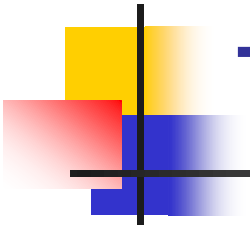
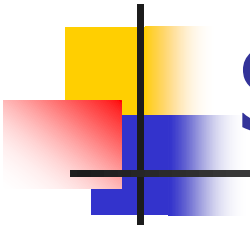
ICT 133
Structured Programming

Seminar 1



Topics

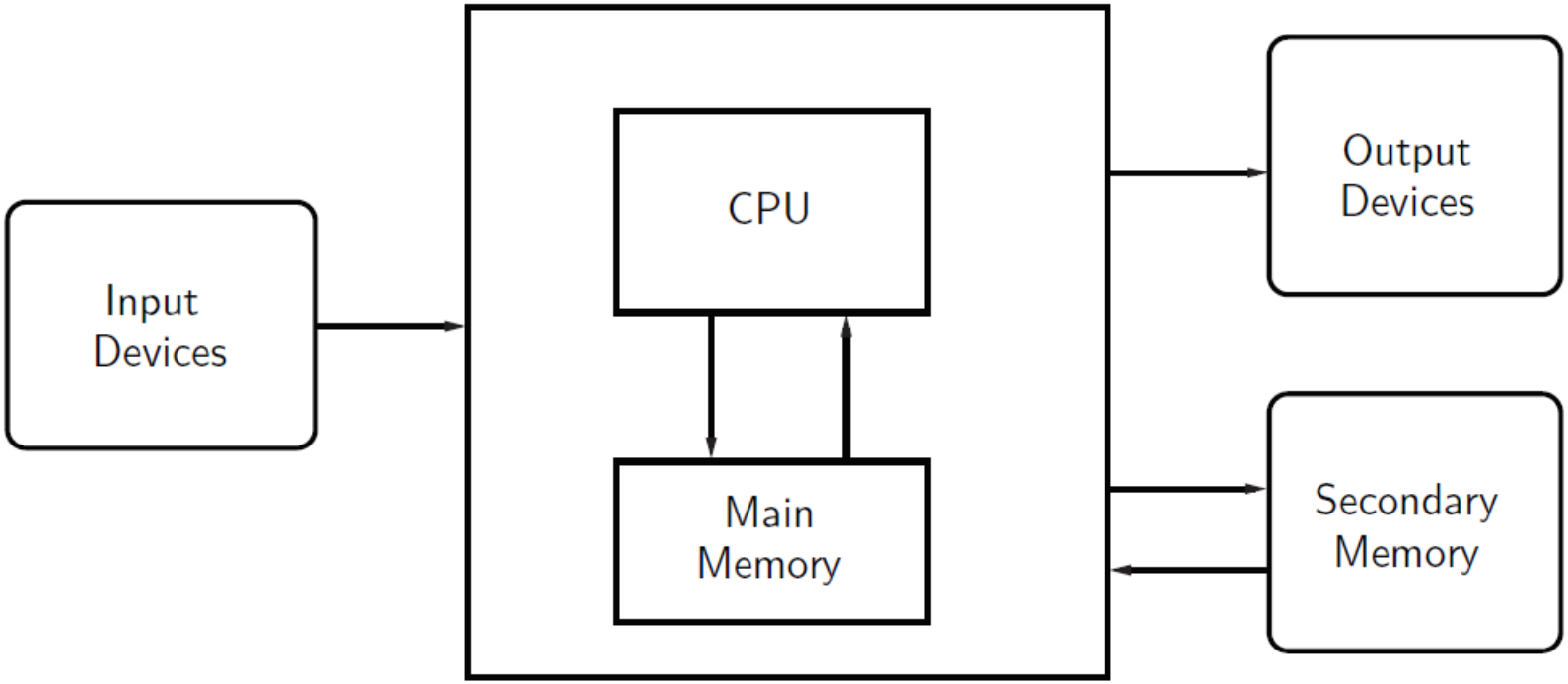
- The roles of hardware and software in a computing system.
- The software development process
- Python programming language.
- Programming with numbers



