Introduction to Programming: Python
Neshat Beheshti
University of Texas Arlington

The Material for this Notebook is Adopted from
Starting Out with Python (4th Edition)- by Tony Gaddis Pearson

Introduction to Python

1. Introduction

Why python:

- Open source program. You can easily download and install the software and start using it.
- You can use it to build any piece of software.
- High-level language and easy to learn.
- Can easily be used for data science tasks.

There are two differrent versions of Python: Python 2 and Python 3. There is no support for Python 2 anymore. In this course, we use Python 3.

How we can start writing python code?

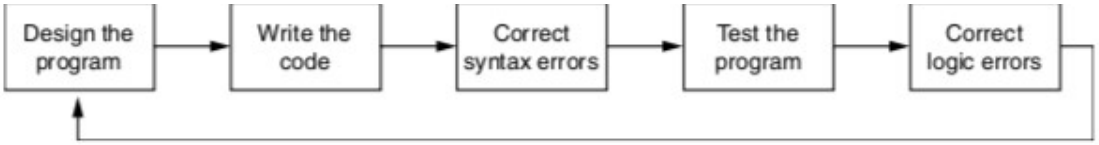
- You can use python console and interactivly write your code.
- You can write your code easily in a text file with **'.py'** extention and later execute your code.
- You can use Jupyter platform to execute your code segment by segment.

Note: In this course, we mostly use Jupyter platform for our coding.

Note: Let's an Example:

In [ ]:

Program Development Cycle



Source: Starting Out with Python (4th Edition)- by Tony Gaddis Pearson

Note: You need to understand the above stepts and know that programming is an iterative process.

How we should start coding as a beginner?

As a new programmer, you need to practice the coding logic to be able to identify various tasks for your program. Then you need to practice, practice and practice coding to learn the overall structure of programming and syntax structure of a particular language that is python in this course. There are several tools that can help you in this process:

The Process of Designing a program:

1. Understand the task that program is to perform.
2. Determine the steps that must be taken to perform the task

Example:

Write a program to calculate and display the gross pay for an hourly paid employee:

1. Get the number of hours worked
2. Get the hourly pay rate
3. Multiply the number of hours worked by the hourly pay rate
4. Display the result of the calculation

1.1. Pseudocode

Informal language that has no syntax rules and is not meant to be compiled or executed. This can be used to describe various steps in your program and later needs to be translated into a programming language.

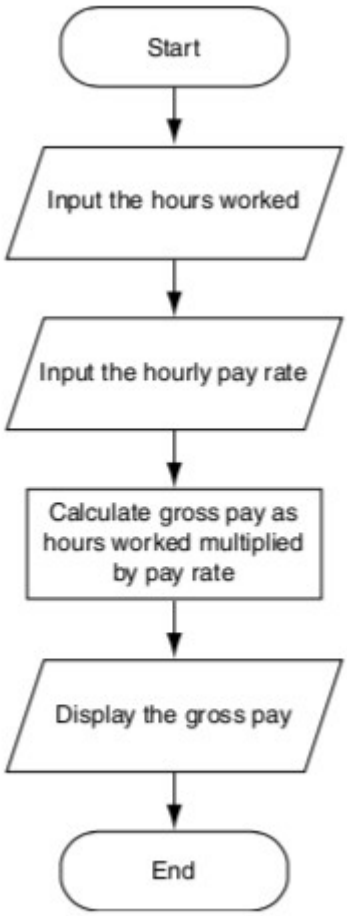
Example:
Input the hours worked
Input the hours pay rate
Calculate gross pay as hours worked multiplied by pay rate
Display the gross pay

Example:

input name of the user
Print 'Hi' + the name of the user
input the birthdate of user
Compute age of the user
Print the age of the user

1.2. Flowchart

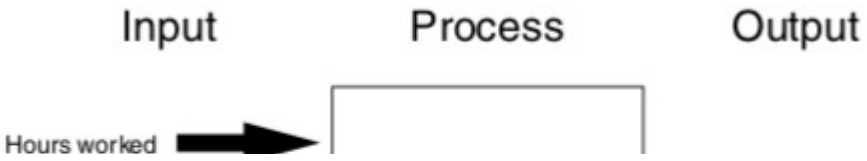
Flowchart is a useful tool to show the sequence of tasks in programming.

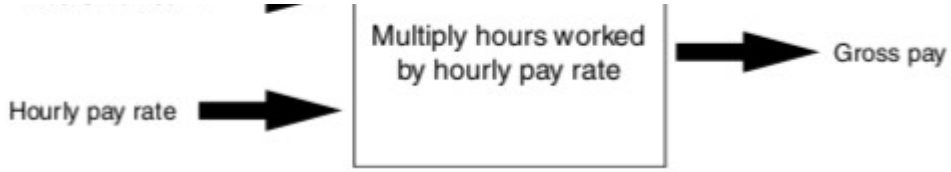


Source:Starting Out with Python (4th Edition)- by Tony Gaddis Pearson

2. Programming

2.1. Input, Processing, and Output





Source:Starting Out with Python (4th Edition)- by Tony Gaddis Pearson

Note: Before you start programming, you need to understand one basic element that is Function. Function is a pre-written piece of code that perfoms an operation.

