LGT3109
Introduction to Coding for Business with Python
(week 2)

Xiaoyu Wang

Dept. of LMS, The HK PolyU

Summary of Week 1

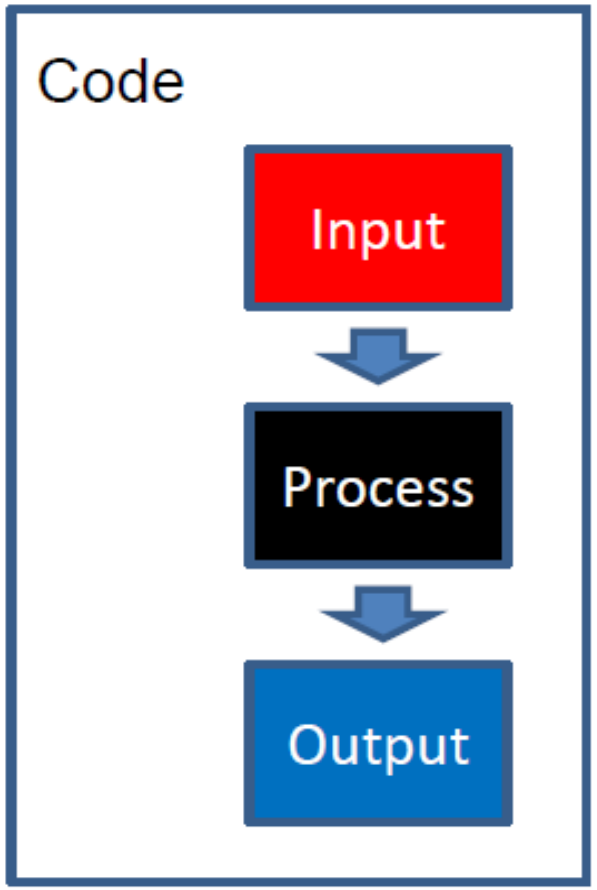
- Code is a sequence of instructions for computers to work as the tool.
- Python is a coding language for expression of the instructions

- Python code structure:

➢Input

➢Process

➢Output



Summary of Week 1

- Python code component:
  - ***Value***, ***Variable***
  - ***Statement***
  - ***Script***

- Python code step:
  - ***Sequential steps***
  - ***Repeated steps***
  - ***Conditional steps***

Values, Variables, Statements, Expressions

- Value and Variables
- Statement
- Expressions
- Comments

Motivation Case

- Foxconn assembles and exports PC worldwide.
- Foxconn plans to ship containers from Hong Kong to San Francisco.
- Foxconn asks carrier to quote a shipping rate (USD per container)
- Foxconn wants to know the total shipping cost of its cargo

Input:

```
What is the shipping quantity? 50
What is the shipping rate? 5000
```

Output:

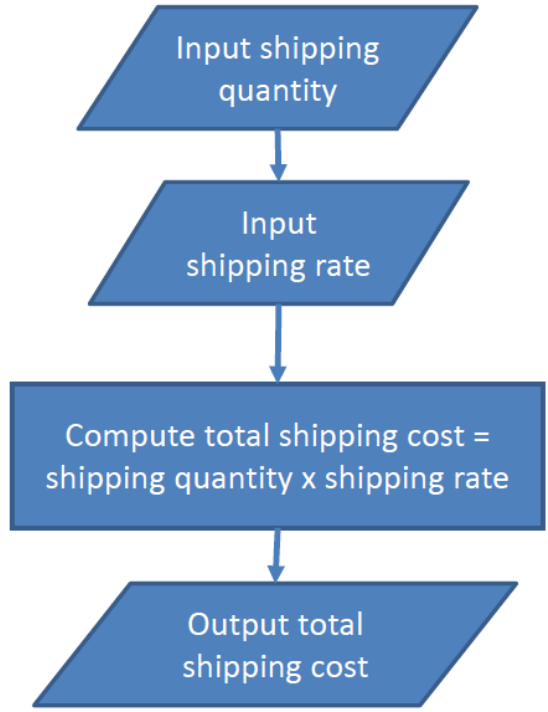
```
The total cost is 250,000
```

Motivation Case-Code

- Foxconn assembles and exports PC worldwide.
- Foxconn plans to ship containers from Hong Kong to San Francisco.
- Foxconn asks carrier to quote a shipping rate (USD per container)
- Foxconn wants to know the total shipping cost of its cargo

```
#input
quantity= int(input('What is the shipping quantity? '))
shipping_rate= float(input('What is the shipping rate? '))
#process
total_cost= shipping_rate*quantity
#output
print(f'The total cost is {str(total_cost)}')
```

Variable Operator Constant Function Comment



Values, Variables, Statements, Expressions

- Value and Variables
- Statement
- Expressions
- Comments

Value and Variables-Value

- Fixed values such as numbers, letters, etc., are called “constants” because their value does not change.
- Value/constant only 3 types: integer, float, string
- Python world: number+string, where number consists of int and float.