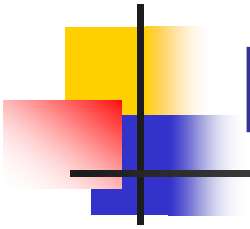
Software vs Hardware

- Computer program - software
  - set of instructions
- Computer - hardware
  - executes the instructions

Hardware

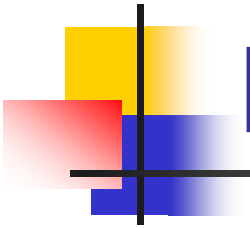

Input
Devices
CPU
Main
Memory
Output
Devices
Secondary
Memory

Natural Language vs Programming Language



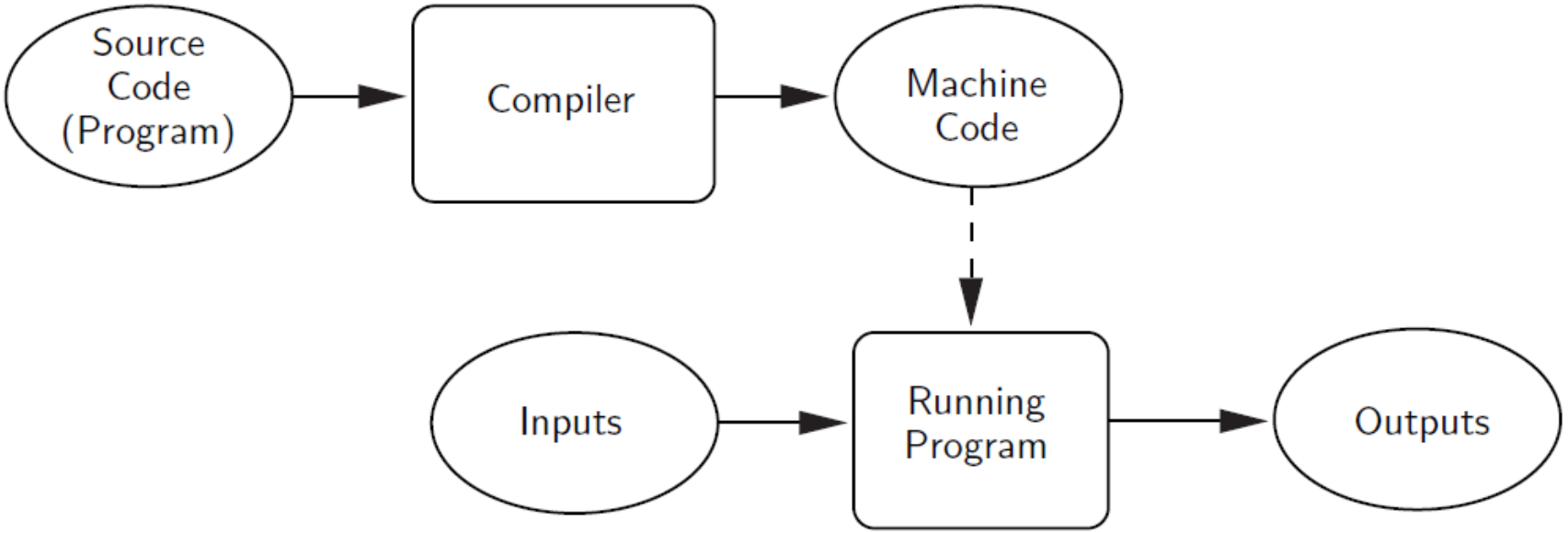
- Natural language
  - Ambiguous and imprecise
- Programming language
  - Unambiguous and precise
  - precise form - *syntax*
  - precise meaning - *semantics*

