ABDUL
fun-1

↳ Function is a piece of code which performs specific tasks
↳ Known functions
↳ Reusable - write once, use many times
↳ Takes i/p as parameters & return o/p
↳ Built-in / User-defined.

→ User defined functions

a. Header
b. return
c. calling
d. parameters

def fun_name(<parameters>): i/p
    statement 1
    statement 2
    return value (o/p)

def volume(length, breadth, height): Formal parameters
    vol = length * breadth * height
    return vol

v = volume(10, 5, 3) → Actual parameters
print(v)

```
def volume(length, breadth, height):
    vol = length * breadth * height
    return vol

if __name__ == '__main__':
    v = volume(10,5,3)
    pt(v)  # 150.
```

Positional / keyword Args

↳ v = volume(10,5,3)

↳ pass in same order

positional arguments

```
def volume(length, breadth, height):
    pt(length, breadth, height)
    vol = length * breadth * height
    return vol
v = volume(10,5,3)
pt(v)
```

10, 5, 3
#150
5, 10, 3
#150

Key based

Keyword based

```
def volume(length, breadth, height):
    pr('height=', height)
    pr('breadth=', breadth)
    pr('length=', length)
V = volume(length=10, breadth=5, height=3)
pr(V)
```

height = 3
breadth = 5
length = 10

length : 10
breadth : 5
height : 3

Pass in any order

Case 1

```
V = volume(length=10, breadth=5, height=3)
```

length : 10
breadth : 5
height : 3

Case 2

```
V = volume(breadth=10, height=3, length=5)
```

Case 3

```
V = volume(5, length=10, height=3)
```

* TypeError : volume() got multiple value

V = volume(length = 10, 5, 3)

// syntax error
// positional argument follows keyword argument

Correct
V = volume(10, 5, height = 3)

Consolidated

def volume(l, b, h)

a. v = volume(10, b = 5, h = 3) ⇒ Right

b. V = volume(10, b = 5, 3)
→ keyword should be right hand side

c. V = volume(l = 10, 5, K = 3)
→ wrong

d. V = volume(10, 5, b = 3)
Multiple times of b

e. V = volume(10, 5, height = 3) ✓

Default arguments

Fun -3
Abdul

↳ L1 = [10, 20, 30, ...]

L1.index(20)
L1.index(20, 2)
L1.index(20, 2, 4)
L1.pop(index)

↳ def volume(l, b, h):
    v = l*b*h
    return v
v = volume(10, 5, 3) # 150

↳ def volume(l, b=1, h=1):
    v = l*b*h
    return v
v = volume(10, 5) # 50
v = volume(10) # 10
v = volume() # error

↳ def volume(l=1, b, h=1): Fill it from right side
    v = l*b*h
    return v
v = volume(10, 5, 3) # syntax error

```
def volume(l=1, b=1, h=1):

v=volume()  # 1 1

def fun(a,b,c):
    pf(a,b,c)
fun(5,10,15)  # 5, 10, 15

def fun(a=1, b=2.5, c="hello")
    pf(a,b,c)
fun(5,10,15)  # 5, 10, 15
fun()         # 1, 2.5, "hello"

def fun(a=1, b=2.5, c=[1,2,3]):
    pf(a,b,c)
fun()         # 1, 2.5, [1,2,3]
fun(5,10,15)  # 5, 10, 15
fun(5,10,[10,11])  # 5, 10, [10,11]
# Function takes any type of argument
# pass any type of argument
```

Abdul

func

```
def func(l=[1,2,3]):
    l.append(len(l))
    # [1,2,3,3]

p(1)

func()  # [1,2,3,3]
func()  # [1,2,3,3,4]
func([10,11])  # 10,11,2
func()  # [1,2,3,3,4,5]
```

default arguments are created only first time.

Position x only Arguments

```
def fun(a,b,/,c,d):
    pt(a,b,c,d)

fun(5,10,c=15,d=20)  X
fun(a=15,b=10,c=15,d=20)  Typeerror
fun(5,10,c=15,d=20)
fun(5,10,15,d=20)
fun(5,b=10,c=15,d=20)  Typeerror
```

a,b → Positional
c,d → positional/keyword

↳ def fn (a,b,c,d,/) → All positional
def fun (/,a,b,c,d) → only not allowed
def fn (a,/,b,c,d) → only a is positional only.

Keyword only

a,b → keyword/positional
c,d → keyword only

```
def fun(a,b,*,c,d):
    pt(a,b,c,d)
def fun(a,b,c,d,*):
    pt(a,b,c,d)
fun(5,10,15,20) // syntax error
```

↳ keyword only

```
def fun(*,a,b,c,d):
    pt(a,b,c,d)
```

Keyword only & positional only

```
def fun(a,b,/,c,d,*,e,f):
    pt(a,b,c,d,e,f)
```

(P&K)

fun(a,b,c,/,*,d,e,f)

positional only — keyword only

↳ def fun(a,/,b,*,c):
pt(a,b,c)

About funs

fun(5,10,15) → Not allowed
fun(5,10,c=15) → allowed
fun(5,b=10,c=15) → allowed
fun(a=5,b=10,c=15) → Not allowed

Challenges :-

Max-3 :-

```
def max3(a,b,c):
    if(a>b and a>c):
        return a
    elif b>c:
        return b
    else:
        return c
pt(max3(5,4,7))
```

Interest

```
def interst(*,P,t,r)
    SI=(P*t*r)/1
    return SI
si=interst(P=10000,
           r=10)
pt(si)
```

```
import re
def pangram(phrase):
    letters = re.sub(r"[^a-zA-Z]", "", phrase
    letters_set = set(letters.lower())
    if letters_set == 26:
        return true
    else return false
str = "The quick brown fox jumps over
    lazy dog"
p = (pangram(str))
```