الوحدة رقم 01: أساسيات لغة البرمجة "بايثون"

المحاضرة رقم 01:

1) مفهوم المتغيرات في البرمجة وفي لغة بايثون

ما هو المتغير في البرمجة؟

المتغير هو اسم **يُعطى لمكان في الذاكرة** لتخزين قيمة يمكن استعمالها لاحقًا داخل البرنامج. بمعنى بسيط: المتغير يشبه صندوقًا نضع فيه قيمة (رقم، نص، نتيجة حساب...) ونستعملها عند الحاجة.

المتغير في لغة بايثون

في بايثون لا نحتاج إلى تحديد نوع المتغير مسبقًا، فقط نكتب: income = 50000

هنا: Income اسم المتغير و 50000 هي القيمة المخزنة فيه

أمثلة:

```
price = 1200

product_name = "Laptop"

tax_rate = 0.19
```

2) قواعد تسمية المتغيرات

لكي يكون اسم المتغير صحيحا في بايثون، يجب احترام القواعد الأساسية التالية:

1. يجب أن يبدأ بحرف أو بشرطة سفلية _
2. لا يمكن أن يبدأ برقم
3. لا يحتوي على فراغات
4. لا يحتوي على رموز خاصة مثل @أو #
5. لا يكون كلمة محجوزة في بايثون مثل: if, for, while

أمثلة صحيحة:

```
total_income = 100000

price1 = 500

_student = "Ali"
```

أمثلة خاطئة:

```
1price = 500       # يبدأ برقم

total income = 5   # يحتوي على فراغ

if = 10            # كلمة محجوزة
```

3) الأنواع المختلفة للمتغيرات في بايثون

في بايثون توجد عدة أنواع أساسية:

أ. **الأعداد الصحيحة (int) :**

تستعمل لتخزين أعداد بدون فاصلة مثال: quantity = 10

ب. **الأعداد العشرية (float) :**

تستعمل للأعداد التي تحتوي على فاصلة مثال : price = 99.5

ج. **النصوص (str):**

تكتب بين علامتي تنصيص مثال : name = "Sara"

د. **القيم المنطقية(bool) :**

تأخذ فقط القيمة : True أو False مثال: is_active = True

4) تعليمة الإسناد:

هي العملية التي نضع بها قيمة داخل متغير باستخدام الرمز: " = "

إذن علامة " =" في البرمجة لا تعني مساواة رياضية، بل تعني إسناد قيمة.

مثال بسيط:

salary = 60000 ، تعني: ضع القيمة 60000 داخل المتغير salary .

يمكننا تغيير قيمة المتغير

```
salary = 60000
salary = 65000
```

القيمة الجديدة تحل محل القديمة.

تمارين تطبيقية

التمرين 1 :

أكتب برنامجا يقوم بـ:

1) إنشاء متغير يمثل سعر منتج (1500)
2) إنشاء متغير يمثل نسبة الضريبة (0.19)
3) حساب السعر النهائي
4) طباعة النتيجة

التمرين 2:

أكتب برنامجًا يقوم بـ:

1) إنشاء متغير يمثل عدد العمال في شركة (25)
2) إنشاء متغير يمثل أجر العامل الواحد (40000)
3) حساب الكتلة الأجرية الكلية
4) طباعة النتيجة

التمرين 3:

ما القيمة النهائية للمتغير x ؟

```
x = 5
x = 10
x = x + 3
print(x)
```

التمرين 4: إيجاد قيم a و b النهائية:

```
a = 2
b = 3
c = a
a = b
b = c
print(a, b)
```

المحاضرة رقم 02:

الأنواع المختلفة للعمليات (Operators) في بايثون

تُستخدم عدة أنواع من العمليات في لغة بايثون لإجراء عمليات على المتغيرات والقيم المرتبطة بها.

تصنّف بايثون أنواع العمليات إلى المجموعات التالية:

1) **العمليات الحسابية:(Arithmetic Operators)**

تُستخدم لتنفيذ العمليات الحسابية الأساسية مثل الجمع (+) والضرب(*) و الطرح (-)و القسمة(/)، باقي القسمة(%)، الأس(**)، القسمة الصحيحة(//).

مثال :

```
a = 5
b = 3

addition = a + b
soustraction = a - b
multiplication = a * b
division = a / b

print("Addition : a + b = ", addition)      # output : a + b = 8
print("Soustraction : a - b = ", soustraction) # output : a - b = 2
print("Multiplication : a * b = ", multiplication) #output : a * b = 15
print("Division : a / b = ", division)     # output : a / b = 1.66
```

2) **عمليات الإسناد (Assignment Operators)**:

الإسناد البسيط : تُستخدم لإسناد قيمة إلى متغير(=)، مثال x= 10

الإسناد المركب: بالإضافة للإسناد البسيط توجد عملية إسناد مركبة (مثل =+) تسمح بتعديل قيمة المتغير مباشرة، الأمثلة في الجدول توضح هذه العملية:

Operators	Exemple	Explication
=	x = 7	x = 7
+ =	x + = 5	x = x + 5
- =	x - = 5	x = x -5
* =	x * = 5	x = x *5
/ =	x / = 5	x = x / 5
% =	x % = 5	x = x %5
// =	x // = 5	x = x //5
** =	x ** = 5	x = x **5

الإسناد المتعدد : لغة بايثون توفر أيضا إمكانية إسناد عدة قيم لعدة متغيرات في نفس الوقت

مثال:

```
n, m = 10, 20
print("n =", n)
print("m =", m)
```

الناتج عند التنفيذ:

```
n = 10
m = 20
```

3) عمليات المقارنة (Comparison Operators):

تُستخدم لمقارنة قيمتين وتُرجع نتيجة منطقية: صحيح (True) أو خطأ(False) ، مثل: يساوي (= =)، أكبر من(>)، أصغر من (<)، أكبر من أو يساوي (=>) ، أصغر من أو يساوي(=<) ، الإختلاف(=!)

