Python
函式 Function

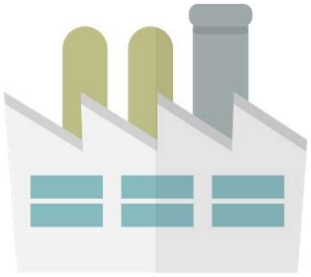
臺北科技大學資訊工程系

Function

- 使用函式的好處
  - 團隊分工：將大程式切割由多人撰寫，縮短程式開發時間。
  - 重複運用：函式可重複呼叫，縮短程式長度。
  - 提高可靠度、可讀性，利於測試驗證、除錯。
  - 後續修改使用：再開發類似功能產品，只需稍微修改即可套用。

零件:輪子、儀錶板、引擎

工廠組裝汽車



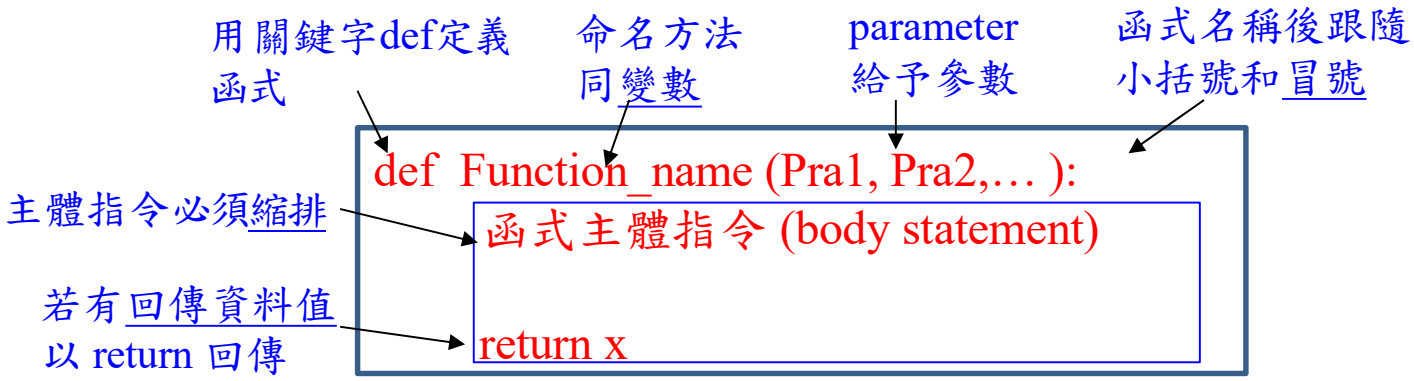
輸出:汽車

Function 呼叫

- 呼叫function
  - abs(-5)
  - print ('Hello, world')
- 內建函式可以直接使用
  - 長度: len()
  - 排序: sorted() v.s. object.sort()
- 內建轉換型態或建立物件
  - int()
  - str()
  - float( )
  - list()
  - set()

Function Definitions

- 自行設計與定義函式



- Parameter(參數)，說明被呼叫時必須接收什麼資料
- 自訂函式

```
def createCar(wheels, dashboard, engine):
    car = dashboard + wheels + engine
    return car
```

parameter

- 函式必須先定義，才能呼叫使用執行

```
carA = CreateCar(wheels, dashboard, engine)
```

argument

Function Definitions

- 定義function
  - def 開頭
  - 名稱自訂
  - 以冒號: 結尾
  - 函式主體指令要縮排

- function種類
  - 無參數、無回傳值
  - 無參數、有回傳值
  - 有參數、有回傳值　　(傳入資料，處理完後回傳結果)
  - 有參數、無回傳值

Definitions and uses

- 函式定義後，可重複呼叫使用

```
01  def newLine():
02      print('end\n')
03      return              # 函式遇到 return 會返回原呼叫者
04
05  def main():
06      newLine()           # 可重複呼叫使用newLine()
07      newLine()
08      newLine()
09
10  main()
```

無參數、無回傳值

```
end
end
end
```

- 函式可以回傳結果

```
01  def my_input():
02      x = int(input())
03      return x
04
05  s = my_input()
```

無參數、有回傳值: 輸入資料

Parameters and arguments

- 函式可定義輸入參數 (parameter)

```
def printTwice(symbol: str):                #parameter 名稱自訂
    print('%c, %c\n' %(symbol, symbol))
```

