MTA 98-381 LESSON 2 VARIABLES

VARIABLES

- A variable in a programming language allows us to store data temporarily in computer memory for processing.
- A variable takes up storage in computer memory.
- Everything in Python including variables and literals is object.
- We can refer to that storage location by variable name.

```
unit_price = 1.5
total_unit = 3
total_price = unit_price * total_unit
```

VARIABLE NAME HAS RULES

- Must start with a letter or the underscore character (_) and can contain letters, numerals (digits), or the underscore.
- Cannot begin with number.
- Must not have space.
- Is case-sensitive: `id` and `Id` are considered 2 different variables.
- Sample valid names: `id1, _id1, student_id`
- Sample invalid names: `1id, student id`

USE MEANINGFUL NAME

- Always use meaningful name to easily identify its purpose.
- Avoid making people guess.
- Good variable names:

```
unit_price = 1.5
total_unit = 3
total_price = unit_price * total_unit
```

- Poor variable names:

```
price1 = 1.5
total = 3
price2 = price1 * total
```

EXERCISE: VARIABLES

- 1. Create file variables.py.
- 2. Type the following code:

```
unit_price = 1.5
total_unit = 3
total_price = unit_price * total_unit
print (unit_price, "x", total_unit, "=", total_price)
```

- 3. Run the program. Output:

```
1.5 x 3 = 4.5
```

EXERCISE: VARIABLES (CONT.)

- 4. Add the following code:

```
d = "2"
e = '3'
f = d + e
print (d, '+', e, '=', f)
```

- 5. Run the program. Output:

```
1.5 x 3 = 4.5
2 + 3 = 23
```

DATA TYPES IN PYTHON

- A simple variable can be of one of the following basic data types:
  - `int` – integer, e.g. `2`.
  - `float` – floating number or real number, e.g. `0.12`.
  - `str` – string/text, e.g. `'2'` or `"2"`.
  - `bool` – Boolean, possible value is `True` or `False` only.
- You can use function `type()` to find out the type of a variable.

```
unit_price = 1.5
print (type(unit_price))
```

EXERCISE: DATA TYPES IN PYTHON

- 1. Add the following code:

```
smaller = unit_price < total_price
print (type(total_unit))
print (type(total_price))
print (type(f))
print (type(smaller))
```

- 2. Run the program. Output:

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

GETTING USER INPUT USING INPUT() FUNCTION

- We often read user input in our program in order to make the program more flexible.
- `input()` function reads user input as `string`.

```
your_name = input("Enter your name: ")
# your name is a string.
```

CONVERTING/CASTING STRING TO NUMBER

- To convert/cast a `string` to `int` or `float`, use `int()` or `float()` function.

```
unit_price = float(input("Enter unit price: "))
total_unit = int(input("Enter total unit: "))
```

EXERCISE: INPUT FUNCTION

- 1. Create a file named user_input.py.
- 2. Convert the previous total price calculation by getting user input for `unit_price` and `total_unit`.
- 3. Ask the user for his/her name.
- 4. Sample output should be:

```
Enter your name: Ali
Enter unit price: RM1.5
Enter total unit: 3

Hello Ali!
Total price = RM1.50 x 3 = RM4.50
```