CENG 113
Programming Basics

Elements of Programming

Questions

- Who went over last two presentations?
- Who checked out George Polya's "How to Solve It"?
- Who started to read about Python?

What is programming?

- Programming is the process of taking an **algorithm** and encoding it into a notation, a **programming language**, so that it can be executed by a computer.
- Although many programming languages and many different types of computers exist, the **important first step** is the need to have the solution.
  - **Without an algorithm there can be no program.**

What is a program?

- A **program** is a **sequence of instructions** that specifies how to perform a computation.
- The computation might be something mathematical, such as
  - solving a system of equations or
  - finding the roots of a polynomial,
- can also be a symbolic computation, such as
  - searching and replacing text in a document or
  - compiling a program.

What is a program?

- **Input**: Get data from the keyboard, a file, or some other device.
- **Output**: Display data on the screen or send data to a file or other device.
- **Math**: Perform basic mathematical operations like addition and multiplication.

What is a program?

- **Conditional Execution**: Check for certain conditions and execute the appropriate code.
- **Repetition**: Perform some action repeatedly, usually with some variation

What is debugging?

- Programming is error-prone.
- Programming errors are called **bugs**.
- The process of tracking them down is called **debugging**.

What is debugging?

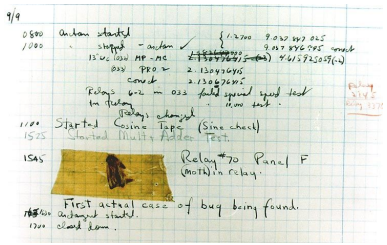
- Programming is error-prone.
- Programming errors are called **bugs**.
- The process of tracking them down is called **debugging**.

Debugging before it was cool:

The concept that software might contain errors dates back to **Ada Lovelace**'s 1843 notes on the analytical engine.

What is debugging?

The term "**bug**" was used in an account in 1947, by computer pioneer Grace Hopper.



What is debugging?

- But actually, it has been a part of engineering jargon for many decades and predates computers and computer software:
  - It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise - this thing gives out and [it is] then that "Bugs" - as such little faults and difficulties are called - show themselves and months of intense watching, study and labor are requisite before commercial success or failure is certainly reached. - **Thomas Edison, 1878**

Programming Errors

- Three kinds of errors can occur in a program:
  - Syntax errors
  - Runtime errors
  - Semantic errors

Syntax errors

Syntax (noun)

1. the arrangement of words and phrases to create well-formed sentences in a language. ("the syntax of English")

2. the structure of statements in a computer language.

Syntax errors

- Python can only execute a program if the syntax is correct; otherwise, the interpreter displays an error message.
- For example, parentheses have to come in matching pairs, so (1 + 2) is legal, but 8) is a syntax error.

Runtime errors

- Runtime error is so called because the error does not appear until after the program has started running.
- These errors are also called exceptions because they usually indicate that something exceptional (and bad) has happened.

Runtime errors

- Runtime errors are rare in the simple programs.
- For example, you cannot divide by zero.

```
a = 0
b = 7 * (10 / a)
```

- This is syntactically correct. But, it'll result in an error: ZeroDivisionError: integer division or modulo by zero

Semantic errors

- Semantic (adj.): relating to meaning in language or logic.
- If there is a semantic error in your program, it will run successfully in the sense that the computer will not generate any error messages, but it will not do the right thing. It will do something else. Specifically, it will do what you told it to do not what you meant it to do.

Semantic errors

- For example, suppose you would like to calculate the hypotenuse of a right-angled triangle:

```
a = 3
b = 4
c = (a**2)+(b**2)
```

- This code will run without any complaint but it will not generate the correct result.
- Identifying semantic errors can be **tricky**.

What is a program?

- Programming languages must provide a notational way to represent both the **data** and the **process**:
  - **Data types**
  - **Control structures**

Values and types

- A **value** is one of the basic things a program works with, like a letter or a number.
  - 2 is an **integer**
  - 1.2 is **float**
- 'Hello, World!' is a **string**, so called because it contains a “string” of letters. You (and the interpreter) can identify strings because they are enclosed in quotation marks.