有參數、無回傳值

- 呼叫函式時，可提供引數(argument)
  - 引數可以是一個值，或是變數
  - 資料型別須與 parameter 一致

```
01  def printTwice(symbol: str):    #parameter 名字自訂
02      print('%c, %c\n' %(symbol, symbol))
03
04  def main():
05      name = 'A'
06      printTwice(name)
07      printTwice('B')
08
09  main()
```

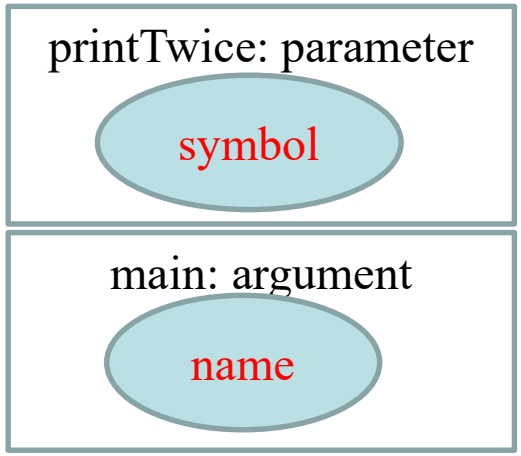
```
A, A

B, B
```

Parameters and variables are local

- 函式定義的參數(parmeter)與變數(variable)，只存在函式內

```
01  def printTwice(symbol: str):     # parameter 名字自訂
02      print('%c, %c\n' %(symbol, symbol))
03
04  def main():
05      name = 'A'
06      printTwice(name)
07      printTwice('B')
08
09  main()
```



Functions with multiple parameters

- 函式可定義多個參數
- 呼叫函式，要輸入相對數量的引數
  - 參數對應順序與資料型別必須一致
  - main函式引數命名，不須與 printTime 參數命名相同
  - main函式引數，與 printTime 參數，兩者存在不同函式，生命週期不同

```
01  def printTime(hour: int, minute: int):
02      print('%d:%d\n' %(hour, minute))
03      return
04
05  def main():
06      hour, _minute = 11, 59
07      printTime(hour, _minute)
08
09
10  main()
```

Function_name (Arg1, Arg2, …)

def Function_name (Pra1, Pra2, …):

傳遞的動作為賦值運算，Pra1 = Arg1,
Pra2 = Arg2
…
PraN = ArgN

Functions with default parameters

- 參數預設值
  - 建立函式時可為參數設定預設值，呼叫函式時，若沒有傳入該參數，就會使用預設值。
  - 參數設定預設值的方法為「參數＝值」

```
01  def get_area(length, width = 4, height = 5):
02      return length*width*height
03
04  print(get_area(3))
05  print(get_area(3,2))
```

Functions with results

- 函式執行完後可以回傳計算結果，也可以不回傳
  - 當執行到 return 指令時，會無條件離開函式，回傳結果
  - 當未使用 return 指令，預設回傳 None
- 回傳值可以為
  - 單一值或物件
  - 多個值或物件所構成的 tuple

```
01  import math
02  def getArea(radius: float)->float:
03      pi = math.acos(-1.0)
04      area = 2*pi*radius
05      return area
06
07  def main():
08      radius = 3
09      print('area=%.3f\n' %getArea(radius))
10
11  main()
```

有參數、有回傳值

area = 18.850

Encapsulation and generalization

- Encapsulation 封裝
  - 函式，是把一段相關的程式指令封裝在一起，完成一個功能

- 設計好的Function
  - 容易閱讀與理解
  - 容易 debug: 每一個function都是獨立小塊程式，容易找出錯誤
  - 方便將不同功能組合成一個大的功能，
  - 大的程式可以切割成許多小塊程式組成
  - 方便遞迴呼叫，或是重複呼叫
  - 一旦充分測試與 debug，往後可以重複使用

