_keys(['USA', 'India', 'china', 'Russia'])
For key in Capitals.keys():
    print(key, end=" ") // USA India china Russia
Values = Capitals.values()
    print(values) // dict_values(['Washington D.C.', 'New Delhi', ...])
For value in Capitals.values()
    Print(value, end=" ") // Washington D.C. New Delhi Beijing
```

```
items = capitals.items()
print(items)
# dict_items([('USA', 'Washington D.C.'), ('India', 'New Delhi')])

items = [(), (), ()]

for key, value in capitals.items():
    print(f"{key}: {value}")
# USA: Washington D.C.
  India: New Delhi
```

```
#Concession stand program
#dictionary {key: value}

menu = {"Pizza": 3.00, "nachos": 4.50, "Popcorn": 6.00,
        "Fries": 2.50, "chips": 1.00, "pretzel": 3.50,
        "Soda": 3.00, "Lemonade": 4.25}

cart = []
total = 0
print("---- Menu ----")
for key, value in menu.items():
    print(f"{key:10}: ${value:.2f}")
print("----------")
while True:
    food = input("Select an item (q to quit): ")
    if food.lower() == "q":
        break
    elif menu.get(food) is not None:
        cart.append(food)
```

```
For food in cart:
    total += menu.get(food)
    print(food, end=" ")
print()
print(f"Total is: ${total:.2f}")
```

*random numbers

```
import random

number = random.randint(1, 6)
print(number)          # 3
```

رقم عشوائي من 1 ← 6
او رقم عشوائي

```
low = 1
high = 100
number = random.randint(low, high)
print(number)          # 75

number = random.random()
print(number)          # 0.327...
```

رقم عشوائي 0 ← 1

```
options = ("rock", "paper", "scissors")
option = random.choice(options)
print(option)          # paper
```

shuffle

```
cards = ["2","3","4","5","6","J","Q","K","A"]
random.shuffle(cards)
print(cards)           # ['K','A','2','5','Q','6','J'...]
```

*Number Guessing Game

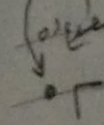
```
import random
low = 1
high = 100
guesses = 0
number = random.randint(low, high)

while True:
    guess = int(input(f"Enter a number between {low} - {high}: "))
    guesses += 1
    if guess < number:
        print(f"{guess} is low")
    elif guess > number:
        print(f"{guess} is high")
    else:
        print(f"{guess} is correct!")
        break

print(f"This round took you {guesses}")
```

ROCK PAPER SCISSORS

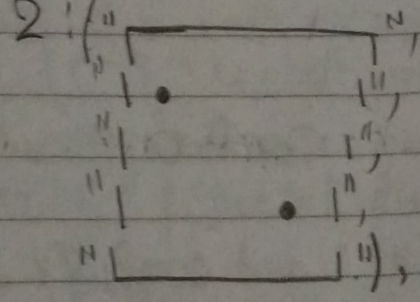
```
import random
options = ("rock", "paper", "scissors")  running = True
while running:
    player = None
    computer = random.choice(options)
    while player not in options:
        player = input("Enter a choice (rock, paper, scissors): ")

    print(f"player: {player}")
    print(f"computer: {computer}")

    if player == computer:
        print("it's a tie!")
    elif player == "rock" and computer == "scissors":
        print("you win")
    elif player == "paper" and computer == "rock":
        print("you win")
    elif player == "scissors" and computer == "paper":
        print("you win")
    else:
        print("you lose")
    if not input("play again(y/n): ").lower() == "y":
        running = False

print("thanks for playing")
```

Dice roller Program

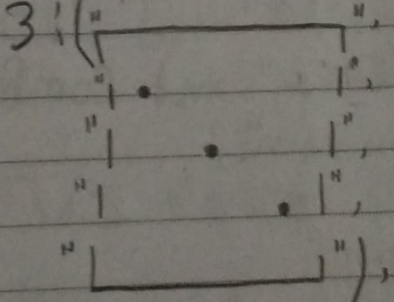
```
import random
print("\u250c\u2500\u2510\u2502\u2514\u2518")
```



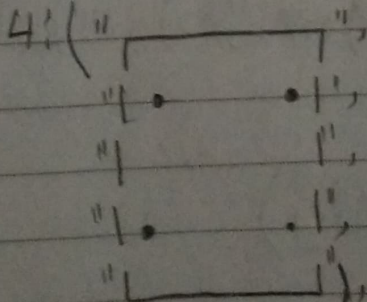
```
dice_art = {1:("
                ",
                "
                ",
                "
                ",
                "
                ",
                "
                "),
```



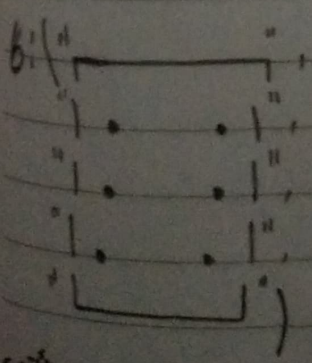
```
2:("
   ",
   "
   ",
   "
   ",
   "
   ",
   "
   "),
```



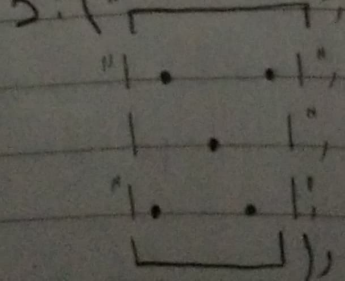
```
3:("
   ",
   "
   ",
   "
   ",
   "
   ",
   "
   "),
```