Note: We use **print( )** function to dispaly output in a Python Program.

Example:

In [ ]:

Note: In python, we use quote marks to show string in python. You need to make sure to put your string between " ", or ' '.

Example:

In [ ]:

Example:

In [ ]:
```
print('Here is my first line')
```

In [ ]:

Example: Now, it's your turn. Try to print some arbitrary text that you want.

In [ ]:

Note: You can use triple quotes to surround multiple line of string or when you have ' or " in your string.

Example:

In [ ]:

In [ ]:
```
print("""Line1
Line2
Line3""")
```

Example: Now, it's your turn. Try to print some arbitrary text using triple quotes.

In [ ]:

2.2. Comments

Comments are short notes that explain lines or sections of a program. They are intended for people who may be reading the source code.
In Python, we begin a comment with the # character.

Examples:

In [ ]:
```
# This program displays a person's name and phone number.
print('John Austen')
```

In [ ]:
```
print('John Austen')   # Display the name.
```

2.3. Variables

A variable is a name that represents a value in the memory of your computer.

we use an assignment statement to create a variable and make it reference a piece of data.

- An assignment statement is written as fallows:

  variable = *expression*

  variable is the name of a variable and *expression* is a value, or any piece of code that results in a value.

Example:

In [ ]:
```
age = 32
weight = 120
```

Note: When you assign a value to a variable, you create a memory space to keep a value during the time you run your program. You can see these values by printing them out.

In [ ]:

In [ ]:

In [ ]:
```
print(first_name)
```

Example:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```
x = 7
print(x)
y = x * 5
```

Note: When we use a variable in the print function, we do not enclose the variable name in quote marks.

In [ ]:

In [ ]:

2.3.1 Variable Naming Rules and Styles

When you want to define a variable, you need to consider the following tips:

- The first character must be one of the letters a through z, A through Z, or an under-score character (_).
  - Variable names **CANNOT** begin with numbers.
- After the first character we may use the letters a through z or A through Z, the digits 0 through 9, or underscores.
- Uppercase and lowercase characters are distinct.
- A variable name **CANNOT** contain spaces.
  - There are different styles for using multiple words as a variable name:

    1. Using the underscore character to represent a space. **Examples:** gross_pay, pay_rate
    2. Using the *camleCase* naming style:
       - The variable name begins with lowercase letters.
       - The first character of the second and subsequent words is written in uppercase. **Examples:** grossPay, payRate
- **Important:** We should always choose names for your variables that give an indication of what they are used for.
- We cannot use one of *Python's key words* as a variable name.

In [ ]:
```
Temperature = 120
print(Temperature)
```

In [ ]:

In [ ]:

Python's key words:

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

Source:Starting Out with Python (4th Edition)- by Tony Gaddis Pearson

Note: Python allows one to display multiple items with one cell to the **print** function

Example:

In [ ]:
```
room = 245
```

Example:

In [ ]:
```
#Using two print functions to dispaly two items.
print('my age is')
print(age)

#Using one print function with comma to seperate items.
```

2.3.2 Variable Reassignment

Variables can reference different values while a program is running.

Example:

In [ ]:
```
#Assigning a value to gross pay
gross_pay = 1200
print(gross_pay)

#Reassigning a different value to gross pay
gross_pay = 2600
print(gross_pay)

#Reassigning a string to gross pay
gross_pay = 'undefined'
```

2.4. Data Types and Literals

When you store an item in memory, it is important for you to be aware of the item's data type. Python uses *data types* to categorize values in memory.

- When an integer is stored in memory, it is classified as an **int**.
- When a real number is stored in memory, it is classified as a **float**.
- When a string is stored in memort, it is classified as a **str**.

One important attribute of Python is that it classifies data types on its own. You need to declare the data type beforehand in some programming languages.

Note: type( ) function determines the data type of a variable or a value.

Example:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

2.5. Reading Input from the Keyboard

Most programs need to read input typed by the user on the keyboard.
Built-in input( ) function reads input from keyboard
When a program reads data from the keyboard, usually it stores that data in a variable so it can be used later by the program.

Format:
***variable** = input(**prompt**)*

- **prompt** is a string that is displayed on the screen. it is typically instructing user to enter a value.
- **variable** is the name of a variable that references the data that was entered on the keyboard.

Example:
name = input('what is your name?')

When this statement executes, the following things happen:

- The string 'What is your name? ' is displayed on the screen.
- The program pauses and waits for the user to type something on the keyboard and then to press the Enter key
- When the Enter key is pressed, the data that was typed is returned as a string and assigned to the name variable.

Now lets run the code:

In [ ]:
```
name = input('what is your name? ')
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Note: The input( ) function **always** returns the user's input as a string, even if the user enters numeric data. Python has built-in functions that you can use to convert a string to a numeric type.