- An integer (int) is a number without a decimal point.
- A float is a floating-point number that has a decimal place.
- A string is a sequence of characters, can be letters, numbers, etc.

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```

Value and Variables-Value

- A quick summary of Value:
- Number
  - Integer
  - Float

- String

Value and Variables-Type

- Use “type” to know the type.

➢Float, integer, or string.

```
>>> type(123)
<class 'int'>
>>> type(98.6)
<class 'float'>
>>> type('Hello world')
<class 'str'>
>>>
```

- Integers cannot have leading zeros.
- String constants use single quotes (') or double quotes (")

```
>>> type(01)
SyntaxError: leading zeros i
 0o prefix for octal integer
>>> type(1)
<class 'int'>
>>> type(01.0)
<class 'float'>
```

```
>>> type('python')
<class 'str'>
>>> type("python")
<class 'str'>
>>>
```

Value and Variables-Number

- Numbers have two main types
  - ➢ Integers are whole numbers: 14, 2, 0, 1, 100, 401233
  - ➢ Floating Point Numbers have decimal parts: 2.5 , 0.0, 98.6, 14.0
- Other number types are variations on float and Integer: complex, fraction, etc.

```
>>> xx = 1
>>> type(xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class'float'>
>>> type(1/2)
<class'float'>
```

Value and Variables-String

- String can be letters, symbols, “numbers”, combination of the above.

- When you input data, string will be “” or ‘’

➢No exception!

```
>>> type('Hello world')
<class 'str'>
>>> type('Hello, world')
<class 'str'>
>>> type('123')
<class 'str'>
>>> type(‘Hello 123 world')
<class ‘str'>
```

Value and Variables-Type Conversion

- Use float() to convert int or str.
- Use int() to convert float or str.
- Use str() to convert float or int.

```
>>> type(float(123))
<class 'float'>
>>> type(float('123'))
<class 'float'>
>>> type(int(123.0))
<class 'int'>
>>> type(int('123'))
<class 'int'>
>>> type(str(123.0))
<class 'str'>
>>> type(str(123))
<class 'str'>
```

- Conversion should make sense!

➢float('Hello 123 world')

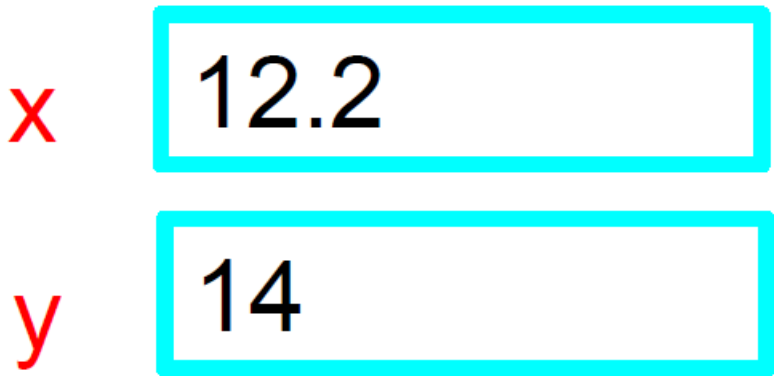
➢Int('Hello 123 world')

➢Int(123.4)

Value and Variables-Variables

- A variable is a named place in the memory.
- Store/retrieve data using the variable “name” (e.g., x, y).
- You choose the names of the variables.
- You can change the contents of a variable in a later statement.