Types of Programming Languages

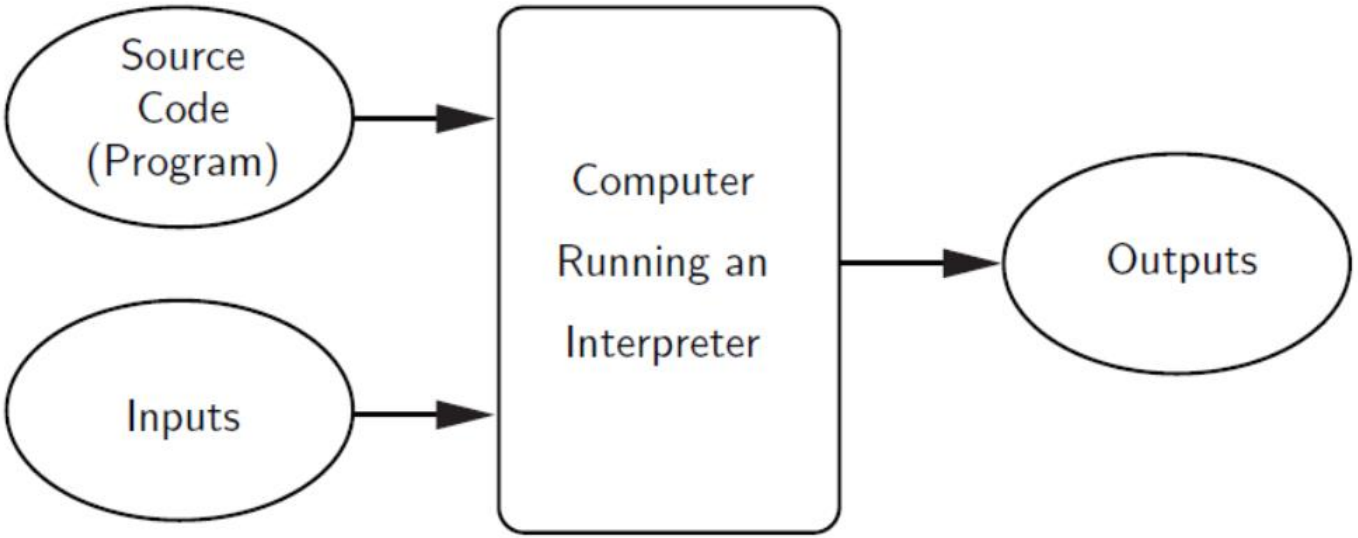


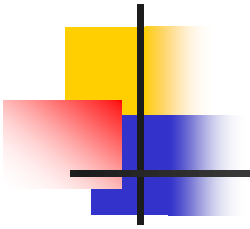
- Low-level or machine language
  - in 0s and 1s E.g., 1100001000000001
- Assembly language
  - Uses mnemonics instead of 0s and 1s
  - E.g., ADD R2, R0, R1
- *High-level* computer languages
  - Understood by humans
  - E.g., c = a + b

Compiler


Source Code (Program)
Compiler
Machine Code
Inputs
Running Program
Outputs

Interpreter





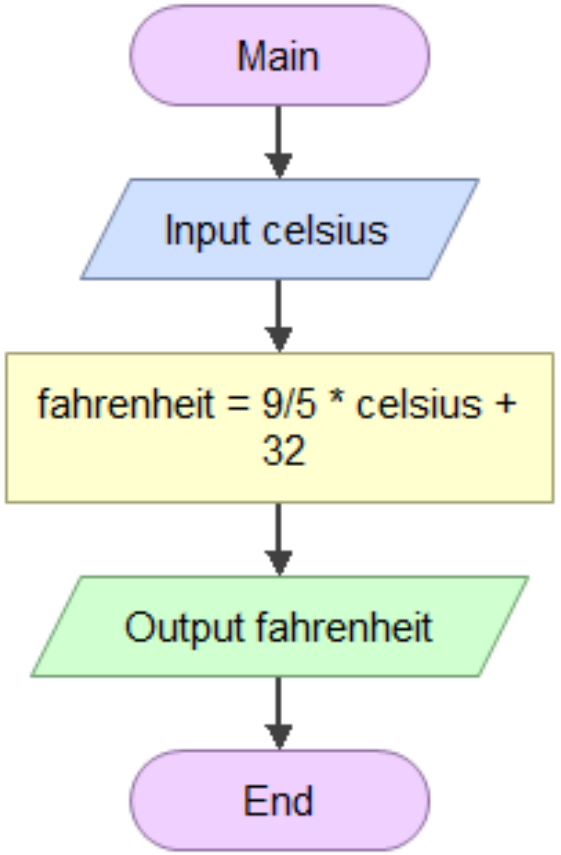
Software Development

- **Analyze the Problem**
  - What problem to solve
- **Determine Specifications**
  - What program must do
- **Create a Design**
  - What steps can solve - the algorithm

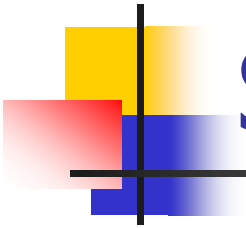
Algorithm

Input	**Input celsius**
Processing	**fahrenheit = 9/5 celsius + 32**
Output	**Output fahrenheit**

Pseudocode

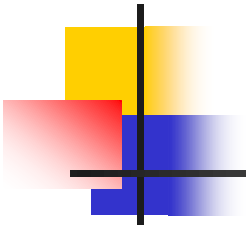


Flowchart



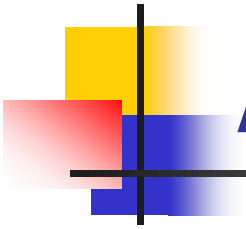
Software Development

- **Implement the Design**
  - The steps in programming language
- **Test/Debug/Run the Program**
- **Maintain the Program**



Python

- Created by Guido van Rossum
- Released in 1991
- Multiple programming paradigms: object-oriented, imperative, functional and procedural
- Large and comprehensive standard library
- Current version 3.7.2

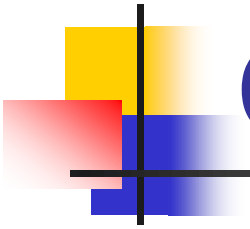


A First Python Program

```
def main():
    print("Hello World!")

main()
```

- Case Sensitive
- Indent block of statements



Output statement

print(expr$_1$, ..., expr$_n$)

```
print("Hello world")
print(2+3)
print("2+3 =", 2+3)
```

Output:

```
Hello world
5
2+3 = 5
```

Output statement

print(expr$_1$, ..., expr$_n$, end = " ")

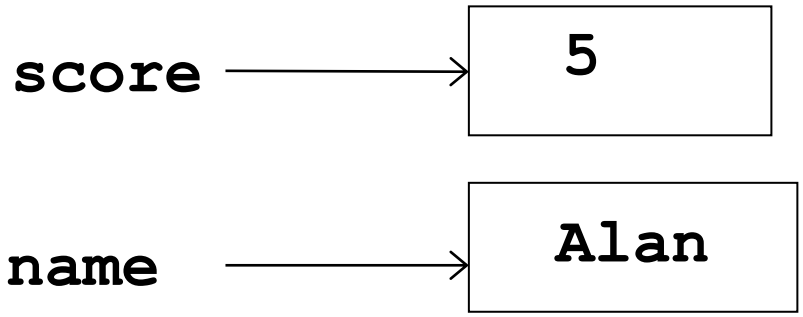
```
print("Hello world", end = " ")
print(2+3, end = " ")
print("2+3 =", 2+3)
```

Output:

```
Hello world 5 2+3 = 5
```

Variable

- A name given to a value.

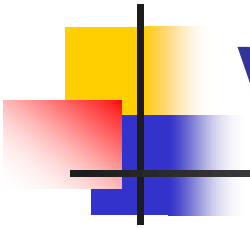


- Use the variable to refer to the value. e.g.,

```
print(name, "scored", score, "points")
```

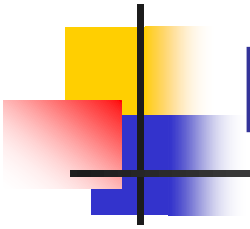
Output: `Alan scored 5 points`

Variable Name



- Case sensitive
- Cannot be any Python keyword
- Only uppercase and lowercase alphabets, digits and _
- No spaces, commas and symbols such as $, &, % and *
- Cannot start with a digit
- Are the following variable names acceptable?