Values and types

- Built-in function “**type**” in Python
  - With one argument, return the type of an object.

```
>>> type('Hello, world!')
<type 'str'>
>>> type(17)
<type 'int'>
>>> type(3.2)
<type 'float'>
>>> type('17')
<type 'str'>
>>> type('3.2')
<type 'str'>
```

Variables

- A **variable** is a name of the memory location that holds a value.
- An **assignment** statement creates new variables and gives them values:
- (variable) = (value)

```
>>> message = 'What hath God wrought?'
>>> n = 17
>>> pi = 3.1415926535897932
```

Variable names

- Variables names must start with a letter or an underscore, such as:
  - _underscore
  - underscore_
- The remainder of your variable name may consist of letters, numbers and underscores.
  - password1
  - n00b
  - un_der_scores

Variable names

- Names are case sensitive.
  - case_sensitive, CASE_SENSITIVE, and Case_Sensitive are each a different variable.
- Variable names cannot be among the keywords used in the programming language.
  - **In Python 3.5.2, there are 33 keywords:** 'False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'

Variable names

- What is wrong with the following?

```
>>> 76trombones = 'big parade'
SyntaxError: invalid syntax
>>> more@ = 1000000
SyntaxError: invalid syntax
>>> class = 'Advanced Theoretical
Zymurgy'
SyntaxError: invalid syntax
```

Operators and operands

- **Operators** are special symbols that represent computations like addition and multiplication.
- The values or variables the operator is applied to are called **operands**.
  - The operator '=' we have seen previously is an **assignment operator**.
  - The operators '+', '-', '*', '/' and '**' perform **addition**, **subtraction**, **multiplication**, **division** and **exponentiation** with numerical operands.

Order of operations

- When there is more than one operator in an expression, the order of evaluation depends on the rules of precedence.
- For mathematical operators, Python follows mathematical convention.
  - The acronym **PEMDAS** is a useful way to remember the rules: **P**arentheses, **E**xponentiation, **M**ultiplication and **D**ivision, **A**ddition and **S**ubtraction
  - Operators with the same precedence are evaluated from left to right.

Exercise

- The volume of a sphere with radius r is $4/3\pi r^3$. What is the volume of a sphere with radius 5?

Exercise

- The volume of a sphere with radius r is $4/3\pi r3$. What is the volume of a sphere with radius 5?

```
>>> radius = 5
>>> pi = 3.14
>>> sphere = (4 / 3) * pi * radius ** 3
>>> print(sphere)
```

String operations

- The + operator works with strings, but it might not do what you expect: it performs concatenation.

```
>>> first = 'Good'
>>> second = 'Day'
>>> print(first + second)
```

- The output of this program is 'GoodDay'.

String operations

- The * operator also works on strings; it performs repetition.

```
>>> print('Spam'*3)
```

- The output is 'SpamSpamSpam'.

Expressions

- An **expression** is a combination of values, variables, and operators.
- A value all by itself is considered an expression, and so is a variable, so the following are all legal expressions (assuming that the variable x has been assigned a value):
  - 17
  - x
  - x + 17

Statements

- A **statement** is a unit of code that the Python interpreter can execute.
- We have seen one kind of statement:
  - assignment
- A program is formed by a sequence of one or more statements.

Comments

- As programs get bigger and more complicated, they get more difficult to read.
- It is a good idea to add notes to your programs to explain in natural language what the program is doing. These notes are called **comments**, and they start with the # symbol.

Comments

- Everything from the # to the end of the line is ignored - it has no effect on the program.

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

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

Inputs from keyboards

```
# Interactive program to input user info.
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = input("Your age? ")
print("So, you are are already " + age + " years old,
" + name + "!")
```

```
>>> What's your name? Frank
>>> Nice to meet you Frank!
>>> Your age? 30
>>> So, you are are already 30 years old, Frank!
```

Exercise

- Suppose the cover price of a book is $24.95, but bookstores get a 40% discount. Shipping costs $3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

Exercise

- Suppose the cover price of a book is $24.95, but bookstores get a 40% discount. Shipping costs $3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?
- (Write using **input** function!)

Exercise

- A book club promises to send 8 books for $1 each, if you join the club. After you receive the first 8 books, you may select more books at a rate of $19.99 per book. If you spend a total of $80.96, how many extra books did you purchase?

Exercise

- If I leave my house at 6:52 am and run 1 mile at an easy pace (8 minutes per mile), then 3 miles at tempo (6 minutes per mile) and 2 mile at easy pace again, what time do I get home for breakfast?