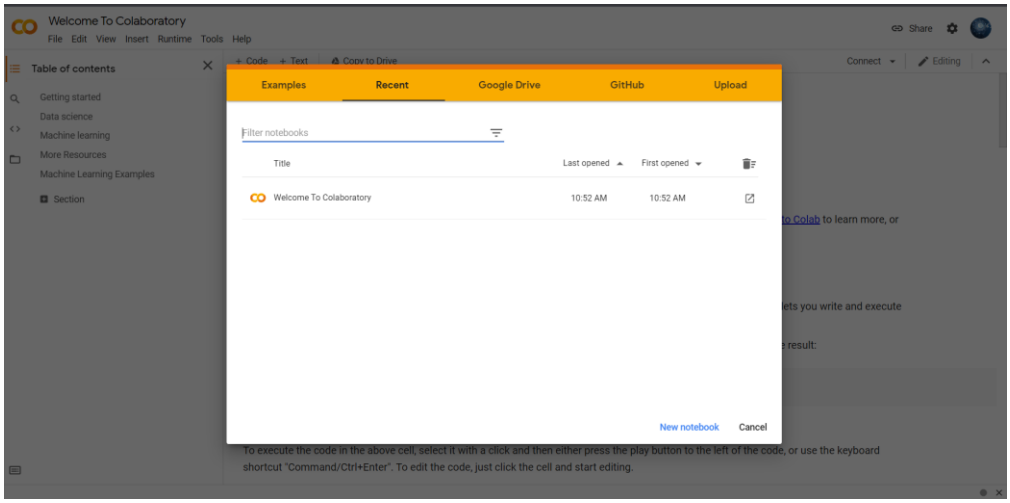
Thực hành: PYTHON CĂN BẢN (phần 1)

1. Sử dụng Colab để viết và thực thi chương trình

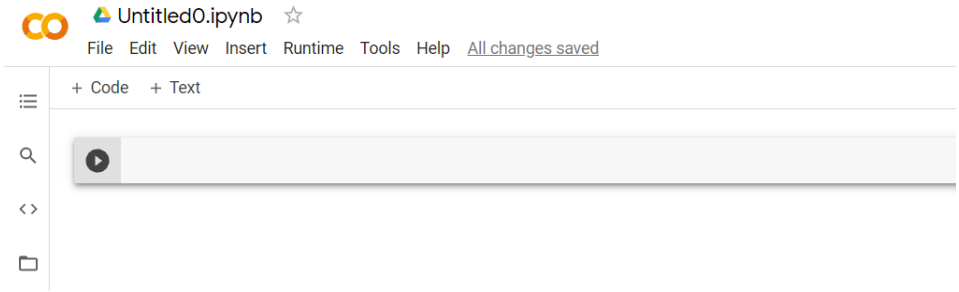
https://colab.research.google.com/

sử dụng email student hoặc tài khoản google để đăng nhập



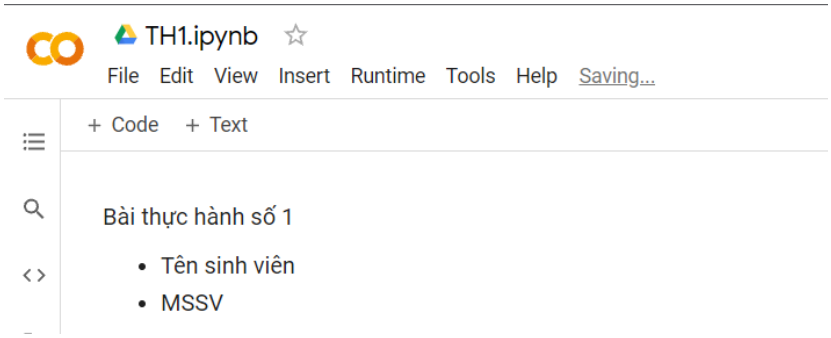
Tạo một notebook mới. Đặt tên THB1_MSSV_Hoten

2. Viết code trong notebook



- Double click vào tên để rename lại tên mới (không xóa hoặc điều chỉnh phần mở rộng .ipynb)

- Chọn +Text để thêm phần mô tả



- Chọn +Code thể thêm code cell dưới tên

- Chọn Run (nút play) để thực thi

3. Toán tử

Các toán tử thường sử dụng

Toán tử	Input	Output
+	123 + 345	
-	93 – 3.5	
*	10 * 10	
/	100 / 3	
//	93 // 10	
**	8 ** 3	
%	93 % 10	

???Câu hỏi: Cho biết phần output?

