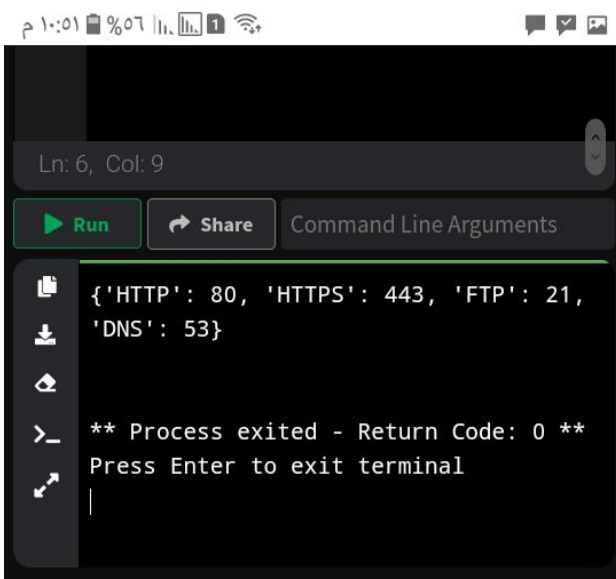


ليندا محمد إسماعيل 2686

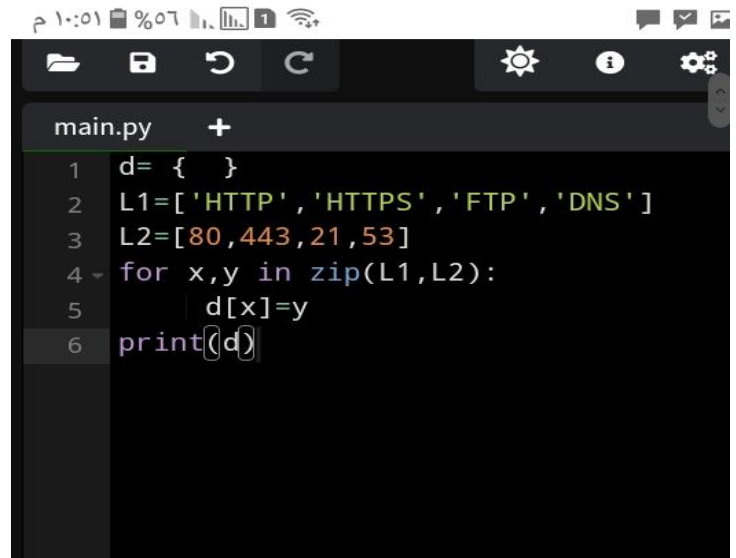
السؤال الأول:

-A-

- نعرف القائمة L1 التي تحوي أسماء البروتوكولات كمفاتيح
نعرف القائمة L2 التي تحوي ارقام المنافذ لهذه البروتوكولات كقيم
من خلال الحلقة for وباستخدام التابع Zip ندمج العناصر المتشابهة المتقابلة من القائمتين معا
ثم ننشئ القاموس d حيث تعتبر عناصر القائمة L1 مفاتيح في القاموس وعناصر القائمة L2
المقابلة قيم لهذه المفاتيح



```
Ln: 6, Col: 9
Run Share Command Line Arguments
{'HTTP': 80, 'HTTPS': 443, 'FTP': 21, 'DNS': 53}
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```