count1	total-price	for	1stName
high_Score	maxDiscount		$amount



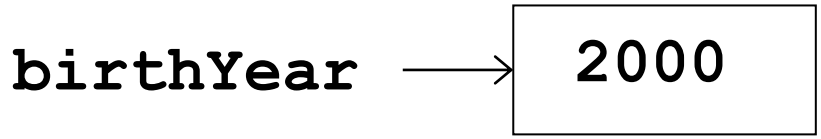
Python Keywords

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

Assignment statement

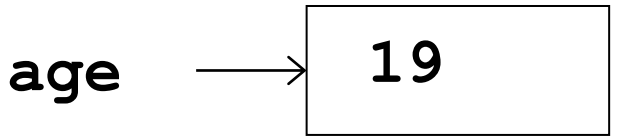
var = value

```
birthYear = 2000
```




```
age = 2019 - birthYear
```

expression



Simultaneous Assignment Statement

var_1, ..., var_n = $value_1$, ..., $value_n$

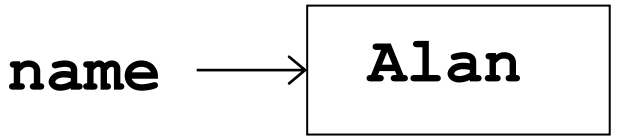
```
x, y = 5, 2
sum, diff = x+y, x-y
x, y = y, x
```

Input statement

input (prompt)

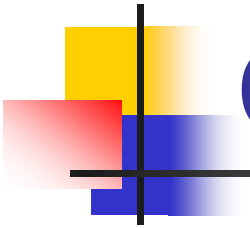
```
name = input("Enter friend's name: ")

print("Hello", name)
```




```
Enter friend' name: Alan
Hello Alan
```

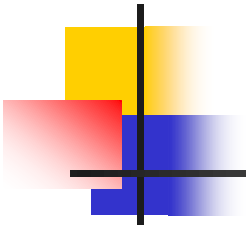
- All user input are values of str type
- A str consists of zero or more characters.



Comment

 # comment

```
# A simple program that prints out Hello World!
def main():
    print("Hello World!") #prints Hello World!
```



Number Data Types

- int
  - Whole number, exact
- float
  - Decimal number, imprecise

Type Conversion

- int()

```
score = int(input("Enter score: "))
```

Recall that all user input are values of str type

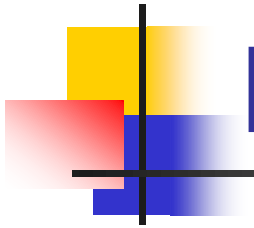
- float()

```
weight = float(input("Enter weight: "))
```

A str must be converted to numeric types before performing arithmetic operations on them

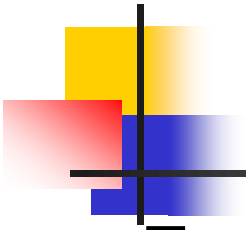
Arithmetic (Binary) Operators

Operator	Use	Description	precedence
+	x + y	Add x and y	
-	x - y	Subtract y from x	lowest
*	x * y	Multiply y by x	
/	x / y	Divide x by y	
//	x // y	Divide x by y, result in int	next highest
%	x % y	Compute remainder after dividing x by y	
**	x ** y	x raised to the power of y	highest



Examples

3 + 4.0	7.0
3 + 4	7
10.0 / 3.0	3.3333333333333335
10 / 3	3.3333333333333335
10 // 3	3
1/2	0.5
1//2	0
10.5 % 3.0	1.5
4 % 5	4