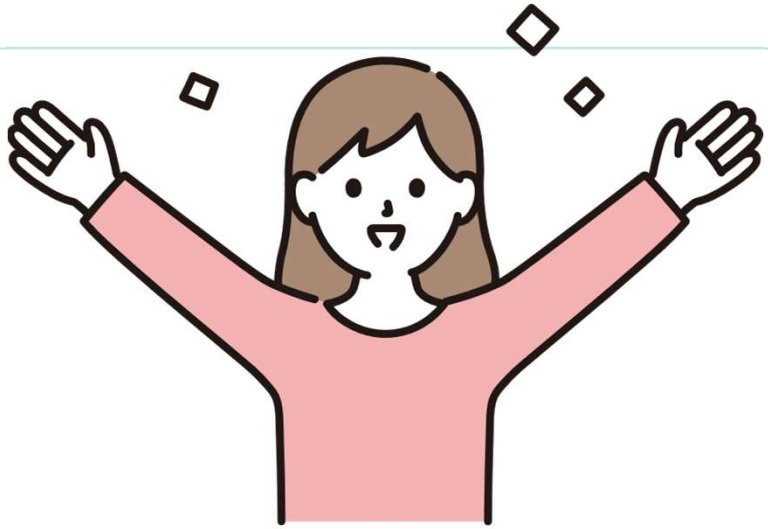
命名規則

- 函數命名應有意義，指出函數目的或回傳資料意義
  - 以駝峰式(Camel-Case)命名、或以小寫或輔以底線命名
  - 避免非慣用縮寫，e.g. def et(): …
- 若由多個單字組成，第一個單字要小寫，第二個單字開始，每個單字的第一個字母大寫
  - studentName, bianryData, dataLength。
- 名稱不使用縮寫，以清楚、明確為原則。
  - 例如 distance 比 d 清楚
  - dataLength 比 length 更明確易理解。
- Boolean 命名 用 is 開頭
  - isPrime
- 函式表示一種操作，使用動詞作為第一個單字。
  - computeAverage()

命名規則

```
01  def a():
02      a = int(input())
03      b = int(input())
04      c = int(input())
05      d = (a+b+c)/3
06  return d
07  # 變數命名無意義
08  # 難以理解
```

```
01  def ComputeAverage():
02          englishGrade = int(input())
03          mathGrage = int(input())
04          chineseGrade = int(input())
05          average = (englishGrade+mathGrage+chineseGrade)/3
06          return average
07
08
```



Exercise

- 神奇算年齡
  - 看一下你手機號碼的最後一位；
  - 把這個數字乘上2；
  - 再加上5；
  - 再乘以50；
  - 把得到的數目加上1770
  - 最後一個步驟，用這個數目減去你出生西元年。
  - 現在你看到一個三位數的數字
  - 第一位數字是你手機號的最後一位，接下來就是你的實際年齡
  - 如果你看到二位數的數字
  - 你手機號的最後一位是0，二位數是你的實際年齡
- 寫成一個function
  - 要傳入那些參數?

Exercise

- 神奇算年齡
  - 不用 function

```
phoneNum, year = 5, 1969
result = ((phoneNum*2 + 5) *50 + 1770) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
```

```
phoneNum, year = 0, 1969
result = ((phoneNum*2 + 5) *50 + 1770) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
phoneNum, year = 3, 1999
result = ((phoneNum*2 + 5) *50 + 1770) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
```

```
5 51
0 51
3 21
```

Exercise

- 神奇算年齡
  - 有用 function

$$(x \times 2 + 5) \times 50 + 1769 - y$$
$$= (2019 - y) + 100x$$

```
def computeAge(phoneNum, year):
    result = ((phoneNum*2 + 5) *50 + 1770) - year
    newPhoneNum = result//100
    age = result%100
    return newPhoneNum, age

def testAge():
    print(computeAge(5, 1969))      # 5, 50
    print(computeAge(0, 1969))      # 0, 50
    print(computeAge(3, 1999))      # 3, 20

testAge()
```

Exercise

- 神奇算年齡
  - 明年改公式，若有一百個地方要改，少改一個?造成程式錯誤!

```
phoneNum, year = 5, 1969
result = ((phoneNum*2 + 5) *50 + 1771) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
```

```
phoneNum, year = 0, 1969
result = ((phoneNum*2 + 5) *50 + 1771) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
```

```
phoneNum, year = 3, 1999
result = ((phoneNum*2 + 5) *50 + 1771) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
```

Exercise

- 神奇算年齡
  - 明年改公式，改一個地方

```
def computeAge(phoneNum, year):
    result = ((phoneNum*2 + 5) *50 + 1770) - year
    newPhoneNum = result//100
    age = result%100
    return newPhoneNum, age

def testAge():
    print(computeAge(5, 1969))      # 5, 50
    print(computeAge(0, 1969))      # 0, 50
    print(computeAge(3, 1999))      # 3, 20

testAge()
```

Exercise

- 神奇算年齡，輸入參數為何? 回傳值為何?