4. Một số hàm cơ bản

Hàm	Input	Output
abs()	abs(-5)	
max()	max(4, 6, -2, 5, 10, 2)	
min()	min(4, 6, -2, 5, 10, 2)	
print()	print("Bai thuc hanh so 1")	
round()	round(3.43264, 3)	

Tham khảo thêm các hàm tại: https://docs.python.org/3.7/library/functions.html

??Câu hỏi: output của các hàm trên là gì? Mô tả các hàm trên (sử dụng để làm gì)

5. Biến và Kiểu dữ liệu:

Không cần khai báo biến trước khi sử dụng.

Tránh đặt tên biến trùng từ khóa của python

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

Để xem kiểu dữ liệu: type()

```
print(type(3))
print(type(3.5))
print(type(True))
print(type("Hello"))
```

```
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'str'>
```

a). Kiểu số: int, float

Để in định dạng số: sử dụng .format()

```
luong = 6623000
print("${:,.0f}".format(luong))
```

```
$6,623,000
```

Định dạng của chuỗi: bắt đầu với $, dùng dấu ‘,’ để phân tách hàng ngày (**:,**), không có phần thập phân (**.0f**)

Một số ví dụ khác

```
print("{:.2f}".format(3400.1234))
print("{:,.2f}".format(3400.1234))
print("{:+,.2f}".format(3400.1234))
print("{:-,.2f}".format(3400.1234))
print("{:.2%}".format(0.25))
```

```
3400.12
3,400.12
+3,400.12
3,400.12
25.00%
```

b). Kiểu logic: bool

```
t, f = True, False
print(t and f)
print(t or f)
print(not t)
print(1 == 2)
```

```
False
True
False
False
```

c). Kiểu chuỗi: string (str)

Dữ liệu được đặt trong dấu ngoặc kép “” hay ngoặc đơn ‘’

```
hello = 'hello'
world = "world"
hw = hello + " " + world
print(hw)
```

```
hello world
```

Có thể in nhiều chuỗi phân tách nhau bằng một ký tự nào đó. Ví dụ 2 chuỗi được in nối nhau phân tách bằng ký tự #

```
print("Chuoi 1", "Chuoi 2", sep="#")
```
```
Chuoi 1#Chuoi 2
```

Định dạng chuỗi

```
hello = 'hello'
world = "world"
hw = '{} {} {}'.format(hello, world, 12)
print(hw)
```
```
hello world 12
```

Để biết giá trị ascii của một ký tự: ord(ký tự)

```
print("Gia tri ASCII cua 'a' la {}, cua 'A' la {}".format(ord("a"), ord('A')))
```
```
Gia tri ASCII cua 'a' la 97, cua 'A' la 65
```

Một số hàm sử dụng cho chuỗi: capitalize, upper, lower, rjust, center, replace, strip…

?? Câu hỏi: cho biết kết quả của đoạn chương trình trên và ý nghĩa của các hàm

6. Chuyển đổi kiểu

```
# Chuyển chuỗi sang số
a = int("5")
print(a, type(a))
# Chuyển chuỗi sang số
a = float("5.3")
print(a, type(a))
# Chuyển một đối số sang chuỗi
a = str(5)
print(a, type(a))
a = str(True)
print(a, type(a))
# Chuyển một đối số sang bool
a = bool(0)
b = bool(-1)
print(a, b)
```
```
5 <class 'int'>
5.3 <class 'float'>
5 <class 'str'>
True <class 'str'>
False True
```

7. Đọc dữ liệu input

```
name = input("Nhap vao ten: ")
print(name)
```
```
Nhap vao ten:
```

8. Containers (phần 1)

Python cung cấp số kiểu container dựng sẵn: list, dictionary, set, tuple

8.1. Tuple:

- Có thể lưu trữ các kiểu dữ liệu khác nhau trong cùng cấu trúc.
- Không thể cập nhật giá trị trong tuple
- Một tuple có chiều dài và kích thước cố định

```
t = (1, 2.3, 'hello', False, (2, 3))
print(t)
print(t[1])
print(t[1:3])
```

```
(1, 2.3, 'hello', False, (2, 3))
2.3
(2.3, 'hello')
```