```
main.py +
1 d= { }
2 L1=['HTTP','HTTPS','FTP','DNS']
3 L2=[80,443,21,53]
4 for x,y in zip(L1,L2):
5     d[x]=y
6 print(d)
```

نقوم أخيرا بطباعة القاموس الذي يحتوي على
الزوجين معا المفاتيح والقيم حيث أن كل بروتوكول مرتبط بالمنفذ الذي يستخدمه.

-B-

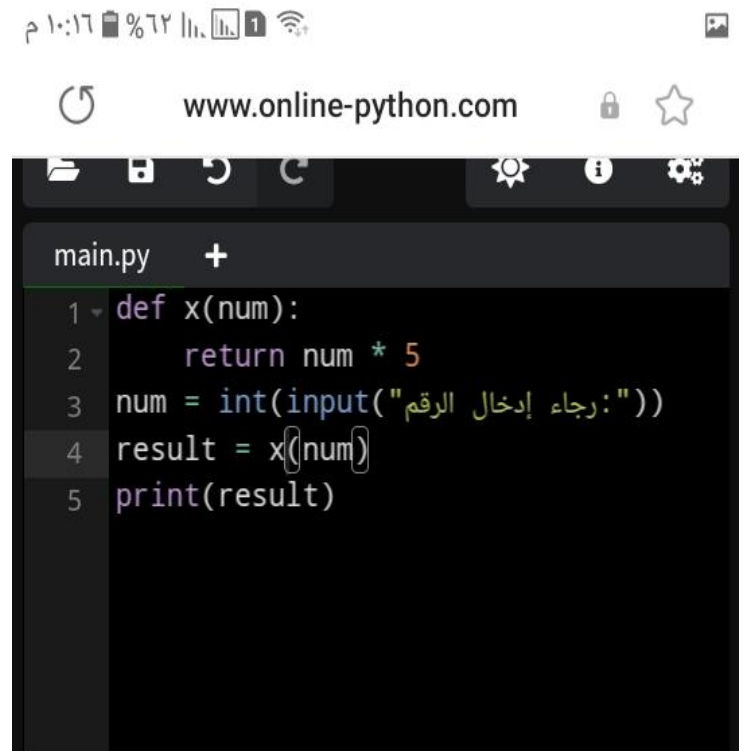
هنا أخذت الرقم كعامل وسوف نعيد ناتج ضرب الرقم الذي سيتم إدخاله كعامل في التابع
بالرقم 5

نطلب من المستخدم إدخال رقم

يتم استدعاء التابع ونمرر الرقم الذي أدخله المستخدم كعامل
ثم نطبع النتيجة .



```
Ln: 4, Col: 11
Run Share Command Line Arguments
رجاء إدخال الرقم:
9
45
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```



```
www.online-python.com
main.py +
1 def x(num):
2     return num * 5
3 num = int(input("رجاء إدخال الرقم:"))
4 result = x(num)
5 print(result)
```

-C-

نقوم بتعريف قائمة L تحوي مجموعة عناصر

ثم باستخدام الحلقة For نقوم بالتكرار على كل عنصر في القائمة وداخل كل تكرار يتم التحقق

فيما اذا كان العنصر يبدأ بحرف B باستخدام التابع startswith()

واذا كان الشرط صحيح اطبع العنصر .

١٢:٤٠ م ٢٣% ١٠٠% ١٠٠%

www.online-python.com

١٢:٤٠ م ٢٣% ١٠٠% ١٠٠%

```
Ln: 4, Col: 20
Run Share Command Line Arguments
Bio
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

```
main.py +
1 L = ['Network', 'Bio', 'Programming', 'physics', 'Music']
2 for item in L:
3     if item.startswith('B'):
4         print(item)
```

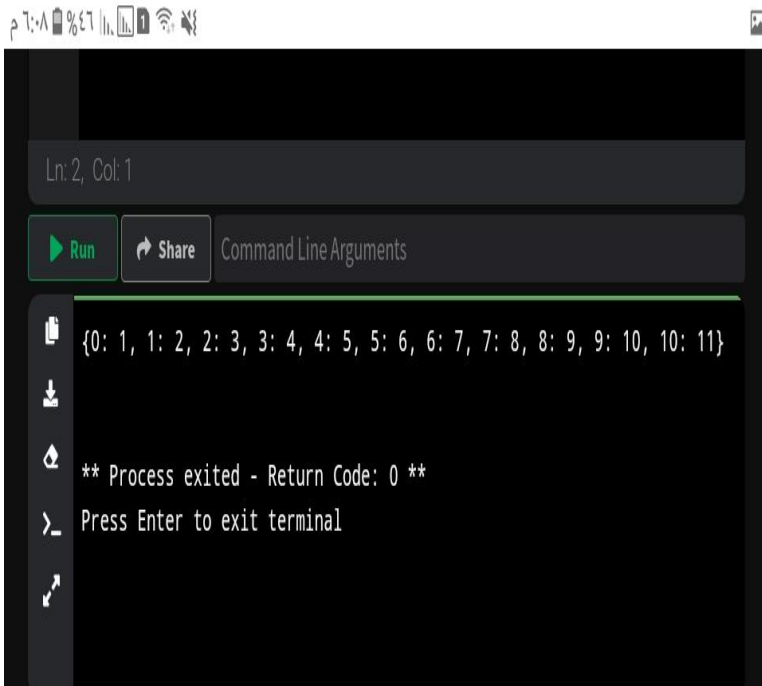
-D-

نقوم بإنشاء قاموس d

هذا القاموس يأخذ قيم من 0 إلى 10 كمفاتيح

وتكون قيمة كل مفتاح هي القيمة نفسها مضافا لها 1

ثم نقوم بطباعة القاموس.

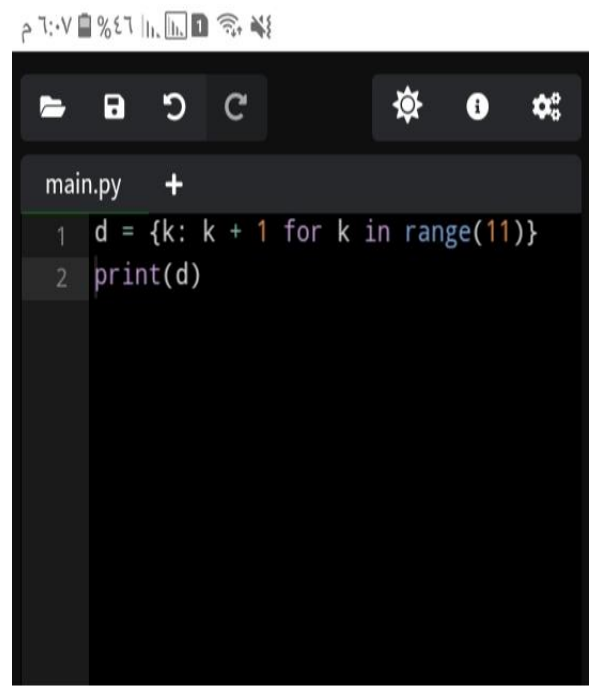