- 另一個算年齡程式

```
from datetime import date

def computeAgeDay(born):
    age = date.today() - born
    return age.days

def computeAge01(born):
    today = date.today()
    return today.year - born.year - ((today.month, today.day) < (born.month, born.day))
```

Exercise

- A、B、C三本書，價格及折扣表如下，一顧客欲購買A: x 本、B: y 本、C: z 本（ x 、 y 、 z 為使用者輸入），請計算需花費多少錢？

	定價	1~10本	11~20本	21~30本	31本以上
A	380	原價	打9折	打8.5折	打8折
B	1200	原價	打9.5折	打8.5折	打8折
C	180	原價	打8.5折	打8 折	打7折

Exercise

- Code

```
x = int(input('A:'))
y = int(input('B:'))
z = int(input('C:'))

if x>=31:
    A_discount = 0.8
elif x>=21:
    A_discount = 0.85
elif x>=11:
    A_discount = 0.9
elif x>=1:
    A_discount = 1
else:
    A_discount = 0
```

```
if y>=31:
    B_discount = 0.8
elif y>=21:
    B_discount = 0.85
elif y>=11:
    B_discount = 0.95
elif y>=1:
    B_discount = 1
else:
    B_discount = 0
```

```
if z>=31:
    C_discount = 0.7
elif z>=21:
    C_discount = 0.8
elif z>=11:
    C_discount = 0.85
elif z>=1:
    C_discount = 1
else:
    C_discount = 0
```

```
cost= x*380*A_discount + y*1200*B_discount + z*180*C_discount
print('The total cost is %d' %(cost))
```

Exercise

- 有使用 function 的 Code

```
# 給購買本數，得到折扣數
def getDiscount(x, discounts):
    discount = 0
    if x>=31:
        discount = discounts[0]
    elif x>=21:
        discount = discounts[1]
    elif x>=11:
        discount = discounts[2]
    elif x>=1:
        discount = discounts[3]
    else:
        discount = discounts[4]
    return discount
```

```
x = int( input('A:') )
y = int( input('B:') )
z = int( input('C:') )

# 計算折扣
A_discounts = [ 0.8, 0.85, 0.90, 1.0 ]
B_discounts = [ 0.8, 0.85, 0.95, 1.0 ]
C_discounts = [ 0.7, 0.80, 0.85, 1.0 ]

A_discount = getDiscount(x, A_discounts)
B_discount = getDiscount(y, B_discounts)
C_discount = getDiscount(z, C_discounts)
```

```
cost = x * 380 * A_discount + y * 1200 * B_discount + z * 180 * C_discount
print('The total cost is %d' %(cost))
```

Exercise

- A、B、C三本書價格及折扣表如下，一顧客欲購買A: x 本、B: y 本、C: z 本（ x 、 y 、 z 為使用者輸入），請計算需花費多少錢？
  - 幾本區間是否可以輸入? 1~10 => 1~20
  - 定價表格(區間個數)是否可以輸入?

	定價	1~10本	11~20本	21~30本	31本以上
A	380	原價	打9折	打8.5折	打8折
B	1200	原價	打9.5折	打8.5折	打8折
C	180	原價	打8.5折	打8 折	打7折

Exercise

- 檢查三門課程是否衝堂，每一門課上課2小時
  - 依序輸入課程編號(數字)、上課時間(星期1-5, 第1-9節)

Input	說明	Output	
1001 59 25 2020 25 59 2030 11 25	(第一門課課程編號) (星期5 第9節課) (星期2 第5節課) (第二門課課程編號) (星期2 第5節課) (星期5 第9節課) (第三門課課程編號) (星期1 第1節課) (星期2 第5節課)	1001 and 2020 conflict on 25	(兩課程編號衝突在哪幾節)

Exercise (Encapsulation 封裝)

parameter annotations 註解

```
01  def match(section1: int, section2: int, result: list, index: int,) -> int:
02      if section1!='99' and section1== section2:
03          result[index] = section1
04          index = index + 1
05      return index
06
07  def isConflict(course1:list, course2: list):
08      result = [0, 0]
09      sections = ['99', '99']
10      index = 0
11      index = match(course1[1], course2[1], sections, index)
12      index = match(course1[1], course2[2], sections, index)
13      index = match(course1[2], course2[1], sections, index)
14      index = match(course1[2], course2[2], sections, index)
15      if index > 0:
16          sections = sorted(sections)
17          c_ids = [course1[0], course2[0]]
18          c_ids = sorted(c_ids)
19          result = [c_ids, sections]
20
21      return (result)
```