8.2 List:

- Có thể thay đổi được giá trị trong biến

- Python sử dụng list thay cho array

Ví dụ tạo list và truy cập vào phần tử

```
listSV = ['Nguyen Van A', 23, 3.5]
print(listSV)
print(listSV[0])
print(listSV[1])
print(listSV[2])
```

```
['Nguyen Van A', 23, 3.5]
Nguyen Van A
23
3.5
```

- Cắt list: truy cập một tập cong dữ liệu của list:

+ Cú pháp: phần từ đầu tiên muốn lấy : index của phần tử cuối + 1

```
listMoney = [12, 34, 2, 14, 45, 56]
print(listMoney[1:5])
```

```
[34, 2, 14, 45]
```

- Thay đổi giá trị

```
listMoney = [12, 34, 2, 14, 45, 56]
listMoney[1] = 30
print(listMoney)
```

```
[12, 30, 2, 14, 45, 56]
```

```
listMoney = [12, 34, 2, 14, 45, 56]
listMoney[1:3] = [30, 23]
print(listMoney)
```

```
[12, 30, 23, 14, 45, 56]
```

- Thêm vào phần tử mới

```
listMoney = [12, 34, 2, 14, 45, 56]
listMoney += [100]
print(listMoney)
listMoney += [0, 21]
print(listMoney)
listMoney.append(15)
print(listMoney)
listMoney.insert(2, 1)
print(listMoney)
```

```
[12, 34, 2, 14, 45, 56, 100]
[12, 34, 2, 14, 45, 56, 100, 0, 21]
[12, 34, 2, 14, 45, 56, 100, 0, 21, 15]
[12, 34, 1, 2, 14, 45, 56, 100, 0, 21, 15]
```

- Xóa phần tử khỏi list

```
listMoney = [12, 34, 2, 14, 45, 56]
del listMoney[3] # vị trí của phần tử muốn xóa
print(listMoney)
listMoney.remove(45) # giá trị cần số
print(listMoney)
a = listMoney.pop(1) # vị trí của phần tử cần lấy ra, trả về giá trị
print(listMoney)
print(a)
```

```
[12, 34, 2, 45, 56]
[12, 34, 2, 56]
[12, 2, 56]
34
```

- list comprehension: giúp viết code ngắn gọn để lọc hoặc chỉnh sửa danh sách dựa trên một tiêu chí

Ví dụ:

```
#Tạo danh sách chứa các từ viết hoa của input
ds1 = ['a', 'b', 'c', 'd']
ds2 = [x.upper() for x in ds1]
print(ds2)
```

```
['A', 'B', 'C', 'D']
```

8.3 Dictionary:

Kiểu dữ liệu dictionary là kiểu lưu trữ các giá trị chứa key và value.

Khai báo bằng cách sử dụng dấu {} theo cú pháp:

`{key1: value1, key2: value2, … , keyN: value N}`

- key: có thể số hoặc chuỗi; phải là duy nhất; đã khai báo thì không thể đổi tên

```
sv = {'ten': 'Trần Lâm', 'tuoi': 23, 'nam': True}
print(sv)
print(sv['ten'])
# Cập nhật giá trị
sv['ten'] = "Trần Thị Lan"
sv['nam'] = False
# Thêm mục mới
sv.update({'luong': 20000})
print(sv)
# Xóa một mục
del sv['nam']
sv.pop('luong')
print(sv)
```

```
{'ten': 'Trần Lâm', 'tuoi': 23, 'nam': True}
Trần Lâm
{'ten': 'Trần Thị Lan', 'tuoi': 23, 'nam': False, 'luong': 20000}
{'ten': 'Trần Thị Lan', 'tuoi': 23}
```

Duyệt các phần tử trong dictionary

```python
sv = {'ten': 'Trần Thị Lan', 'tuoi': 23, 'nam': False, 'luong': 20000}
# Dyệt các key
for x in sv:
  print(x, end = " ")
print()
for k in sv.keys():
  print(k, end=" ")
print()
# Duyệt các value
for v in sv.values():
  print(v, end=" ")
print()
# Duyệt các cặp key, value
for k, v in sv.items():
  print(k, v)
```

```
ten tuoi nam luong
ten tuoi nam luong
Trần Thị Lan 23 False 20000
ten Trần Thị Lan
tuoi 23
nam False
luong 20000
```

8.4. Set

Set là một collection không thứ tự, không có chỉ mục, không cho phép chứa dữ liệu trùng lặp.