Function	Description
`int(item)`	You pass an argument to the `int()` function and it returns the argument's value converted to an `int`.
`float(item)`	You pass an argument to the `float()` function and it returns the argument's value converted to a `float`.

Source:

In [ ]:

In [ ]:
```
type(Height)
```

There are two options to convert a string to a numeric formats:

In [ ]:
```
#Option 1:
string_value = input('what is your age? ')
age = int(string_value)
```

In [ ]:
```
#Option 2:
age = int(input('what is your age? '))
```

Lets show a complete progaram that uses the input function to read a string, an int, and a float, as input from the keyboard.

In [ ]:
```
# Get the user's name, age and income
name = input('what is your name? ')
age = int(input('what is your age? '))
income = float(input('what is your income? '))

#Dispaly the data
print('Name:', name)
print('Age:', age)
```

2.6. Performing Calculations

Python has many operators that can be used to perform mathematical calculations.

Symbol	Operation	Description
+	Addition	Adds two numbers
-	Subtraction	Subtracts one number from another
*	Multiplication	Multiplies one number by another
/	Division	Divides one number by another and gives the result as a floating-point number
//	Integer division	Divides one number by another and gives the result as a whole number
%	Remainder	Divides one number by another and gives the remainder
**	Exponent	Raises a number to a power

Source:Starting Out with Python (4th Edition)- by Tony Gaddis Pearson

2.6.1 Operator Precedence

- Operations that are enclosed in parentheses are performed first.
- When two operators share an operand, the operator with the higher precedence is applied first. The precedence of the math operators, from highest to lowest, are:
  1. Exponentiation: **
  2. Multiplication, division, and remainder: * / // %

3. Addition and subtraction: + −

Example:

In [ ]:
```
salary = float(input('what is your salary? '))
bonus = int(input('what is your bonus? '))

#calculate total pay by adding salary and bonus
pay = salary + bonus

#Dispaly total pay
```

2.7. More About Data Output

2.7.1 Suppressing the print Function's Ending Newline

The print function normally displays a line of output. If you do not want the print function to start a new line of output when it finishes displaying its output, you can pass the special argument end='' to the function.

Example:

In [ ]:
```
#The following three statements will produce three lines of output
print('Alisson')
print('Mark')
```

In [ ]:
```
#The following three statements will produce one lines of output
print('Alisson', end='')
print('Mark' , end='')
```

Note: If we want the print function to print anything at the end of its output, not even a space, we can pass the argument end=''(without space between quote marks) to the print function.

Example:

In [ ]:
```
print('Alisson', end='')
print('Mark' , end='')
```

2.7.2 Specifying an Item Separator

When multiple arguments are passed to the print function, they are automatically separated by a space when they are displayed.

Example:

In [ ]:

Note: If we do not want a space printed between the items, we can use the argument sep='' to the print function.

Example:

In [ ]:

Note: We can also use argument sep to specify a character other than space to seperate multiple items.

Example:

In [ ]:

In [ ]:

2.7.3 Escape Character

An escape character is a special character that is preceded with a *backslash*, appearing inside a string literal.

Escape Character	Effect
\n	Causes output to be advanced to the next line.
\t	Causes output to skip over to the next horizontal tab position.
\'	Causes a single quote mark to be printed.
\"	Causes a double quote mark to be printed.
\\	Causes a backslash character to be printed.

Source:Starting Out with Python (4th Edition)- by Tony Gaddis Pearson

Examples:

In [ ]:

In [ ]:

In [ ]:

Note: When the + operator is used with two strings, it performs *string concatenation*. It appends one string to another.

Example:

In [ ]:
```
message = 'Hello ' + 'world'
```

In [ ]:
```

```

2.8. Displaying Formatted Output with F-strings

F-strings give you a easy way to format the ouput that you want to display with the *print* function.

Example:

In [ ]:
```
name = "John"
```

In [ ]:
```
temperature = 50
```

In [ ]:
```

```

In [ ]:
```
val = 10
```

In [ ]:
```

```

In [ ]:
```
pi = 3.1415926535
```

In [ ]:
```
a = 2
b = 3
```

In [ ]:
```
amount_due = 5000
monthly_payment = amount_due / 12
```

In [ ]:
```
number = 1234567890.1234
```

In [ ]:
```

```

In [ ]:
```
discount = 5
```

In [ ]:
```
discount = 0.5
```

2.8.2 Formatting in Scientific Notation

If you prefer to display floating-point numbers in scientific notation, you can use the letter **e** or the letter **E**.

Examples:

In [ ]:

In [ ]:

2.8.3 Formatting Integers

We can also use the f strings to format integers.
There are two differences when writing a format specifier for displaying integer numbers:

- We use **d or D** as the type designator.
- We cannot specify precision.

Examples:

In [ ]:

In [ ]:

2.8.3 Concatenation with F-strings

In [ ]:
```
name = 'Abbie Lord'
department = 'Sales'
position = 'Manager'
```

In [ ]: