مُلخَّصُ الوحدةِ

تعرَّفْتُ في هذه الوحدة مفهوم كلٍّ من البرمجة، والخوارزميات، والبرامج، إضافةً إلى أساسيات لغة البرمجة بايثون Python وقواعد كتابة الأوامر فيها، وكيفية توظيفها في كتابة البرامج المفيدة وتنفيذها.

في ما يأتي أبرز الجوانب التي تناولتها هذه الوحدة:

- لغة البرمجة هي مجموعة من الأوامر، تُكتَب وَفق قواعد مُحدَّدة، ويراد تحويلها إلى تعليمات يُمكِن لجهاز الحاسوب تنفيذها.
- تُصنَّف لغات البرمجة إلى نوعين رئيسين، هما: لغات عالية المستوى ولغات مُنخفِضة المستوى.
- استخدام المُبرمِجين كلًّا من المُترجِم Compiler والمُفسِّر Interpreter في تنفيذ البرامج.
- استخدام الدالَّة input في تمكين المُستخدِم من إدخال البيانات في البرنامج.
- اشتمال العناصر الرئيسة للغة البرمجة بايثون Python على كلٍّ من التعليقات Comments التي تُحسِّن مقروئية البرنامج، وتشرح أجزاء المقاطع البرمجية؛ والمُعرِّفات Identifiers التي هي أسماء مُستخدَمة للمُتغيِّرات والدوالِّ التي يجب أنْ تتبع قواعد مُعيَّنة؛ والكلمات المحجوزة Reserved Words التي لا يُمكِن استخدامها مُعرِّفاتٍ بسبب حجز اللغة لها؛ والثوابت Constants التي تظلُّ ثابتة طوال مُدَّة تنفيذ البرنامج، إضافةً إلى المُتغيِّرات التي تُعرَّف بتخصيص مساحة تخزينية في الذاكرة لقِيَم المُتغيِّرات. تشمل أنواع المُتغيِّرات في لغة البرمجة بايثون Python كلًّا من الأعداد الصحيحة int والأعداد العشرية Float والنصوص Strings والقِيَم المنطقية Booleans والقوائم lists والصفوف، والمجموعات sets والقواميس Dictionaries

- العوامل الحسابية Arithmetic Operators في لغة البرمجة بايثون Python والتي تشمل على كلٍّ من الجمع، والطرح، والضرب، والقسمة، وباقي القسمة، والقوَّة، والقسمة التحتية. أمّا العوامل المُستخدَمة في المقارنات Comparison Operators فتشمل المساواة، وعدم المساواة، وأكبر من، وأصغر من، وأكبر من أو يساوي، وأصغر من أو يساوي.
- العوامل المنطقية Logical Operators والتي تشمل على كلٍّ من AND و OR و NOT أمّا العوامل المُستخدَمة في إعطاء المُتغيِّرات قِيَمًا فتشمل الإسناد الأساسي، والإضافة والإسناد، والطرح والإسناد، والضرب والإسناد، والقسمة والإسناد، وباقي القسمة والإسناد، والقوَّة والإسناد، والقسمة التحتية والإسناد. وبينما تُحدِّد أسبقية العوامل ترتيب تنفيذ العوامل في التعابير، فإنَّ ترابط العوامل يُحدِّد اتجاه تنفيذ العوامل المُتماثِلة من حيث الأسبقية.
- تعريف الكتل البرمجية بمجموعة من الجمل ذات الصلة التي تُحدَّد باستخدام المسافات الفارغة، وهي مسافات ضرورية لتحديد الكتل البرمجية في لغة البرمجة بايثون Python استعمال الجمل الشرطية في لغة البرمجة بايثون Python لتعليق تنفيذ أوامر مُعيَّنة بناءً على شروط يُحدِّدها المُبرمِج. والجمل الشرطية الأساسية هي: if و if elif if else .أمّا الجملة الشرطية if فتُستعمَل لكتابة شرط مُعيَّن، في حين تُستعمَل الجمل الشرطية if else و if elif و if elif else لكتابة شروط مُتعدِّدة. أمّا العوامل المنطقية مثل: and و or و not فتُستعمَل لربط الشروط بعضها ببعض، في حين يُعَدُّ استخدام المسافات البادئة الصحيحة مُهِمًّا لضمان عمل المقطع البرمجي بصورة صحيحة، وبيان كيفية كتابة الجمل الشرطية المُتداخِلة، وكيف يُمكِن استخدام جملة ' pass ' عند الحاجة إلى ترك جملة شرطية فارغة؛ تجنُّبًا لحدوث الأخطاء.
- استعمال الحلقات في لغة البرمجة بايثون Python لتكرار مجموعة من الجمل مَرّات عديدة. يوجد نوعان رئيسان من الحلقات في لغة البرمجة بايثون Python هما: حلقات ' while'، وحلقات ` for ` . أمّا حلقات '

while ' فتُستعمَل لتكرار تنفيذ جملة أو أكثر طالما تحقَّق شرط مُعيَّن، وإلّا توقَّف البرنامج عن تنفيذ الجمل. وأمّا حلقات ' for ' تُستخدَم في تنفيذ المقطع البرمجي مَرّات مُحدَّدة.

- استعمال جمل التحكُّم لضبط سَيْر تنفيذ الحلقات. أمّا جملة التحكُّم ' break ' فتُستعمَل لإيقاف الحلقة عند تحقُّق شرط مُعيَّن، ثمَّ تنفيذ الجمل التي تلي الحلقة. وأمّا جملة التحكُّم ' 'continue فتُستعمَل لإيقاف الدورة الحالية في الحلقة، والانتقال إلى الدورة التالية عند تحقُّق شرط مُعيَّن. كذلك يُمكِن استعمال جملة ` else ` مع الحلقات لتنفيذ مقطع برمجي إضافي بعد انتهاء الحلقة.
- استعمال الدالَّة range لإنشاء سلسلة من الأرقام وَفقًا للمُعامِلات المُحدَّدة.
- إنشاء قوائم في لغة البرمجة بايثون Python واستخدامها في تخزين مجموعة من القِيَم المُتنوِّعة. ويُمكِن إضافة عناصر إلى القائمة أو حذفها، والوصول إلى العناصر باستخدام الفهارس. كذلك يُمكِن تقطيع القوائم للوصول إلى أجزاء منها خلال مُدَد زمنية مُحدَّدة، فضلًا عن إنشاء قوائم مُركَّبة تحتوي على قوائم أُخرى؛ ما يسمح بتمثيل البيانات ثنائية الأبعاد مثل المصفوفات.
- إنشاء سلاسل نصية والتعامل معها في لغة البرمجة بايثون Python وإمكانية الوصول إلى أجزاء من هذه السلاسل باستخدام الفهارس والتقطيع، فضلًا عن دمجها، وإجراء عمليات فيها،تشمل العمليات الأساسية في القوائم والسلاسل النصية كلًّا من الجمع والتكرار. كذلك يُمكِن استعمال المُعامِل `+` لجمع القوائم، واستعمال المُعامِل `*` لتكرار العناصر، واستعمال الدوالِّ الجاهزة لمعالجة القوائم، مثل len ، و max ، و min ، فضلًا عن استخدام حلقات التكرار والشروط في التحقُّق من القِيَم المُخزَّنة في القوائم المُركَّبة.
- استخدام الدوالُّ الجاهزة في معالجة القوائم والسلاسل النصية في لغة البرمجة بايثون Python إذ تُستخدَم الدالَّة split والدالَّة join في تقسيم السلاسل النصية وجمعها، في حين تُستخدَم الدالَّة sort والدالَّة reverse في ترتيب القوائم وعكس ترتيبها.

- تجزئة المشكلة الكبيرة إلى أجزاء صغيرة يُمكِن تحليلها وكتابتها بوصفها وحدات برمجية أو كائنات. وكذلك استيراد الوحدات البرمجية Modules في لغة البرمجة بايثون Python واستخدام الدوالِّ الجاهزة التي تُوفِّرها هذه الوحدات؛ فضاً عن تعريف الدوالِّ البرمجية الخاصة لتنفيذ وظائف مُحدَّدة يُمكِن استخدامها في البرامج وإعادة استخدامها فيه. يضاف إلى ذلك تعرُّف المقصود بنطاق المُتغيِّرات، وما يتفرَّع منها من مُتغيِّرات محلية Local Variables ومُتغيِّرات عامة Global Variables
- توثيق البرامج باستخدام سلاسل التوثيق Docstrings وكذلك استخدام التعابير الشرطية في التحقُّق من الشروط داخل الدوالِّ، وكيفية إرجاع النتائج باستخدام جملة return

أسئلةُ الوحدةِ

السؤال الأوَّل :أختار رمز الإجابة الصحيحة في كلٍّ ممّا يأتي:

1 .إحدى الآتية تُعَدُّ لغة برمجة عالية المستوى:

- لغة الآلة.
- لغة التجميع.
- لغة البرمجة بايثونPython
- اللغة الثنائية.

2 .يعمل المُترجِم **Compiler** على ترجمة:

- اللغة عالية المستوى إلى لغة الآلة دفعة واحدة.
- اللغة عالية المستوى إلى لغة الآلة سطرًا بسطر.
- لغة الآلة إلى لغة عالية المستوى.
- لغة التجميع إلى لغة عالية المستوى.

3 .ناتج print(2**3) في برمجية بايثون Python هو:

- 5
- 6
- 7
- 9

.
4البيانات التي تُستعمَل لتخزين النصوص في برمجية بايثون Pytho
n هي من نوع:

- int
- float
- string
- Bool

5 .إحدى الجمل الآتية تتسبَّب في حدوث خطأ في برمجية بايثون Python

- print("Hello , World ! ")
- print("Hello" + "World")
- " print("Hello", "World)
- print("Hello" + 2)

6 . 134 يُمكِن إنشاء قائمة في برمجية بايثون (Python)باستخدام:

- [] = list
- {} = list
- () = list
- "" = list

7 .تعمل الدالَّة len() في برمجية بايثون (Python)على:

- إيجاد عدد الأحرف في سلسلة نصية.
- إيجاد عدد العناصر في قائمة ما.
- إيجاد عدد الأحرف في سلسلة نصية، وإيجاد عدد العناصر في قائمة ما.
- لا شيء ممّا ذُكِر.

.
8الكلمة المفتاحية التي تُستعمَل لبدء الدالَّة في برمجية بايثون Pythonهي:

- fun
- def
- function
- define

9 . ناتج ((print(type(10 في برمجية بايثون Python هو:

-
  -
    -

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

.
10إحدى الآتية تُمثِّل الطريقة الصحيحة لبَدْء حلقة for في برمجية بايثون Python :

- for (i = 0; i < **10**; i++)
- for i in range(10)
- for i in 0 to 10

- loop i in range(10)

11. الوظيفة التي تؤدّيها جملة التحكُّم (break)في حلقة (for)هي:

- إنهاء التكرار الحالي، وبَدْء تكرار جديد.
- إنهاء الحلقة بصورة كاملة.
- تجاوز التكرار الحالي.
- إعادة الحلقة إلى وضع البداية.