Set được khai báo với dấu ngoặc nhọn {}

```python
taphop = {'sach', 'but bi', 'but chi', 'thuoc', 'phan', 'thuoc', 'sach'}
print(taphop)
taphop = set(('CSDL', 'TTNT', 'Toan Roi Rac', 'KTLT'))
print(taphop)
# duyệt các phần tử trong set
for x in taphop:
  print(x, end=' ')
# kiểm tra 01 phần tử có trong set không
print('CSDL' in taphop)
# Thêm vào set
taphop.add('Lap trinh web')
taphop.update({'Java 1', 'Java 2'})
print(taphop)
# Xóa phần tử
taphop.remove('Java 1')
taphop.discard('Java 2')
# Xóa tất cả phần tử
taphop.clear()
print(taphop)
```

9. So sánh và điều kiện

- Toán tử so sánh: <, <=, >, >=, ==, !=

- Toán tử điều kiện: or, and, not

- Câu lệnh if

```
if điều kiện:
     thực hiện các câu lệnh trong if
elif điều kiện:
     thực hiện các câu lệnh
else:
     thực hiện các câu lệnh
```

10. Cấu trúc điều khiển

Cấu trúc for:

```
for biến_lặp in chuỗi_lặp:
    Khối lệnh của for
```

Ví dụ:

```
ds1 = ['abc','def', 'ghi']
for x in ds1:
  print(x)

str1 = 'abc'
for x in str1:
  print(x)
```

```
ds2 = []
for i in range(5):
  ds2.append(i)

ds3 = []
for i in range(1, 5):
  ds3.append(i)

ds4 = []
for i in range(0, 5, 2):
  ds4.append(i)

print('ds2 ', ds2)
print('ds3 ', ds3)
print('ds4 ', ds4)
```

Cấu trúc while

```
while điều kiện:
    thực hiện câu lệnh
```

Cấu trúc for

Lệnh for có thể duyệt tuần tự các đối tượng (list, tuple, string) và các đối tượng có thể lặp

```
listMoney = [12, 34, 2, 14, 45, 56]
for x in listMoney:
  print(x, end=', ')

print()
hello = "hello"
for x in hello:
  print(x)

for i in range(len(listMoney)):
  print(i, listMoney[i])
```

11. Định nghĩa hàm

```
def tên_hàm(tham_số, tham_số…):
    thực hiện  lệnh
```

```
    return giá trị # nếu cần
```

12. Lớp/đối tượng

Tạo lớp: sử dụng từ khóa class

```
class SinhVien:
  ten = 'Trang'
  tuoi = 20
  diem = 10

sv1 = SinhVien()
print(sv1.ten)
```

Hàm __init__(): mọi lớp đề có hàm này; hàm được thực hiện khi lớp khởi tạo.

Tham số seft: tham chiếu đến thể hiện hiện tại của một lớp, được sử dụng để truy cập biến thuộc về lớp.

```
class SinhVien:
  ten = 'Trang'
  tuoi = 20
  diem = 10
  def __init__(self, t, tu, d):
    self.ten = t
    self.tuoi = tu
    self.diem = d
  def inTT(self):
    print('Sinh viên {}, {} tuoi dat {} diem'.format(self.ten,
                                                     self.tuoi, self.diem))
  def congDiem(self, c):
    self.diem += c

sv1 = SinhVien('Ha', 21, 9)
sv1.inTT()
sv1.congDiem(1)
sv1.inTT()
sv1.diem -= 2
sv1.inTT()
```

13. Class/Object

Định nghĩa một lớp

```
class name:
    <<khối lệnh trong class>>
```

Thuộc tính: được khai báo trực tiếp trong lớp hoặc trong hàm khởi tạo

```
class People:
  age = 10
  name = 'An'
```