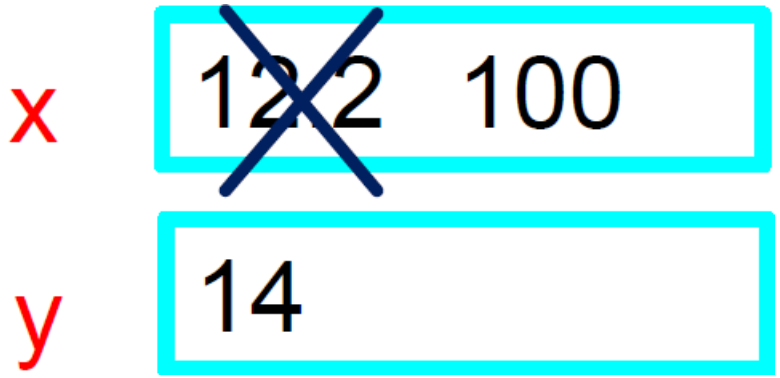
```
x = 12.2
y = 14
```



Value and Variables-Variables

- A variable is a named place in the memory.
- Store/retrieve data using the variable “name” (e.g., x, y).
- You choose the names of the variables.
- You can change the contents of a variable in a later statement.

```
x = 12.2
y = 14
x = 100
```



Value and Variables-Reserved Words

- You cannot use reserved words as variable names!

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

- Otherwise, syntax errors

```
>>> false=3
>>> false
3
>>> False=3
SyntaxError: cannot assign to False
>>>
```

Value and Variables-Name Rules

- Cannot be reserved keywords.
- Must start with a letter or underscore "_".
- Must consist of letters, numbers, and underscores.
- Case Sensitive.

```
spam     spam23   _     _eggs
23spam   #sign    var.12
spam     Spam     SPAM
```

Whether a variable name is legal can be verified in Python IDLE

Value and Variables-Name Rules

- What's wrong with these names?

```
>>> 76trombones = 'big parade'
  File "<stdin>", line 1
    76trombones = 'big parade'
               ^
SyntaxError: invalid syntax
>>> more@ = 1000000
  File "<stdin>", line 1
    more@ = 1000000
          ^
SyntaxError: invalid syntax
>>> class = 'Advanced Theoretical Zymurgy'
  File "<stdin>", line 1
    class = 'Advanced Theoretical Zymurgy'
          ^
SyntaxError: invalid syntax
>>>
```

Value and Variables-Name Rules

- Choose convenient names. (a12b3c45 vs a vs hour)
- Variables names help us remember what we store.
- However, well named variables sometimes “sound” so good that they must be reserved keywords and cannot be used for variable names.

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

What are these bits of code doing?

Value and Variables-Name Rules

- Choose convenient names. (a12b3c45 vs a vs hour)
- Variables names help us remember what we store.
- However, well named variables sometimes “sound” so good that they must be reserved keywords and cannot be used for variable names.

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

What are these bits of code doing?

Value and Variables-Name Rules

- Choose convenient names. (a12b3c45 vs a vs hour)
- Variables names help us remember what we store.
- However, well named variables sometimes “sound” so good that they must be reserved keywords and cannot be used for variable names.

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

What are these bits of code doing?

```
quantity = 35.0
shipping_rate = 12.50
total_cost = quantity * shipping_rate
print(total_cost)
```

Value and Variables-Name Rules

- Choose convenient names. (a12b3c45 vs a vs hour)
- Variables names help us remember what we store.
- However, well named variables sometimes “sound” so good that they must be reserved keywords and cannot be used for variable names.

➢For example: yield

Value and Variables-Name Rules

- The underscore character “_” can appear in a name.

- It is often used in name with multiple words

➢such as my_name , hour_rate , student_id , etc.

- The official Python Style Guide says that variables should be “lowercase with words separated by underscores as necessary to improve readability”

Value and Variables-Rethink

- What are the types for variables?
- Can you convert a variable type?

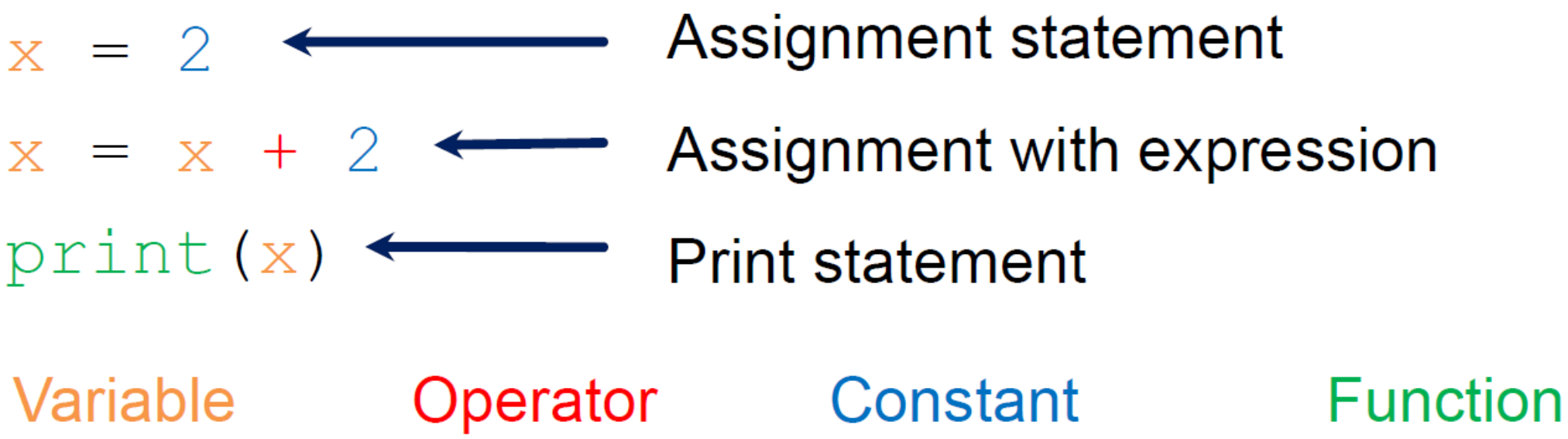
- Structure:
- Value: Number (integer, float), string
- Word: lower/higher case
- Variable: use “reasonable” variable name.

Values, Variables, Statements, Expressions

- Value and Variables
- Statement
- Expressions
- Comments

Statement

- A statement is a unit of code (or instruction) that Python interpreter can execute.
- A script contains a sequence of statements.



Statement-Print Statement

- To display a value or the value of a variable, you can use a print statement

➢print (value, value, value, …)

➢By default, values are separated by space ‘ ’.

```
>>> print('3','4')
3 4
>>> print('3 '+'4')
3 4
>>> y=4
>>> print(f'3 {str(y)}')
3 4
>>>
```

Statement-Input Statement

- Use input () function to instruct Python to pause and read data from the user.

- input() function returns a string.

- Pass a string to input() to be displayed before pausing for input.

```
nam = input('Who are you? ')
print('Welcome', nam)
```

```
Who are you? Chuck
Welcome Chuck
```

Statement-Assignment Statement

- The assignment statement (=) assign values to variables.

- An assignment statement consists of:
  - an expression on the right-hand side.
  - a variable to store the result on the left-hand side.

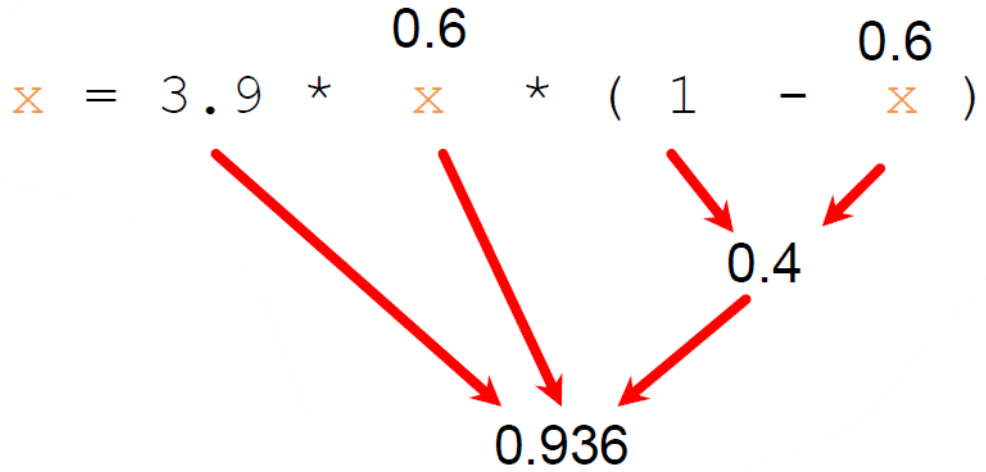
Variable = Expression

```
x = 3.9 * x * ( 1 - x )
```

Statement-Assignment Statement

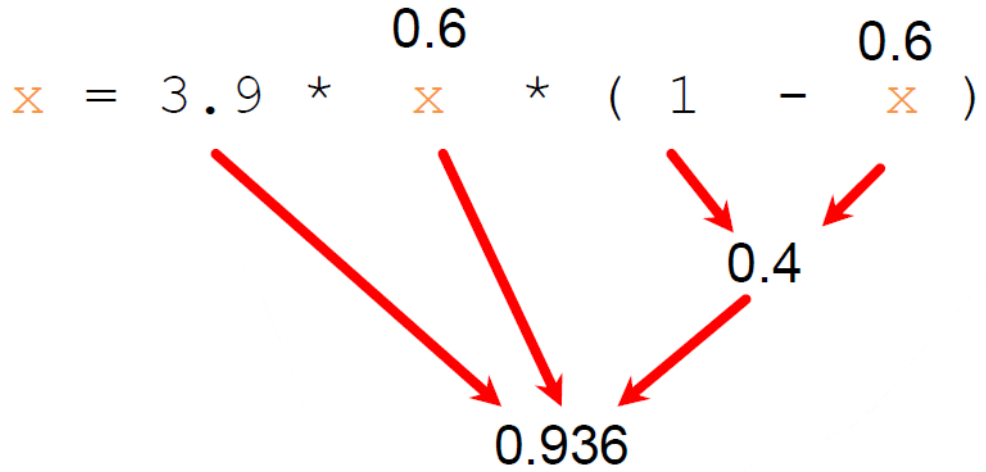
- A variable (e.g., x) is a memory location used to store a value (e.g., 0.6).

- The right side of "=" is an expression. Once the expression is evaluated, the result is placed in (assigned to) the variable on the left side of "=" (e.g., x).



Statement-Assignment Statement

- The value stored in a variable can be updated by replacing the old value (e.g., 0.6) with a new value (e.g., 0.936).



Statement-Exercise

What is the value of y?

```
x=100
y=x
x=200
```

Statement-Exercise

What is the value of y?

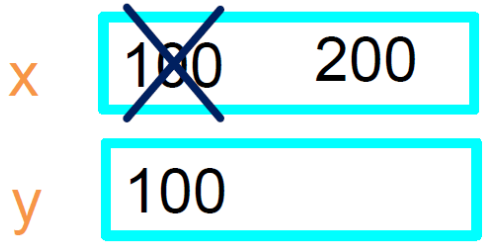
```
x=100
y=x
x=200
```

x	100
y	100

Statement-Exercise

What is the value of y?

```
x=100
y=x
x=200
```



Statement-Exercise

Are values of x and y swapped?

```
x=100
y=200
x=y
y=x
```

Statement-Exercise

Are values of x and y swapped?

```
x=100
y=200
x=y
y=x
```

x	100
y	200

Statement-Exercise

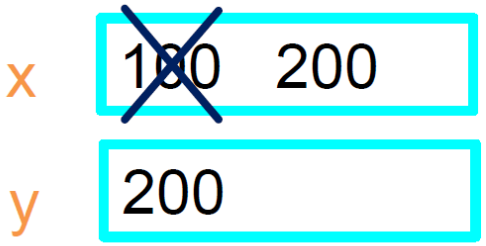
Are values of x and y swapped?

```
x=100
y=200
x=y
y=x
```

Statement-Exercise

Are values of x and y swapped?

```
x=100
y=200
x=y
y=x
```



Statement-Exercise

How to swap values of x and y?

```
x=100
y=200
temp=x
x=y
y=temp
```

Statement-Exercise

How to swap values of x and y?

```
x=100
y=200
temp=x
x=y
y=temp
```

x	100
y	200
temp	100

Statement-Exercise

How to swap values of x and y?

```
x=100
y=200
temp=x
x=y
y=temp
```

Statement-Exercise

How to swap values of x and y?

```
x=100
y=200
temp=x
x=y
y=temp
```

Values, Variables, Statements, Expressions

- Value and Variables
- Statement
- Expressions
- Comments

Expressions-Numerical Expressions

- Operators: special symbols representing computations (+,-,…)

➢** is exponentiation: 3**2 = 9

➢% is remainder: 25%7=4

➢// is integer division: 25//7=3

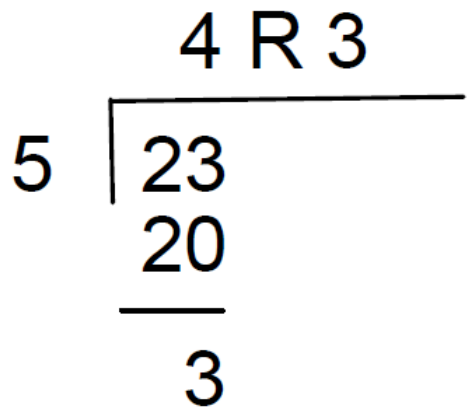
Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
//	Integer division
%	Remainder

- An expression is a combination of values, variables, and operators (e.g., 3*(4+hour)+5).

Expressions-Numerical Expressions Examples

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
>>> print(4 ** 3)
64
```

```
>>> jj = 23
>>> quotient = jj // 5
>>> remainder = jj % 5
>>> print(quotient,
remainder)
4 3
```



Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
//	Integer division
%	Remainder

Expressions-Numerical Expressions Examples

- Would expression operations change type?
- Division of two integers produces a floating-point result.
- How about multiplication, addition, subtraction…

➢Int and float. (talk about string later)

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 /
100.0)
0.99
```

Expressions-Operator Precedence

- Multiple operators in one expression, Python must know which one to do first.

- This is called “operator precedence”.

- Which operator “takes precedence” over the others?

```
x = 1 + 2 * (3 - 4) / 5 ** 6
```

Expressions-Operator Precedence

- Highest precedence rule to lowest precedence rule:

➢Parentheses are always respected

➢Exponentiation (raise to a power)

➢Multiplication, Division, and Remainder

➢Addition and Subtraction

➢Left to right

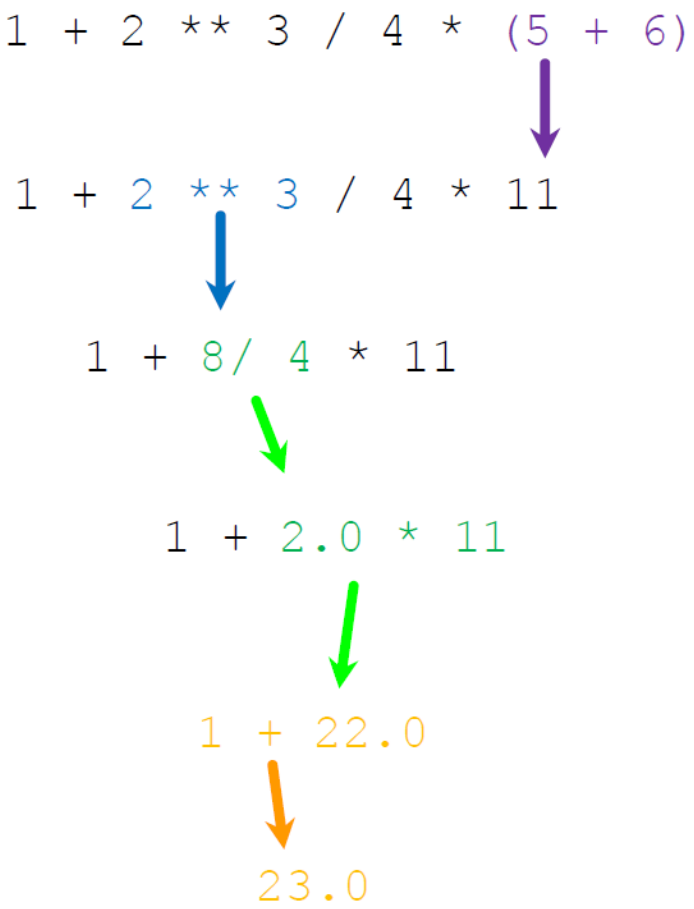
Parenthesis
Power
Multiplication
Addition
Left to Right

LAMPP

Expressions-Operator Precedence

```
>>> x = 1 + 2 ** 3 / 4 * (5 + 6)
>>> print(x)
11.0
>>>
```

Parenthesis
Power
Multiplication
Addition
Left to Right



Expressions-Operator Precedence

- Remember the rules top to bottom.
- When writing code use parentheses.

➢1 + ((2 ** 3) / 4) * (5 + 6)

- When writing code keep mathematical expressions simple enough that they are easy to understand.
- Break long series of mathematical operations up and make them clearer.

```
volume = 1*2*3 #width*hegith*length
density = 2 * 997 #twice of the water
weight  = volume * density
```

Parenthesis
Power
Multiplication
Addition
Left to Right

LAMPP

Expressions-Different Types

- Remind: operators works for float and int.
- How about numbers and strings?
- How about strings and strings?

Expressions-Different Types

- Numbers and strings: never good friends! Unless strings can be converted to numbers. Try add 1 to a string.
- Strings and strings: only “+”, unless all converted to numbers.
- Addition is different for types.

```
>>> ddd = 1 + 4
>>> print(ddd)
5



>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

Values, Variables, Statements, Expressions

- Value and Variables
- Statement
- Expressions
- Comments

Comments-Using # Mark

- Anything after # is ignored by Python.

➢Can be put on a line by itself, or at the end of a line.

- Why comment?

➢Describe what is going to happen in a sequence of code.

- For debugging: Temporarily turn off a line of code.
- Useful in debugging a code, e.g., test which one of the two possible expressions is correct.

```
# compute the percentage of the hour that has elapsed
percentage = (minute * 100) / 60
```

```
percentage = (minute * 100) / 60      # percentage of an hour
```

Comments-Using # Mark

```
# Get the name of the file and open it
name = input('Enter file:')
handle = open(name, 'r')

# Count word frequency
counts = dict()
for line in handle:
    words = line.split() # Split a line into words
    for word in words:
        counts[word] = counts.get(word) + 1

# Find the most common word
bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# All done
print(bigword, bigcount)
```

Comments-Using # Mark

```
# Get the name of the file and open it
name = input('Enter file:')
handle = open(name, 'r')

# Count word frequency
counts = dict()
for line in handle:
    words = line.split() # Split a line into words
    for word in words:
        #counts[word] = counts.get(word) + 1    for debugging
        counts[word] = counts.get(word,0) + 1

# Find the most common word
bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# All done
print(bigword, bigcount)
```

Comments-Debugging

- Three types of errors:

➢Syntax error

➢Runtime error

➢Semantic error

Comments-Debugging

- Syntax error: violating python “grammar”

➢At this point: an illegal variable name, invalid expression of values.

```
>>> bad name = 5
SyntaxError: invalid syntax
```

```
>>> month = 09
  File "<stdin>", line 1
    month = 09
              ^
SyntaxError: invalid token
```

Comments-Debugging

- Runtime error: errors when the code is executed.
- At this point: "use before defined", i.e., trying to use a variable before you have assigned a value.
- E.g., spell a variable name wrong or not notice that Variables names are case sensitive.

```
>>> principal = 327.68
>>> interest = principle * rate
NameError: name 'principle' is not defined
```