Expressions

- Evaluates to value

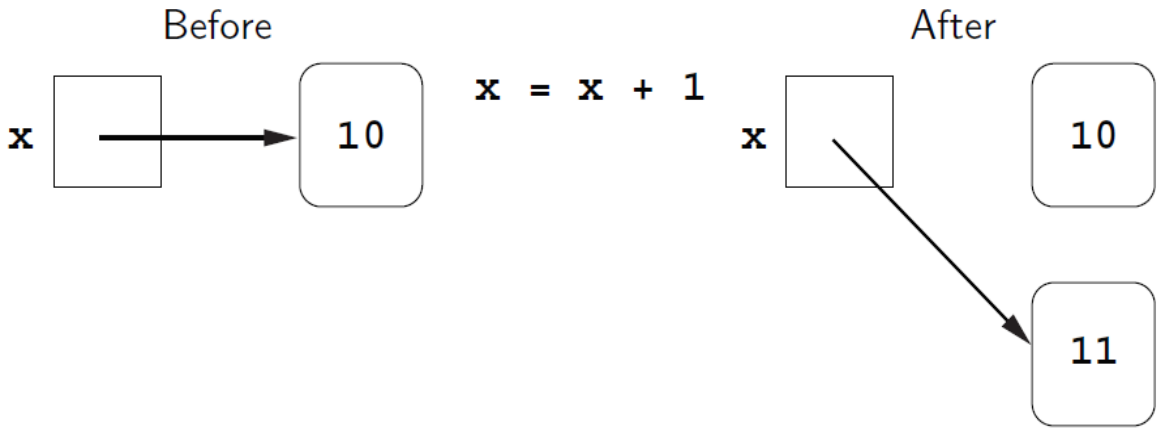
  e.g., `0.5 * mass * pow(c,2)`

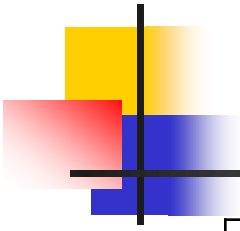
- May include:
  - Literals
  - Identifiers
  - Operators and function calls

Increment

```
x = x + 1
```

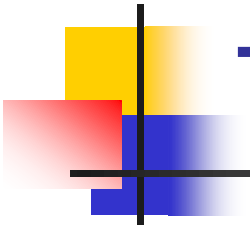
Once the value on the RHS is computed, it is stored back into (*assigned*) into x





Special Assignment Operators

Operator	Use	Description
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
//=	x //= y	x = x // y
%=	x %= y	x = x % y



Type Conversion

- Implicit typing
  - *mixed-typed expressions* :

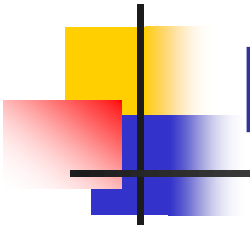
    `3 + 4.0` evaluates to `7.0`

- Explicit typing

  `3 + int(4.0)` evaluates to `7`

  `int("32")` evaluate to `32`

  `float("32")` evaluate to `32.0`



Rounding

- `round(n)`, to the nearest whole number
  - `round(3.912)` evaluates to `4`

- `round(n, p)`, to another float with p decimal digits
  - `round(3.912, 2)` evaluates to `3.91`
  - `round(145/2)` evaluates to `72` - why?

Using the Math Library

Python	Mathematics	English
pi	π	An approximation of pi
e	e	An approximation of e
sqrt(x)	$\sqrt{x}$	The square root of x
sin(x)	$\sin x$	The sine of x
cos(x)	$\cos x$	The cosine of x
tan(x)	$\tan x$	The tangent of x
asin(x)	$\arcsin x$	The inverse of sine x
acos(x)	$\arccos x$	The inverse of cosine x
atan(x)	$\arctan x$	The inverse of tangent x

Using the Math Library

Python	Mathematics	English
log(x)	ln x	The natural (base e) logarithm of x
log10(x)	$\log_{10} x$	The common (base 10) logarithm of x
exp(x)	e^x	The exponential of x
ceil(x)	$\lceil x \rceil$	The smallest whole number >= x
floor(x)	$\lfloor x \rfloor$	The largest whole number <= x

Using the Math Library

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
from math import sqrt, pow

a, b, c = 6, 11, -35
discrim = sqrt(pow(b, 2) - 4*a*c)
print("First root =", (-b + discrim )/(2 * a))
print("Second root =", (-b - discrim )/(2 * a))
```