```
Ln: 2, Col: 1
```

```
{0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 10, 10: 11}
```

```
** Process exited - Return Code: 0 **
```

```
> Press Enter to exit terminal
```



```
main.py +
```

```
1 d = {k: k + 1 for k in range(11)}
```

```
2 print(d)
```

السؤال الثاني:

نطلب من المستخدم ادخال رقم ثنائي

يتم تخزين الأرقام المدخلة من المستخدم في قائمة X

نعطي قيمة بدائية تساوي 0 لتخزين القيمة العشرية النهائية Y

باستخدام حلقة For يتم تكرار العملية حسب عدد الأرقام الثنائية المدخلة من المستخدم

نستخرج الرقم الثنائي الأخير من القائمة X

إذا كان الرقم يساوي 1 يضاف للمتغير y القيمة المحسوبة باستخدام الصيغة 2 مرفوعة للقوة i

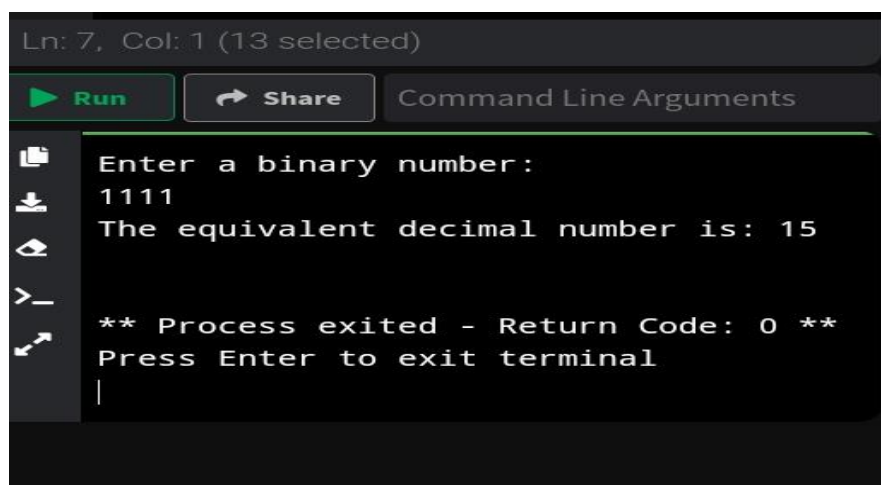
ثم نطبع الرقم العشري



main.py