Phương thức:

```
def name(self, para_1,…, para_n):
    <<khối lệnh>>
```

- `self`: là tham số đầu tiên của các phương thức

- truy xuất các giá trị của thuộc tính thông qua tham chiếu `self`

```
class People:
  ...
  def rename(self, n):
    self.name = n
  ...
```

Gọi phương thức: bằng 2 cách

```
object.method (para1,…)
```

hoặc

```
Class.method (object, para1, …)
```

```
class People:
  ...
  def rename(self, f, l):
    self.first_name = f
    self.last_name = l
  ...

p = People('Trần', 'Lý')
# Cách 1
p.rename('Nguyễn', 'Phi')
# Cách 2
People.rename(p,'Lý', 'Như')
```

Hàm khởi tạo

```
def __init__(self, para1, …, paran):
     <<khối lệnh>>
```

```
class People:
  def __init__(self, f, l):
    self.first_name = f
    self.last_name = l
```

Phương thức: __str__

```
def __str__(self):
     return string
```

được gọi tự động khi `str` hoặc `print` được gọi

```
class People:
  def __init__(self, f, l):
    self.first_name = f
    self.last_name = l

  def __str__(self):
    return '(' + self.first_name + ', ' + self.last_name + ')'
```

Nạp chồng toán tử

Toán tử	Phương thức	Toán tử	Phương thức
+	__neg__(self, other)	==	__eq__(self, other)
-	__pos__(self, other)	!=	__ne__(self, other)
*	__mul__(self, other)	<	__lt__(self, other)
/	__truediv__(self, other)	>	__gt__(self, other)
-	__neg__(self)	<=	__le__(self, other)
+	__pos__(self)	>=	__ge__(self, other)

Thừa kế

```
class name(superclass):
    <<khối lệnh>>
```

```
class Teacher(People):
  course = 'Decision Support Systems'
```

- Gọi phương thức lớp cha

```
class.method(object, paras…)
super().method(paras…)
```

```
class Teacher(People):
  course = 'Decision Support Systems'
  degree = 'Dr.'
  def __init__(self, f, l, c, d):
    # People.__init__(self, f, l)
    super().__init__(f, l)
    self.course = c
    self.degree = d

  def reprofile(self, f, l, c, d):
    # People.rename(self, f, l)
    super().rename(f, l)
    self.course = c
    self.degree = d
```

Public, private, protected:

- Mục tiêu của python là cung cấp một ngôn ngữ linh hoạt. Vì vậy tính bao đóng của Python sẽ không nghiêm ngặt như các ngôn ngữ khác. Tuy nhiên, Python khuyến nghị một số quy ước; nếu tuân theo sẽ duy trì tính bao đóng cho các đối tượng

+ Private: không thể gọi bên ngoài lớp, có ít nhất hai dấu gạch dưới ở đầu và tối đa một dấu gạch dưới ở cuối

Ví dụ: `__foo, __bar_()`

+ Protected: một dấu gạch dưới ở đầu; không nên truy cập bên ngoài lớp

+ Public

```python
class Animal:
  leg = 4
  _eat = 'grass'
  __fur = False
  def __init__(self, l, e, f):
    self.leg = l
    self._eat = e
    self.__fur = f

# Phù hợp
print("Animal's leg: {}".format(Animal.leg))
a = Animal(4, 'meat', True)
print("Animal's leg: {}".format(a.leg))
# Vẫn hoạt động nhưng không khuyến khích
print("Animal eat {}".format(Animal._eat))
print("Animal eat {}".format(a._eat))
# Lỗi
print("Fur: {}".format(Animal.__fur))
print("Fur: {}".format(a.__fur))
```

BÀI TẬP THỰC HÀNH 1

1. Yêu cầu người dùng nhập một số nguyên. Kiểm tra số nhập vào là chẵn hay lẻ và in ra màn hình

2. Yêu cầu nhập vào 02 số (a và b). In thông báo là a có chia hết cho b không.

3. Viết chương trình tìm tất cả các số từ 200 đến 250 chia hết cho 7 nhưng không chia hết cho 5 lưu vào 1 danh sách. In danh sách ra màn hình.