```
4:("
   ",
   "
   ",
   "
   ",
   "
   ",
   "
   "),
```



```
6:("
   ",
   "
   ",
   "
   ",
   "
   ",
   "
   "),
```



```
5:("
   ",
   "
   ",
   "
   ",
   "
   ",
   "
   "),
```

```
dice = []
total = 0
num_of_dice = int(input("How many dice?: "))

for die in range(num_of_dice):
    dice.append(random.randint(1,6))
for die in range(num_of_dice):
    for line in dice_art.get(dice[die]):
        print(line)
```

=

```
for line in range(5)
    for die in dice:
        print(dice_art.get(die)[line], end = " ")
    print()
for die in dice:
    total += die
print(f"TOTAL: {TOTAL}")
```

Encryption Program

```
import random
import string

chars = " " + string.punctuation + string.digits + string.ascii_letters
chars = list(chars)
key = chars.copy()
random.shuffle(key)
print(f"chars: {chars}")
print(f"key: {key}")
plain_text = input("Enter a message to encrypt: ")
cipher_text = ""

for letter in plain_text:
    index = chars.index(letter)
    cipher_text += key[index]
print(f"original message: {plain_text}")
print(f"encrypted message: {cipher_text}")
```

Function = A block of resuable code

define ↑ Place () after the function name to invoke it

```
def happy_birthday():
    print("Had HBD")
    print("U're old")
    print("HBD")
    print()

happy_birthday()
```

```
def happy_birthday(name):
    print(F"happy birthday to {name}")
    print()

happy_birthday("Rabha")
```

argument ↑

```
def happy_birthday(name, age):
    print(F"happy birthday to {name}")
    print(F"your age is {age}")

happy_birthday("Rabha", 21)

def display_invoice(username, amount, due_date):
    print(F"Hello {username}")
    print(F"your bill of ${amount:.2f} is due {due_date}")
display_invoice("Brocode", 42.50, 01/01
```

return = statement used to end a function and send a result back to the caller

```
def add(x, y):
    z = x + y
    return z

def subtract(x, y):
    z = x - y
    return z

def multiply(x, y):
    z = x * y
    return z

def divide(x, y):
    z = x / y
    return z

print(add(1, 2))      // 3
print(subtract(1, 2)) // -1

def create_name(first, last):
    first = first.capitalize()
    last = last.capitalize()
    return first + " " + last
full_name = creat_name("Rabia", "gharib")
print(full_name) // Rabia Gharib
```

default arguments - A default value for certain parameters
default is used when that argument is omitted
make your functions more flexible, reduce#
1. Positional 2. default 3. keyword 4. arbitrary

```
def net_price(list_price, discount, tax):
    return list_price * (1 discount) * (1 + tax)

net_price
print(net_price(500, 0, 0.05))  // 525.0

def net_price(list_price, discount=0, tax=0.05):
    return list_price * (1 - discount) * (1 + tax)

print(net_price(500))  // 525.0

print(net_price(500, 0.1))  // 472.0

print(net_price(500, 0.1, 0))  // 450.0
```

default default

exercises

```
import time

def count(start, end):
    for x in range(start, end+1):
        print(x)
        time.sleep(1)
    print("Done")

count(0, 10)
```

keyword arguments = an argument preceded by an identifier
helps with readability
order of arguments doesn't matter
1. Positional 2. default 3. keyword 4. arbitrary

```
def hello(greeting, title, first, last):
    print(f"{greeting} {title}{first} {last}")

hello("Hello", first="spongebob", title="Mr.", last="squarepants")
// Hello Mr. spongebob squarepants

for x in range(1, 11):
    print(x, sep="_", end="")   // 1_2_3_4_5_6
```

keyword arguments

```
def get_phone(country, area, first, last):
    return f"{country}-{area}-{first}-{last}"

phone_num = get_phone(country=1, area=123, first=456, last=7890)

print(phone_num) // 1-123-456-7890
```

Tuple arguments

*args = allows you to pass multiple non-key arguments
** Kwargs = allows you to pass multiple keyword_arguments
keyword arguments / dictionary
* unpacking operator
1. Positional 2. default 3. keyword 4. ARBITRARY

```
def aff(*args)
    total = 0
    For arg in args:
        total += arg
    return total

print(aff(1,2,3)) //6

def display_name(*args):
    For arg in args:
        print(arg, end=" ")

(display_name("Rabha", "Gharib")) // Rabha Gharib

def print_address(**kwargs):
    For value in kwargs.values():
        print(value)

print_address(street="123 Fakest",
              city="Detroit", state="MI",
              zip="54321")
```

//123 Fakest St
Detroit
MI
54321