+

```
1 X = list(input("Enter a binary number: "))
2 Y = 0
3 for i in range(len(X)):
4     digit = int(X.pop())
5     if digit == 1:
6         Y += pow(2, i)
7 print("The equivalent decimal number is:", Y)
```



السؤال الثالث:

نستدعي المكتبة Json للتعامل مع بيانات Json

ننشئ قاموس لتخزين الأسئلة

نعرف متغير Score ونعطيه قيمة ابتدائية تساوي 0 حيث يقوم بتتبع عدد الإجابات الصحيحة

نعرف متغير Number ونعطيه قيمة ابتدائية تساوي 1 لعرض رقم كل سؤال

يتم فتح ملف question.text الذي يحتوي على الأسئلة لتحميل الأسئلة الى

القاموس وقرائته كـ Json

نطبع رسالة ترحيب للمستخدم ونطلب منه ادخال اسمه الكامل

نعرض الأسئلة ونطلب من المستخدم الإجابة ثم نطبع رقم السؤال والسؤال نفسه

نطلب من المستخدم ادخال اجابته

ثم نختبر الإجابة التي أدخلها المستخدم حيث اذا كانت صحيحة نزيد عدد الإجابات

بمقدار 1 ونطبع صحيح والا نطبع خطأ

نزيد قيمة Number لعرض رقم السؤال التالي

ننشئ قاموس يحوي اسم المستخدم وعدد الإجابات الصحيحة

نفتح ملف score.text بوضع الكتابة ونسجل فيه نتيجة الاختبار على شكل Json

م ١٢:١٣ %٤٧

```
1 import json
2 questions = {}
3 scores = 0
4 number = 1
5 f = open("questions.txt", 'r') |
6 questions = json.load(f)
7 f.close()
8 print("اختبر برنامج بايثون")
9 print("ادخل خ للخطأ وص للصح")
```

م ١٢:١٣ %٤٧

```
10 name = input("ادخل اسمك الكامل")
11 for ques in questions.keys():
12     print("",number,":",ques)
13     ans = input("الإجابة هي ")
14     if ans.upper() == questions[ques].upper():
15         scores += 1
16         print("صحيح")
17     else:
18         print("خطأ")
```

```
19     number+=1
20 result = {name: scores}
21 m = open("score.txt", 'w')
22 json.dump(result, m)
23 m.close()
```

Ln: 20, Col: 24

السؤال الرابع:

BankAccount

يمثل حساب بنكي يحتوي على رقم الحساب واسم صاحبه والرصيد

ويحتوي على الطرق:

تهيئة البيانات الأساسية للحساب--init

إيداع مبلغ مالي للحسابDeposit

سحب مبلغ مالي من الحساب والتحقق من توفر رصيد Withdraw

استرداد الرصيد الحالي للحساب Get-balance

Savingsaccount

يمثل حساب التوفير يحتوي على نفس السمات بالإضافة لمعدل الفائدة ويحتوي على طريقة

Apply-interest لحساب وتطبيق الفائدة على الرصيد الحالي

بعد ذلك يتم انشاء حساب توفير جديد باسم

lola ورقم حساب 777777 ويتم إيداع مبلغ \$1000 في الحساب ثم نطبق الفائدة ويتم طباعة

المبلغ بعد تطبيق الفائدة

نستخدم--str--لعرض تفاصيل الحساب بشكل مخصص

...   

```
1 class BankAccount:
2     def __init__(self, account_number, account_holder, balance=0):
3         self.account_number = account_number
4         self.account_holder = account_holder
5         self.balance = balance
6     def deposit(self, amount):
7         self.balance += amount
8         return self.balance
9     def withdraw(self, amount):
```

...

```

10 ~         if amount > self.balance:
11 ~             return "Insufficient funds"
12 ~         self.balance -= amount
13 ~         return self.balance
14 ~     def get_balance(self):
15 ~         return self.balance
16 ~ class SavingsAccount(BankAccount):
17 ~     def __init__(self, account_number, account_holder, balance=0.0):
18 ~         super().__init__(account_number, account_holder, balance)
19 ~         self.interest_rate = interest_rate

```

```
19         self.interest_rate = interest_rate
20     def apply_interest(self):
21         interest_amount = self.balance * self.interest_rate
22         self.balance += interest_amount
23         return interest_amount
```

```

23         return interest_amount
24     def __str__(self):
25         return f"Account Number: {self.account_number}\nAccount H
26
27 my_savings_account = SavingsAccount("777777", "lola", interest_ra
28 my_savings_account.deposit(1000.0)
29 interest_amount = my_savings_account.apply_interest()
30 print(f"Interest applied: ${interest_amount}")
31 print(my_savings_account)
32

```