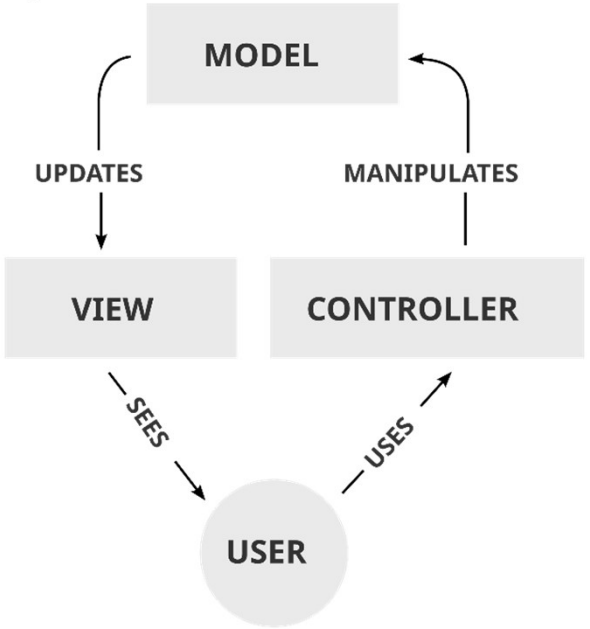
return annotation

Exercise (Encapsulation 封裝)

```
22  def getData():
23      course = [0, 0, 0]
24      course[0] = input()
25      course[1] = input()
26      course[2] = input()
27      return course
28
29  def printResult(result):
30      print(result[0][0],result[0][0], result[1][0], result[1][1])
31
32  def compute():
33      course1 = getData()
34      course2 = getData()
35      course3 = getData()
36      courses = [course1, course2, course3]
37      result1 = isConflict(courses[0], courses[1])
38      result2 = isConflict(courses[0], courses[2])
39      result3 = isConflict(courses[1], courses[2])
40      results = [result1, result2, result3]
41      results = sorted(results)
42      printResult(results[0])
43      printResult(results[1])
44      printResult(results[2])
```

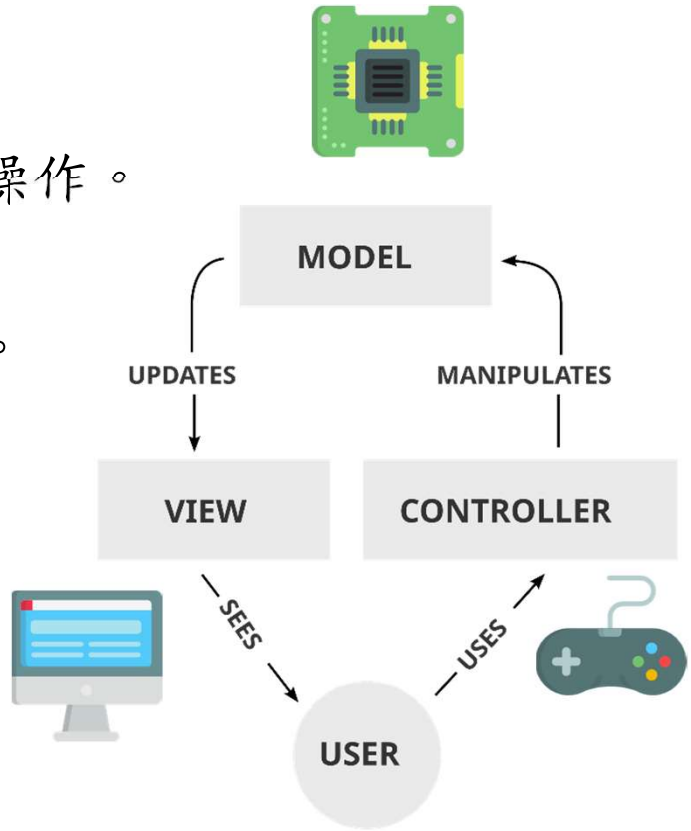
Exercise

- 程式要切割模組
  - 好的設計
  - MVC架構
    - 模型(Model)、視圖(View)和控制器(Controller)
    - 軟體工程中的一種軟體架構模式
    - 把軟體系統分為三個基本部分



Exercise

- MVC簡化複雜度，使程式結構更直覺。
  - User 透過介面發出請求。
  - Controller 截獲使用者發出的請求。
  - Controller 呼叫 Model 完成狀態的讀寫操作。
  - Controller 把資料傳遞給 View。
  - View 渲染最終結果，並呈獻給使用者。

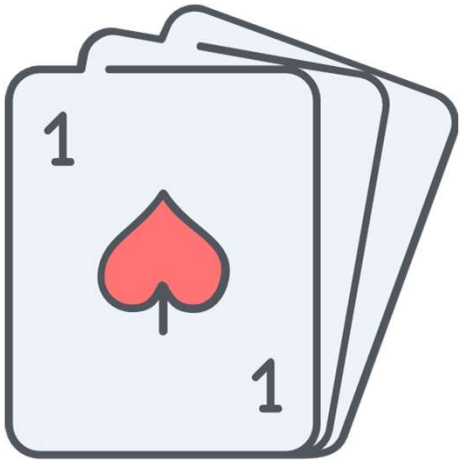


Exercise

- 將計算成績練習，寫成三個function
  - input
  - compute
  - output

Exercise

- 撲克牌
  - A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
  - A~10　　點數為 1~10，
  - J, K, Q　　點數為 0.5。



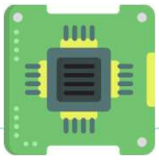
- X, Y 兩個人
  - 各發三張撲克牌，
  - 加總點數接近 10.5 則贏。
  - 超過 10.5 爆掉分數為 0 。
- 程式
  - 輸入 X, Y 兩個人各發的三張撲克牌。
  - 輸出兩個人的點數，以及A贏或B贏或平手。

Exercise

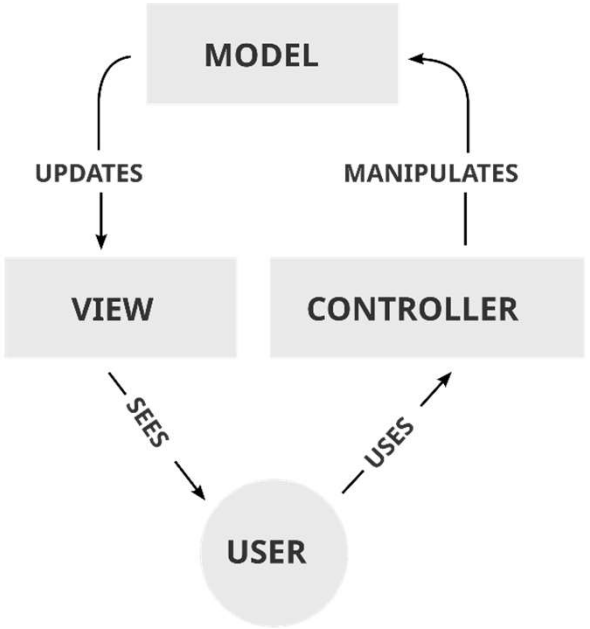
- 寫出程式流程/步驟
  - 1.輸入牌面符號>2.轉換點數>3.加總點數(調整點數)>4.比大小
- 設計function 和參數傳遞介面

```
def transferPoint(card):
    pork = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
    points =[1,2,3,4,5,6,7,8,9,10,0.5, 0.5,0.5]
    index = pork.index(card)
    return points[index]
```

```
def getSum(x, y, z):
    a = transferPoint(x)
    b = transferPoint(y)
    c = transferPoint(z)
    aPoint = a + b + c
    if aPoint > 10.5:
        aPoint = 0
    return (aPoint)
```



```
def compare(A, B):
    if A > B:
        print('A win')
    elif A < B:
        print('B win')
    else:
        print('Tie')
```