13 .
12ناتج print("Hello"+"World") في برمجية بايثون Python هو:

- HelloWorld
- Hello World
- Hello+World
- Helloworld

13 . ناتج 5 % 2في برمجية بايثون Python هو:

- 2
- 2.5
- 1
- 0.5

السؤال الثاني :أُميِّز الجمل الصحيحة من الجمل غير الصحيحة في ما يأتي:

1. في لغة البرمجة بايثون Python ، تُستعمَل عبارة if لإنشاء حلقة. (خطأ)
2. يُمكِن للقوائم في لغة البرمجة بايثون Python تخزين عناصر تحوي أنوا عًا مختلفةً من البيانات. (صح)

3. العامل =في لغة البرمجة بايثون Python يُستخدَم في المقارنة بين قي متين. (خطأ)
4. يُستخدَم العامل =+ في إضافة قيمة إلى أحد المُتغيّرات، وإسناد النتيجة إلى ه ذا المُتغيّر.(صح)
5. تُستخدَم الكلمة المفتاحية elif في لغة البرمجة بايثون Pythonللتعامل مع شروط مُتعدِّدة.(صح)
6. يجب دائمًا أنْ تعيد الدوالُّ قيمة في لغة البرمجة بايثون Python (خطأ)
7. تُستعمَل علامات الاقتباس الفردية والمزدوجة لتعريف سلسلة نصية في لغة البرمجة با يثون Python (صح)
8. تتطلَّب حلقة for في لغة البرمجة بايثون Pythonوجود مُتغيِّر فهرسة صريح. (خطأ)
9. في لغة البرمجة بايثون Python يكون ناتج كلٍّ من 2 * 3و 2 ** 3 مُتماثِلًا. (خطأ)
10. تبدأ التعليقات في لغة البرمجة بايثون Python بالرمز// .(خطأ)

السؤال الثالث : أملأ الفراغ بما هو مناسب في الجمل الآتية:

1. ___+___ عامل يُستعمَل لجمع رقمين في لغة البرمجة بايثون Python
2. ___list___ مجموعة من العناصر، مُرتَّبة وقابلة للتغيير في لغة البرمجة بايثون Python
3. ___def___ الكلمة المفتاحية لتعريف دالَّة في لغة البرمجة بايثون Pyt hon
4. تُستعمَل الدالَّة ___print()___ لطباعة المخرجات في لغة البرمجة بايثون Python
5. تُستخدَم ___for___ في تكرار مجموعة من الجمل.

6. في لغة البرمجة بايثون Python تستمر حلقة while في التنفيذ ما دام الشرط صحيحًا.
7. في لغة البرمجة بايثون Python تُستعمَل الدالَّة int() لتحويل قي مة إلى عدد صحيح.
8. يُطلَق على الخطأ الناجم عن صياغة غير صحيحة في لغة البرمجة بايث ون Python اسم SyntaxError
9. الكلمة المفتاحية لاستيراد وحدة في لغة البرمجة بايثونPy import thon
10. يُستعمَل المُعامِل + لدمج النصوص في لغة البرمجة بايثون

السؤال الرابع: أكتب برنامجًا بلغة البرمجة بايثون Pythonيأخذ رقمًا بوص فه مدخلًا، ويطبع مُربَّعه.

```
def  square(num):
   return num**2
num = int(input("Enter a number: "))
result = square(num)
print("The square of", num, "is", result)
```

السؤال الخامس: أكتب برنامجًا بلغة البرمجة بايثون Python يستخدم حلقة for في طباعة الأرقام من 1 إلى 10

```
for  i  in range(1, 11):
  print (i)
```

السؤال السادس: أكتب برنامجًا بلغة البرمجة بايثون Python يُستخدَم في ح ساب مجموع كل الأعداد الزوجية التي تقع بين العدد 1والعدد 50

```
sum = 0
for number in range(1, 51):
  if number % 2 == 0:
    sum += number
print ("The sum of even numbers is", sum)
```

السؤال السابع : أكتب برنامجًا بلغة البرمجة بايثون Python يُدخِل قائمة م ن الأرقام، ويطبع أكبر رقم فيها

```
numbers = [ int (x) for x in  input  ("Enter a list of numbers separated by spaces: "). split()]
largest_number = max(numbers)
print("The largest number is:", largest_number)
```

السؤال الثامن : في ما يأتي مجموعة من المقاطع البرمجية المكتوبة بلغة البرم جة بايثون Python أتتبَّع الأوامر في هذه المقاطع، وأكتشف الأخطاء الموجود ة في البرنامج من دون تنفيذه، ثمَّ أقترح طرائق لتصحيحها:

البرنامج بعد التصحيح

```
x = int(input ("Enter a number:"))
if x>10 :
    print ("x is greater than 10")
else:
    print ("x is less than or equal to 10")
    first_number = int(input("Enter first number: "))
second_number = int(input("Enter second number: "))
sum = first_number + second_number
print("The sum is:", sum)
#code 8-2
i = 1
for i in range ( 1 , 11 ):
    print (i)
    i+=1
```

```
x = input("Enter a number: ")
if x > 10:
    print("x is greater than 10")
else:
    print("x is less than or equal to 10")
1st_number = int(input("Enter first number: "))
2nd_number = int(input("Enter second number: "))
sum = first_number + second_number
print("The sum is:", sum)
code 8-2
i = 1
for i <= 10:
    print(i)
    i += 1
```

الحل:

التصحيح	الخطأ
first_number	1st_**number**
second_number	2nd_number
#code 8-2	code 8-2
for i in range (1,11) :	for i <= 10:

السؤال التاسع : أُعدِّل المقطع البرمجي الآتي؛ لكي يتمكَّن البرنامج من قبول م ا يُدخِله المُستخدِم من مدخلات، ثمَّ يطبع عبارة تُبيِّن نوع العدد(فردي أو زوج ي:)

```
number = int("Enter a number: ")
if number % 2 == 0:
print("The number is even")
else:
print("The number is odd")
```

الحل : اضافة دالة الإدخال input() كما يلي :

```
number = int(input("Enter a number: "))
if number % 2 == 0:
   print("The number is even")
else:
   print("The number is odd")
```

السؤال العاشر : أكتب برنامجًا بلغة البرمجة بايثون) Python (، تُستخدَم فيه الدوالُّ Functionsلتنفيذ مجموعة من العمليات الحسابية(الجمع، الطرح، ا لضرب، القسمة)بناءً على مدخلات المستخدم .

```
def add(x, y):
    return x + y
def subtract(x, y):
  return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
  if y == 0:
    print ("error div by 0")
  else:
      return  x / y
num1 = float(input("enter first number: "))
num2 = float(input("enter second number: "))
 choice = input("enter your choice (+ or - or * or /
): ")
if choice == '+':
  print(num1, "+", num2, "=", add(num1, num2))
elif choice == '-':
  print(num1, "-", num2, "=", subtract(num1,
num2))
elif choice == '*':
  print(num1, "*", num2, "=", multiply(num1,
num2))
elif choice == '/':
  print(num1, "/", num2, "=", divide(num1, num2))
else:
  print(" wrong choice!")
```

السؤال الحادي عشر :أكتب برنامجًا بلغة البرمجة بايثون Python لاختب ار قوَّة كلمات المرور بناءً على مجموعة من المعايير، مثل :الطول، ووجود ا لحروف الكبيرة والحروف الصغيرة، والأرقام، والرموز الخاصة، علمًا بأنَّ ا لبرنامج سيطلب من المُستخدِم إدخال كلمة المرور، ثمَّ يتحقَّق من درجة قوَّتها وتعقيدها، ثمَّ يعرض نتيجة الاختبار.

```
import re
def   test(password):
  """
  """
  if len (password ) < 8 :
    return "weak password على  أحرف 8 يجب أن تحتوي على :
الأقل."

  if not re.search("[a-z]", password):
    return "  weak password يجب أن تحتوي على أحرف :
صغيرة."
  if not re.search("[A-Z]", password):
    return " weak passwordيجب أن تحتوي على أحرف كبيرة :."
  if not re.search("[0-9]", password):
    return " weak passwordيجب أن تحتوي على أرقام :."

  if not re.search("[!@#$%^&*()]", password):
```

```
    return " weak passwordيجب أن تحتوي على رموز خاصة :."
  return "strong password!"
password = input("Enter password: ")
result = test(password)
print( result)
```