```
def shipping_label(*args, **kwargs):
    for arg in args:
        print(arg, end=" ")
    print()
    for value in kwargs.values():
        print(value, end=" ")
    print(f"{kwargs.get('street')}")
shipping_label("Dr", "Spongebob", "Squarepants", "III",
               street="123 Fake St",
               city="Detroit",
               state="MI",
               zip="54321")
```

module = a file containing code you want to include in your program

use 'import' to include a module (built-in or your own)

useful to break up a large program reusable separate files

```
print(help("modules"))
```

math →

```
import math
print(math.pi)    # 3.14...

import math as m
print(m.pi)       # 3.14...

from math import pi
print(pi)         # 3.14...
```

```
From math import e

a, b, c, d, e = 1, 2, 3, 4, 5
                    ↓Power
Print(e**a)   // 2.7182...

Print(e**e)   // 3125

Print(e**a)   // 5

print(math.e**e)  // 148.4131...
```

to create a module
click Right in Project Folder → new → Python File
example.py

```
PI = 3.14159

def square(x):
    return x**2

def cub(x):
    return x*x3

def circumference(radius):
    return 2*Pi*radius

def area(radius):
    return Pi*radius*2
```

```
import example

result = example.Pi
print(result)  // 3.14159

result = example.square(3)
Print(result)  // 9
```

Variable scope = Where a variable is visible and accessible
scope resolution = (LEGB) Local -> Enclosed -> Global -> Built-in

```
def Func1():
    a = 1
    print(a) b  //error
```

Func1 لا تعرف Func2 متغير

```
def func2():
    b = 2
    Print(b)

Func1()  // 1
Func2()  // 2
```

```
def Func1():
    Print(x)

def func2():
    Print(x)

x = 3  ---> Global
Func1()  // 3
func2()  // 3
```

```
def Func1():
    x = 1
    print(x)

def Func2()
    x = 2
    print(x)

x = 3     ←     مش هاخدها     ,     Global اسمها

Func1     // 1
Func2     // 2


from math import e  → bultin

def Func1():
    print(e)

Func1()   // 2.71828...

from math import e → built-in

def func1()
    print(e)

e = 3   → Global                    هاخد دي
Func1()          //3        LEGB     عشان
                            →
```

exception = events detected during execution that interrupt the flow of a program

```
try:
    numerator = int(input("enter a number to divide: "))
    denominator = int(input("enter a number to divide by: "))
    result = numerator / denominator
    print(result)
except Exception:
    print("Something went wrong")
except ZeroDivisioError:
    print("you can't divide by zero")
except ValueError:
    print("Enter only number")

except Exception as e:
    print(e)
```

```
except ValueError as e:
    print(e) → // invalid literal for int() with base 10:
    print("Enter an integer")

else:
    print(result)   exception ...  else ...

Finally:
    print("This will always execute")
```

Python File detection

Import os small

← Properties ← Click Right ← test.txt ← location ← copy

Path = "C:\\Users\\Carlon\\Desktop\\test.txt"

```
if os.Path.exists(Path):
    Print("that location exists")
else:
  Print("not exist")
```

```
if os.Path.exists(Path):
    print("That location exist")
    if os.Path.isFile(Path):
        print("that is a file")
    elif: os.Path.isdir(Path):
        Print("That is Folder")
else:
    print("not exist")
```

Read File in Python

File Path

```
with open('test.txt') as File:
    print(File.read())
print(File.closed) //True
```

```
try:
    with open('test.tx') as File:
        print(File.read())
except FileNotFoundError:
    print("that File was not found")
```

write File in python

```
text='Yoooo\nThis is somtext\n Have agoodone\n'
with open('test.txt','w') as File:
    File.write(text)
```

w → write

a → appen

Copy a file

```
copyfile() = Copies contents of a file
copy() = copyfile + Permission mode + destination can be a directory
copy2() = copy + Copies metadata (File's creation and modification Times)

import shutile
shutile.copyfile('test.txt','copy.txt') # src, dst
```

copy()
copy2()

move a file

test.txt ← Fille , وأنتا في بروجيكت علي هضغط

```
import os

source = "test.txt"
destination = "C:\\users\\Cakom\\Desktop\\test.txt"
```

("test.txt": عشان موجود جوه الـ project اكتب اسمه بس)

```
try:
    if os.path.exists(destination):
        print("there is already a file there")
    else:
        os.replace(source, destination)
        print(source + " was moved")

except FileNotFoundError:
    print(source + " was not found")
```

(هتنقل هنا → destination)

delete a file

```
import os

os.remove('test.txt')
```

('test.txt': موجود عندي ← Path)

```
path = "test.txt"

try:
    os.remove(path)
except FileNotFoundError:
    print("That file was not found")
except PermissionError:
    print("You dont have permission to delete that")
except OSError:
    print("You cannot delete that using that fun")
else:
    print(path + " was deleted")
```

os.remove(path) #delete a file
os.rmdir(path) #delete an empty file
shutil.rmtree(path) #delete a directory containing files

الأضخم