```
>>> hour = 5
>>> print(Hour)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Hour' is not defined
```

Comments-Debugging

- Semantic error: errors that cause the code not to do what you intended for it to do.
- –At this point, it could be the wrong order of operations.
- –E.g., to print the result of distance / (2 * speed), we may have a wrong code: print(distance/2*speed).

Summary

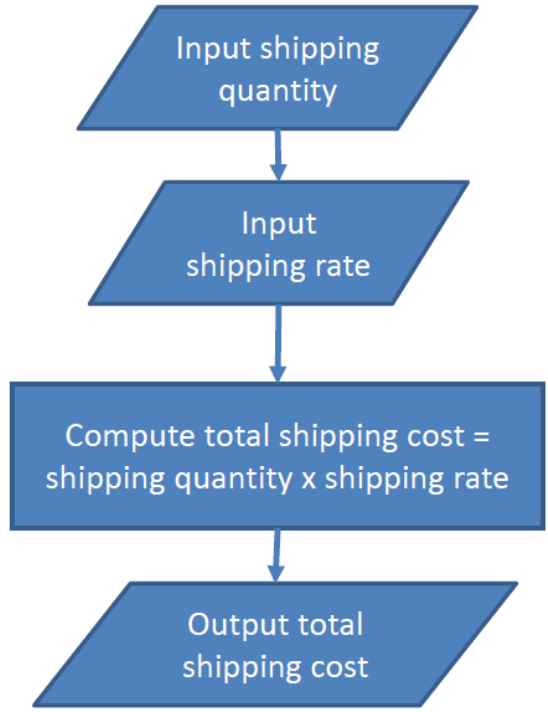
- Values cannot be changed
  - E.g. 1, 2, ‘hello’
- Variables store values for retrieval
  - E.g. name=‘Alice’
- Mnemonic and legal variable names (style)
- A statement is a unit of instruction
- A print statement displays data
  - E.g. print(x), print(‘hello world’)
- An assignment statement stores a value in a variable
  - x = 1
- An expression is a combination of values, variables, and operators
  - E.g. 1+2*x/4+(5-6)*y
- Values have different types, which are different in operations and can be converted
  - E.g., 1+2, ‘1’+’2’, int(‘1’)
- Input statement returns a string, which can be converted to numbers
  - E.g., int(input(‘what’s your age?’))
- Object oriented thinking about type as class
- Comments (anything after #) are ignored in execution, but they are useful for coding
- Possible Errors
  - Syntax error: e.g. illegal variable names
  - Runtime error: e.g. wrong spelling of variable names
  - Semantic error: e.g. wrong evaluation order in expression

Summary

- Foxconn assembles and exports PC worldwide.
- Foxconn plans to ship containers from Hong Kong to San Francisco.
- Foxconn asks carrier to quote a shipping rate (USD per container)
- Foxconn wants to know the total shipping cost of its cargo

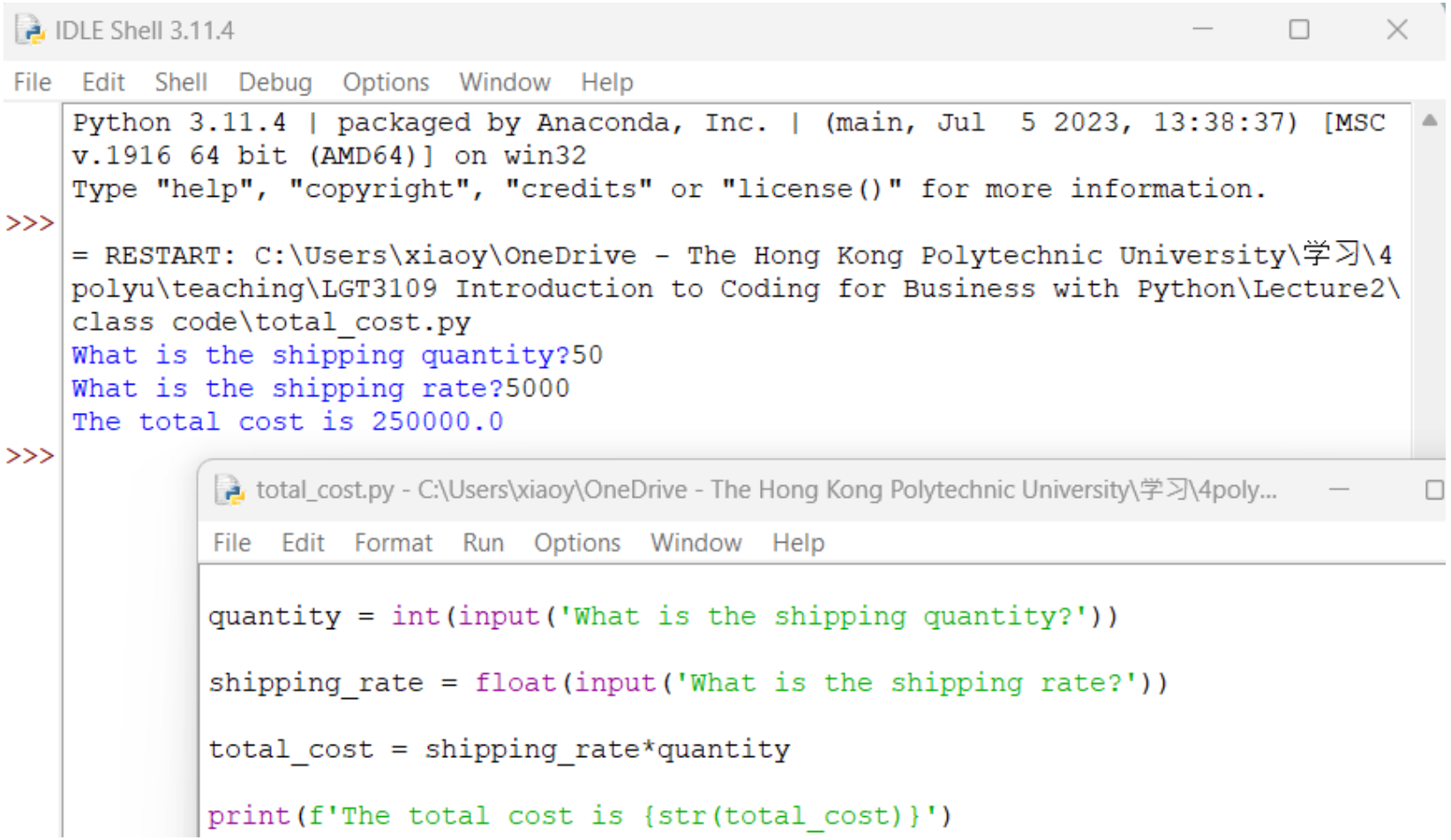
```
#input
quantity= int(input('What is the shipping quantity? '))
shipping_rate= float(input('What is the shipping rate? '))
#process
total_cost= shipping_rate*quantity
#output
print(f'The total cost is {str(total_cost)}')
```

Variable Operator Constant Function Comment



Motivation Case-Execution

- Use shell or editor?



Acknowledgement

- Acknowledgements / Contributions

- Initial Development: Charles Severance, University of Michigan School of Information
- Further Development: Zhou Xu, Hong Kong Polytechnic University
- Continuous development: Xiaoyu Wang, Hong Kong Polytechnic University