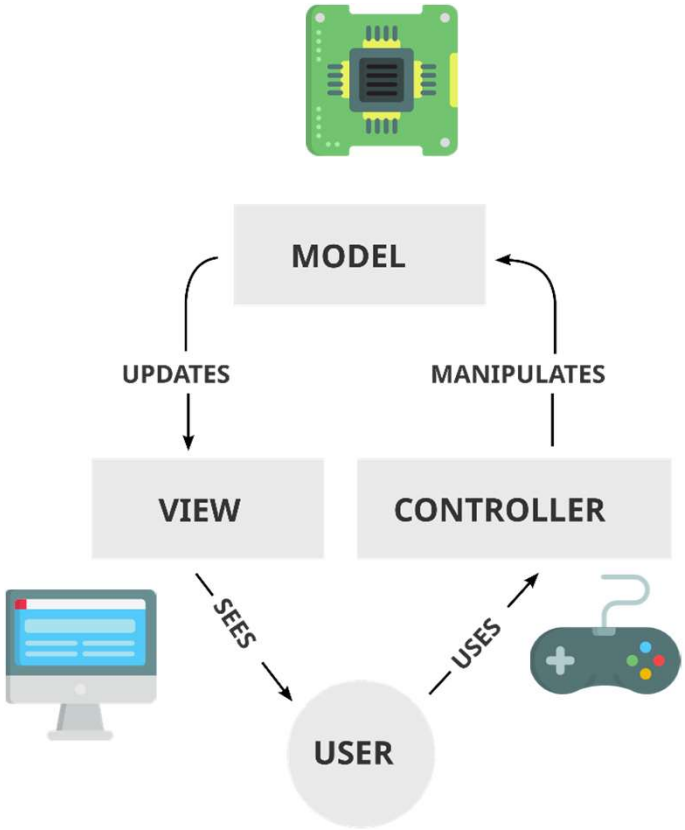


Exercise

- 組合function測試
  - 設計多組資料測試

```
def test():
    x1, x2, x3 = '10', 'Q', 'A'
    y1, y2, y3 = '10', 'K', 'J'
    aPoint = getSum(x1, x2, x3)
    bPoint = getSum(y1, y2, y3)
    compare(aPoint, bPoint)

test()
```



Exceptions Handling

```
def my_divide():
    try:
        10 / 0 #會讓程式出錯,所以需要特別handle
    except ZeroDivisionError:
        print('不能除以0!!!')
    else:
        print('沒有任何錯誤')
    finally:
        print('無論有沒有例外都會執行這一行')

my_divide()
```

Output:

```
不能除以0!!!
無論有沒有例外都會執行這一行
```

Exceptions Handling

- 開啟檔案引發例外情況

```
def computeAge02(born):
    try:
        fp = open("C:\test.txt","r")
        try:
            for index in fp:
                print(index)
        except:
            print("read file error")
        finally:
            fp.close()
    except:
        print("Open error")
```

Exceptions Handling

- 自行引發例外情況

```
def computeAge02(born):
    try:
        raise EOFError #自行引發的例外
    except EOFError:
        print("self error")
    else:
        print("No error")
```

Exercise

- 輸入兩個整數計算相加、相除，輸出相加與相除結果，
  - 如果輸入不是整數，例如字串，輸出 "Input Integer"，
  - 如果第二個數是0，輸出 "divide by zero"，
  - 最後一定要輸出 "OK"，
  - 請使用 try, except, finally。

```
num1 = input()
num2 = input()
try:
    num1 = int(num1)
    num2 = int(num2)
    answer = num1/num2
except ValueError:
    print('Input Integer')
except ZeroDivisionError:
    print('Division zero')
else:
    print(answer)
finally:
    print('OK')
```

import

- 用 import 函式庫
  - 直接匯入整個python函式庫中所有函式，
- 用 from 函式庫 import 函式
  - 插入特定函式
- 用 import x.py
  - 會執行 x.py 未空四格的指令
  - 一般函式庫不會有

```
#匯入sys函式庫所有函式，使用write函式前須加sys
import sys

#從time函式庫匯入time()函式，使用time()前不需加 time
from time import time
sys.stdout.write(str(time())+"\n")
```

Output:
1409796132.99 #當下的time

import

- module
  - 一個包含有 Python 的程式定義及敘述的檔案。
  - 檔案名稱是 module 名稱加上延伸檔名 .py 。
  - module 裡存在 module名字變數，如 sys, time，當作全域變數 (global variable)使用。

- 常見module
  - math 模組：　　數學運算相關模組
  - sys 模組：　　作業系統的相關資訊與函數的模組
  - time 模組：　　時間相關模組
  - re 模組：　　正則表達式 (Regular Expression)

import

- __main__
  - A模組/檔案，直接執行時，__name__的值預設是【'__main__'】

```
# A.py
if __name__ == '__main__':
    print('Direct running')
else:
    print('import __name__ = ', __name__)
```

Direct running

  - A模組/檔案，被import後，__name__的值是，檔案名稱
    - 執行 B.py

```
#B.py
import A
print('import A')
```

import __name__ = A
import A

內建Function

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

END