



SC1003/SC5001 Introduction to computational thinking and programming

A large, irregular blue ink splatter or watercolor blotch serves as the background for the title text. The splatter has a textured, painterly appearance with various shades of blue and white, creating a dynamic and artistic look.

Course Review Flowchart Pseudocode

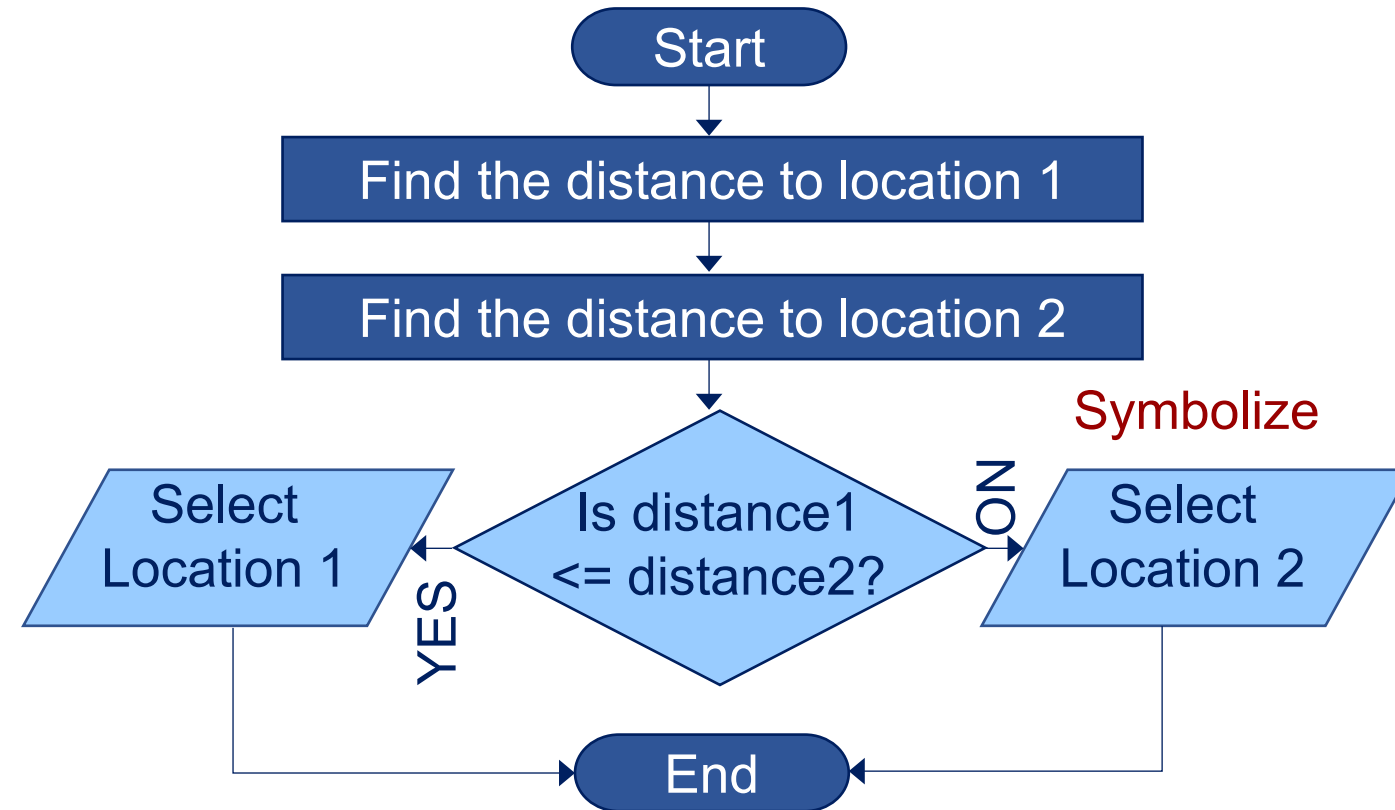


Which of the following statements is true regarding pseudocode and flowcharts?

① Start presenting to display the poll results on this slide.

Flowchart vs. Pseudocode

Flowchart

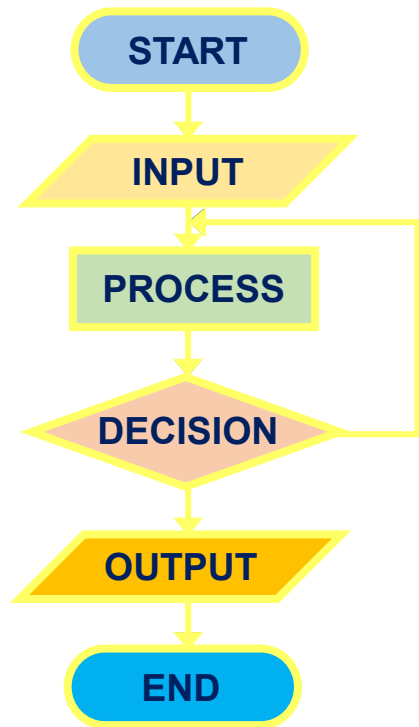






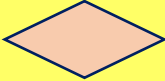
Symbolize

Pseudocode

```
FIND the distance to location 1
FIND the distance to location 2
IF distance1 <= distance2
    SELECT Location 1
ELSE
    SELECT Location 2
END IF
```

Flowchart: a representation of an algorithm using diagram for effective visualization



Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of a program
Flow line		Denotes the direction of logic flow in a program
Parallelogram		Denotes either an input operation (e.g. INPUT) or an output operation (e.g. PRINT)
Rectangle		Denotes a process to be carried out (e.g. an addition)
Diamond		Denotes a decision or branch to be made; the program should continue along one of two routes

1. Understand the Process or Problem

- **Clarify the purpose** of the flowchart: Is it to explain a process, solve a problem, or plan a system?
- Gather all relevant **steps, decisions, inputs, and outputs**.