Gợi ý: sử dụng hàm range(#begin, #end) với #begin = 200, #end=251

4. Viết một hàm tính giai thừa của một số nhập từ bàn phím.

5. Nhập vào 1 câu và kiểm tra trong câu đó có bao nhiêu chữ cái và bao nhiêu số. In ra kết quả.

Ví dụ: nếu câu là hello world! 123 thì kết quả hiển thị là 10 chữ cái và 3 số

Gợi ý: sử dụng hàm isalpha() và isdigit()

```
x = 'a'
print(x.isalpha(), x.isdigit())
x = '1'
print(x.isdigit(), x.isalpha())
```

```
True False
True False
```

6. Viết chương trình:

- Yêu cầu người dùng nhập vào một chuỗi

- In chuỗi đó ra màn hình

- In chuỗi đảo ngược ra màn hình

Ví dụ:

Đầu vào: My name is Mary

In ra màn hình: Mary is name My

Gợi ý:

- Hàm split: trách chuỗi tại ký tự phân tách được chỉ định và trả về một danh sách

```
chuoi = "xin chao! Toi ten la An"
words = chuoi.split(" ")
print(words)
```

```
['xin', 'chao!', 'Toi', 'ten', 'la', 'An']
```

- Hàm join: Nối các phần tử trong tuple/list thành một chuỗi, sử dụng ký tự như dấu phân tách

```
list_str = ['a', 'b', 'c', 'd']
chuoi = ' '.join(list_str)
print(chuoi)
chuoi = '#'.join(list_str)
print(chuoi)
```

```
a b c d
a#b#c#d
```

7. Viết chương trình in ra tam giác Floyd.

Tam giác Floyd là một tam giác vuông được tạo từ các số tự nhiên. Các số trong tam giác Floyd có giá trị tăng dần. Ví dụ 1 tam giác Floyd

1

2 3

4 5 6

Chương trình yêu cầu người dùng nhập vào số dòng và chương trình in ra tam giác Floyd với số dòng tương ứng.

8. Viết chương trình nhận vào chuỗi các số phân tách nhau bằng dấu phẩy từ bàn phím và trả về 1 list và 1 tuple chứa các số đó

Ví dụ:

Input: 12,13,14

Output: [12,13,14] và (12,13,14)

9. Kết hợp 2 danh sách theo chỉ mục

Ví dụ:

```
input:
l1 =  ['M', 'na', 'i', 'Jo']
l2 =  ['y', 'me', 's', 'hn']
output
l3 = ['My', 'name', 'is', 'John']
```

Gợi ý:

Hàm `zip()` trả về đối tượng là iterator của các tuple kết hợp các phần tử từ các iterator khác. (Iterator là các đối tượng cho phép lấy từng phần của nó, có thể được lặp đi lặp lại).

Ví dụ:

```
list_so1 = [1, 2, 3, 4]
list_so2 = [5, 6, 7, 8]
for x in zip(list_so1, list_so2):
  print(x)
```

```
(1, 5)
(2, 6)
(3, 7)
(4, 8)
```

10. Ghép 02 danh sách theo thứ tự

```
input:
l1 =  ['Hello', 'Dear']
l2 =  ['Peter', 'Mary']
output: ['Hello Peter', 'Hello Mary', 'Dear Peter', 'Dear Mary']
```

11. Cho người dùng nhập vào một dãy số từ bàn phím, lưu dãy số đó vào danh sách và in ra theo chiều đảo ngược

- Sử dụng input để người dùng nhập vào dãy số

- Sử dụng split để tách chuỗi

- Sử dụng int để chuyển chuỗi thành số

- Lưu vào danh sách

- Sử dụng chỉ số -1 để in theo chiều đảo ngược

```
ds = [2, 5, 6, 9, 10, 13, 14, 17]
print(ds[::-1])
```

```
[17, 14, 13, 10, 9, 6, 5, 2]
```

12. Viết chương trình tính số tiền ròng của tài khoản ngân hàng dựa trên nhật ký giao dịch. Định dạng nhật ký giao dịch như sau: G 100 R 200.

Ví dụ: G 500 G 100 R 200 G 100 thì kết quả hiển thị là 500

13. Mô phỏng trò chơi Kéo-Búa-Bao

- Yêu cầu người dùng chọn một số ứng với Kéo/Búa/Bao

- Chương trình sinh một số ngẫu nhiên tương ứng với Kéo/Búa/Bao

- So sánh với luật trò chơi và thông báo người chơi thắng/thua/hòa

Gợi ý:

Sử dụng module random để sử dụng hàm dựng sẵn choice trong python

```
import random
ds = [1, 2, 3]
a = random.choice(ds)
print(a)
```

```
3
```

14. Viết 1 lớp Circle: có thuộc tính là radius, 02 phương thức calArea và calPerimeter tính diện tích và chu vi của hình tròn.

