import os
import time
import platform

Function

*

if (platform.system() == 'Windows'):
    my_code ("cls", "dir")
else:
    my_code ("clear", "ls -l")

*→ def my_code (cmd1, cmd2):
    if (time.sleep(2)
    os.system (cmd1)
    os.system (cmd2)

↳ A function is a block of code for some specific operation.
↳ Reusable
↳ A function executes only when it is called.

↳ def display ():
    pt ('first line')
    pt ('second line')
    return None

display()
pt ("Display function called again")
display()

↳ display()

```
def display():
    display()
```

✗ NameError

```
def display():
    :
display()
```

↳ Rule to define function name

a-z, A-Z, 0-9, _

0-9, _

No space

@, #, $, %, ^, & ...

↳ len("hi")

↳ x = 5, 6
pt(min(x))

} Built-in Function ✓

↳ Scope

```
def myfunction1():
    pt("myfunction1")
    myfunction2()
    return None
def myfunction2():
    pt("myfunction2")
    return None
myfunction1()
```

} myfunction1
myfunction2

```
def myfunction1():
    print(x)
def myfunction2():
    print(x)

x=10
myfunction1()
myfunction2()
```

10
10

x = 10 / global variable

fun2

↳

```
def myfunction1():
    print("x")
def myfunction2():
    print("x")

#x=10
myfunction1()
myfunction2()
```

} x is not desired

↳

```
def fun1():
    x=60
    print(x)
def fun2():
    print(x)

x=10
fun1()
fun2()
```

60
10

```
def fun1():
    x=60
    pt(x)
    fun2()
    return None

def fun2()
    pt(x)
    return None

def main()
    x=10
    fun1()
    return None

main()
# 'x' not defined
```

```
def main()
    global x
    x=10
    funfun11()
    return None

main()
```

60
10

```
def fn1():
    x=60
    pt(x)
    return None

def fn2(y)
    pt(y)
    return None

def main()
    #global x
    x=10
    fn1()
    fn2()

main
```

60

10

```
def get_result():
    result = num*10
    pt(result)
    return None

def main():
    num = eval(input(" "))
    get_result()
    return None

main()
```

Lin 3

```
get_result(num)

6 ill
60 result
```

NameError

Positional | Keyword

```
def add(p,q):
    pt(p+q)
    return None

def substract(m,n):
    pt(m-n)

def main():
    a = eval(input(" "))
    b = eval(
    add(a,b)        # 11
    substr(a,b)     # -1
    subtract(5,1)   # 1
    substract(0,5)  # 5
main()              #5,6
```

15
16

} positional argument

```
def add(p,q):
    result = p+q
    print(result)
    return None

def main():
    a =
    b =
    add(a,b)

main()
```

```
6
5
11
```

```
def add(p,q):
    result = p+q
    return result

def main():
    a =
    b =
    result = add(a,b)
    return None

main()
```

5, 6

↳ positional

```
def display(a):
    pf(a)
    return None

display(4)
display()  # TypeError
```

```
def display(a=0):  #1
    pf(a)
    return None

display()   #1
display(4)  #4
display(5.5) #a=5.5
```

```
def add_numbers(p,q=10):
    result = p+q

add_numbers(5,6)  #11
add_numbers(10)   #20
```

```
def add_num(p=20,q=10):
    result = p+q

add_numbers()
# 30
```

```
def add_numbers(p=5,q):
    result = p+q

add_numbers(5)
```

✗ SyntaxError

fun 4

```
def add_numbers(p, q=22):
    result = p+q
    return None
add_numbers(5)          #27
add_numbers(10,20)      #30
add_numbers(10)         #32
```

⟺

```
def display(a,b):
    pt('a={},b={}')
    return None
display(3,4)        # 3, 4
(a=3,b=4)           # a=3, b=4
(b=44,a=33)         # a=33, b=44.
```

⟺ *arg

```
def display(*arg):
    for each in arg:
        pt(type(each), end=" ")
    return None
display(4,5,6)
pt("---")
display("Hi",4.6,64)
```

int int int
str float int

```
def display(a,b):
    pt(a,b)
    return None
display(5,6)  # 5,6
    (a=5,b=4)  # 5,4
    (b=44,a=33)  # 33,44
    (a=8,b=66,c=5)  # TypeError
```

```
def display(**kargs):
    pt(type(kargs))   # dict
    pt(kargs)         # {'a':1,'b':2}
display(a=1, b=2,c=3)
```

```
def display(p, **kargs):
    pt(p)
    pt(kargs)
display(56, q=10,r=20)   # 56 {'q':10,'r':20}
```

```
def display(**kargs):
    for each in kargs:
        pt(each)
display(a=1, b=2, c=3)
# a b c
```

Pgm 5

chdir :-

```
import os

def main():
    rev_path = input("          ")
    try:
        os.chdir(rev_path)
        pt("  os.setcwd())
    except FileNotFoundError:
        pt(
    except NOEADirectoryError:
        pt(
    except PermissionError:
        pt(
    except exception as e:
        pt(e)
    return None

if __name__ == "__main__":
    main()
```

```
pt(__name__ # __main__
def add(a,b):
    pt(f'{a+b}')

def sub(a,b):
    pt(f'{a-b}')

def main():
    pt(script1)
    add(2,3)
    sub(2,3)

if __name__=="__main__":
    main
```

script1:

```
import script1

def mult(a,b):
    pt(f'{a*b}')
    return none

def div(a,b)
    pt(f'{a/b}')
    return none

def main():
    pt("script2")
    mult(2,3)
    script1.add(4,5)
    script1.sub(4,6)
    return None

if __name__=="__main__":
    main()
```