مثال:

```
a = 5
b = 8
print(a == b)     # False : a n'est pas égal à b
print(a != b)     # True  : a est différent de b
```

ذ

4) العمليات المنطقية:(Logical Operators)

تستخدم لدمج الشروط المنطقية أو نفيها، مثل: و(and) ، أو(or) ، ليس (not) . and يُرجع القيمة True فقط إذا كان التعبيران كلاهما صحيحين، or يُرجع القيمة True إذا كان على الأقل أحد التعبيرين صحيحًا، not يعكس القيمة المنطقية.

مثال1:

```
a = 5
b = 10

print(a < b and b > 0)
# Les deux conditions sont vraies, donc:
# Résultat : True
```

مثال 2 :

```
n = 7
print(not n < 5)

#La condition n < 5 est fausse, donc sa négation est vraie, donc:
#Résultat : True
```

تمرين:

- ما هي نتيجة العملية التالية: 2 * 3 + 5 ؟
- ما هي نتيجة العملية التالية: 2/ 8
- إذا كانت القيم التالية:

  a = True ،b = False ،c = True

ما نتيجة الصيغ المنطقية التالية: **a and b or c / a or b and c/ not a or b or c**

الحل بالنسبة للصيغة الأولى:a and b or c

- a and b ⟶ True and False ⟶ False
- False or c ⟶ False or True ⟶ **True**

الإجابة الصحيحة **True**

5) تعليمات الإدخال و الإخراج(Input & Output Functions):

الدالة print() : تُستخدم الدالة **print()** في لغة بايثون لعرض المحتوى (نصوص، متغيرات، وغيرها) ، تستقبل هذه الدالة قيمة واحدة أو عدة قيم، ويمكن أن تكون هذه القيم سلاسل نصية، أعدادًا، متغيرات، إلخ.

بنية الدالة:print()

```
print (valeur1, valeur2, ..., valeur n     )
```

مثال:

```
nom = "John"
age = 30
print("Hello, my name is", nom, "and I am", age, "years old.")
```

من الممكن أيضًا استخدام الصيغة **f-string** (المقدمة في بايثون نسخة 3.6) لتنسيق العرض.

مثال:

```
nom = "John"
age = 30
print(f"Hello, my name is {nom} and I am {age} years old.")
```

الدالةinput() : تُستخدم الدالة **input()** في بايثون للحصول على مدخلات من المستخدم. تقوم بطلب إدخال من المستخدم وقراءة سطر واحد. بعد قراءة البيانات، تقوم بتحويلها إلى سلسلة نصية (string) وإرجاعها.

بنية الدالةinput() :

```
Input (رسالة نصية تطلب من المستخدم إدخال البيانات)
```

.

قيمة الإرجاع لهذه الدالة

- تُرجع الدالة كل ما يدخله المستخدم إلى سلسلة نصية str

مثال على الدالة()input

```
# دع المستخدم يُدخل اسمه
saisie = input("Enter your name: ")

# عرض البيانات التي تم إدخالها
print("Your name is:", saisie)
```

تحويل أو تغيير نوع المتغير

يمكن تحويل نوع أي متغير بسهولة باستخدام الدوال**int()** ، **str()**، **float()**

الدالة **Type()** تعطينا نوع المتغير.

مثال 1:

```
n = 10

x = float(n)
print("نوع n هو:", type(n))
print("نوع x هو:", type(x))
```

الناتج عند التنفيذ:

```
نوع n هو: <class 'int'>
نوع x هو: <class 'float'>
```

مثال 2:

```
x = 10.0
n = int(x)
print("نوع x هو:", type(x))
print("نوع n هو:", type(n))
```

الناتج عند التنفيذ:

```
نوع x هو: <class 'float'>
نوع n هو: <class 'int'>
```

تمرين: التحقق من أهلية القرض

اكتب برنامجًا يطلب من المستخدم إدخال الراتب الشهري والعمر وعدد سنوات العمل، ثم يتحقق مما إذا كان الشخص مؤهلاً للحصول على قرض وفق الشروط: الراتب ≥ 3000، العمر ≥ 21، سنوات العمل ≥ 2.

الحل:

```
# إدخال البيانات
salary = float(input("Enter monthly salary: "))
age = int(input("Enter age: "))
years_work = int(input("Enter years of work: "))

# التحقق من شروط القرض
condition = salary >= 3000 and age >= 21 and years_work >= 2

# عرض النتيجة
print("The person is eligible for a loan:", condition)
```