Gợi ý: lấy hằng số pi

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

```
3.141592653589793
```

15. Viết chương trình chuyển đổi 02 danh sách cho trước thành 1 dictionary với danh sách được sử dụng làm khóa và danh sách 2 được sử dụng làm giá trị

Ví dụ:

lst1 = [1,2,3,4]

lst2 = [‘a’, ‘b’, ‘c’, ‘d’]

dct = {1: ‘a’, 2: ‘b’, 3: ‘c’, 4: ‘d’}

16. Cho một dict như sau

inventory = {'gold':500, 'pouch':['flint', 'twine', 'gemstone'], 'backpack': ['xylophone','dagger', 'bedroll','bread loaf']}

Thực hiện các yêu cầu sau:

- Thêm khóa pocket vào inventory. Giá trị của 'pocket' là một danh sách gồm các chuỗi 'seashell', 'strange berry' và 'lint'.

- Sử dụng .sort() để sắp xếp các item trong danh sách là giá trị ứng với khóa 'backpack'.

- Sau đó, sử dụng .remove('dagger') khỏi danh sách được lưu trữ dưới khóa 'backpack'.

- Cộng 50 vào giá trị ứng với khóa 'gold'.

17. Viết đoạn chương trình minh họa cho trò chơi đoán số.

- Hệ thống sinh một số nguyên ngẫu nhiên trong khoảng 1 đến 100.

- Người dùng nhập vào số đã đoán

- Nếu số đoán > số hệ thống => Thông báo “Nhỏ hơn”, ngược lại thông báo “Lớn hơn”

- Nếu số đoán = số hệ thống => Thông báo kết quả

Gợi ý sử dụng random.randint(#begin, #end) để sinh số ngẫu nhiên

https://www.w3schools.com/python/ref_random_randint.asp

18. Lớp Product:

- Các thuộc tính: name (string), price (float), weight (float), brand (string), status (string)

- Các phương thức:

+ Khởi tạo (trong đó status có giá trị mặc định là *for sale*)

+ sell: gán status thành sold

+ tax: trả về giá bao gồm thuế với mức thuế là giá trị đầu vào

Công thức: price + tax * price

+ ret: có tham số dạng chuỗi và giá trị của tham số sẽ quyết định giá trị của status và price

Nếu ‘defective’ thì status là ‘defective’ và price bằng 0

Nếu ‘in box’ thì status là for sale

Nếu ‘opened’ thì status là used và price giảm 20%

+ displayProduct: in ra thông tin của Product

19. Lớp Store

- Thuộc tính: products (danh sách các Product), location (string), owner (string)

- Phương thức:

+ add_product: thêm một Product vào danh sách products

+ remove_product: xóa một Product có name chỉ định ra khỏi products

+ inventory: hiển thị thông tin của các Product trong Store

20. BenhNhan và BenhVien

BenhNhan có các:

- Thuộc tính: id, ten, trieuchung, sogiuong (sogiuong mặc định là ‘none’)

- Phương thức: printInfo (in thông tin của BenhNhan)

BenhVien:

- Thuộc tính: tenBV, suc_chua (số lượng BenhNhan có thể điều trị nội trú), dsBN (danh sách BenhNhan)

- Khởi tạo: gán tên, sức chứa bằng các giá trị truyền vào, danh sách BN rỗng

- NhapVien (BenhNhan):

+ nếu số lượng bệnh nhân trong danh sách đã lớn hơn hoặc bằng sức chứa thì thông báo không thể thực hiện

+ nếu số giường của bệnh nhân là none thì thêm vào trong danh sách và gán index của bệnh nhân vừa thêm vào danh sách cho bed

+ ngược lại: thông báo số giường của bệnh nhân đã được lưu trong danh sách

- XuatVien(BenhNhan)

+ Nếu số giường là ‘none’: in thông báo bệnh nhân không điều trị nội trú

+ Ngược lại: xóa phần tử trong danh sách và thông báo xuất viện thành công