Number Guessing game

2. Identify the Start and End Points

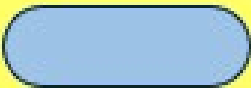


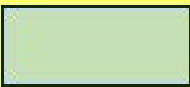

- Decide **where the process begins** and **where it ends**.
- These will be represented using **Terminator symbols** (rounded rectangles or ovals).

3. List All Steps Sequentially

- Break the process into **clear, manageable steps**.
- Note any **decisions, calculations, or actions** required.

4. Choose Flowchart Symbols

Use standard flowchart symbols:

Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of a program
Flow line		Denotes the direction of logic flow in a program
Parallelogram		Denotes either an input operation (e.g. INPUT) or an output operation (e.g. PRINT)
Rectangle		Denotes a process to be carried out (e.g. an addition)
Diamond		Denotes a decision or branch to be made; the program should continue along one of two routes

5. Sketch the Flow

- Start with the "**Start**" symbol.
- Add each step or decision in sequence.
- Use **arrows** to connect the symbols logically and clearly.

6. Add Decision Points

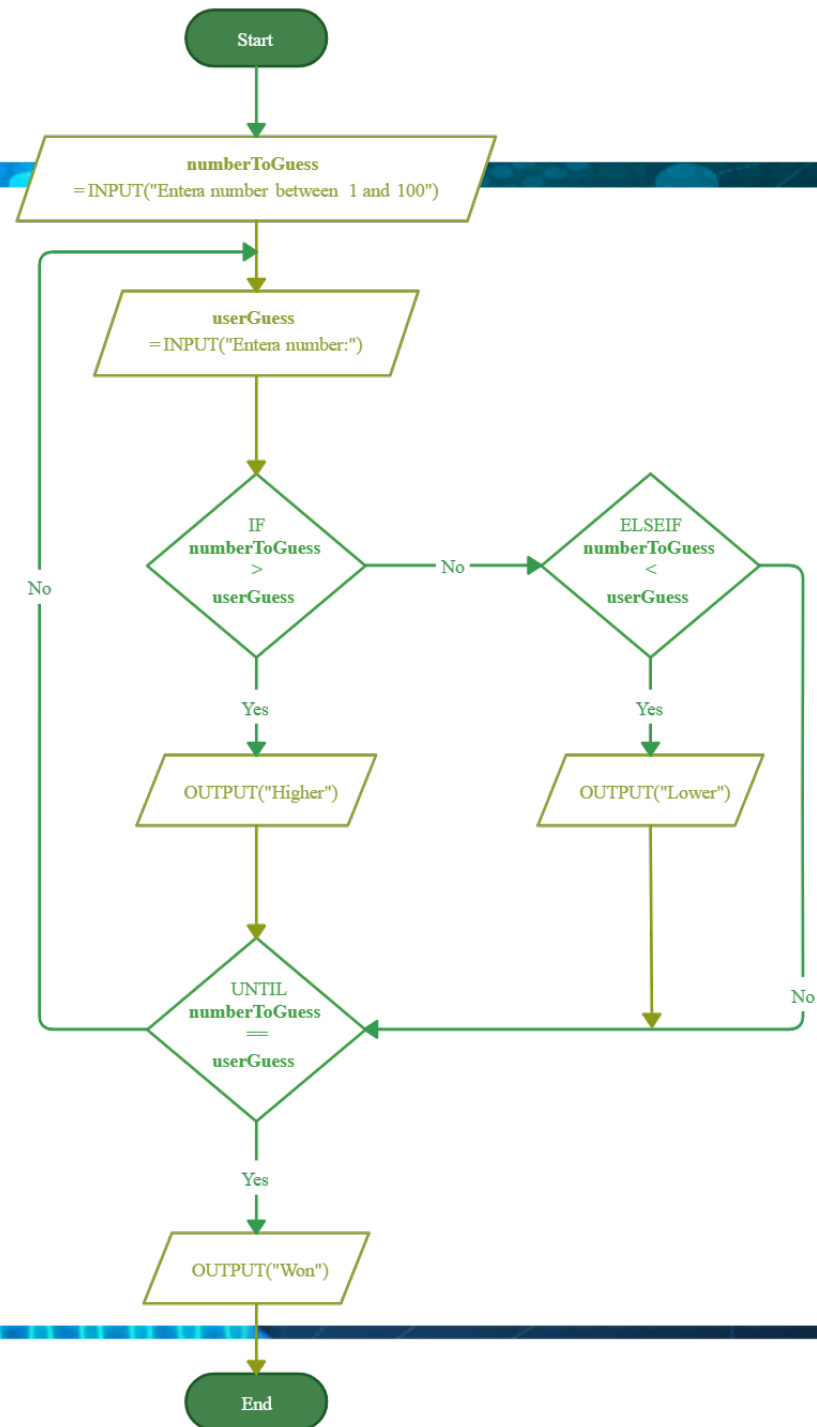
- For each condition or decision, use a **diamond** symbol.
- Ensure each decision has **two or more branches** (e.g., yes/no).

7. Review and Refine

- Ensure the flow is **logical** and **sequential**.
- Check for missing steps or unclear transitions.
- Keep it **simple** and **uncluttered**.

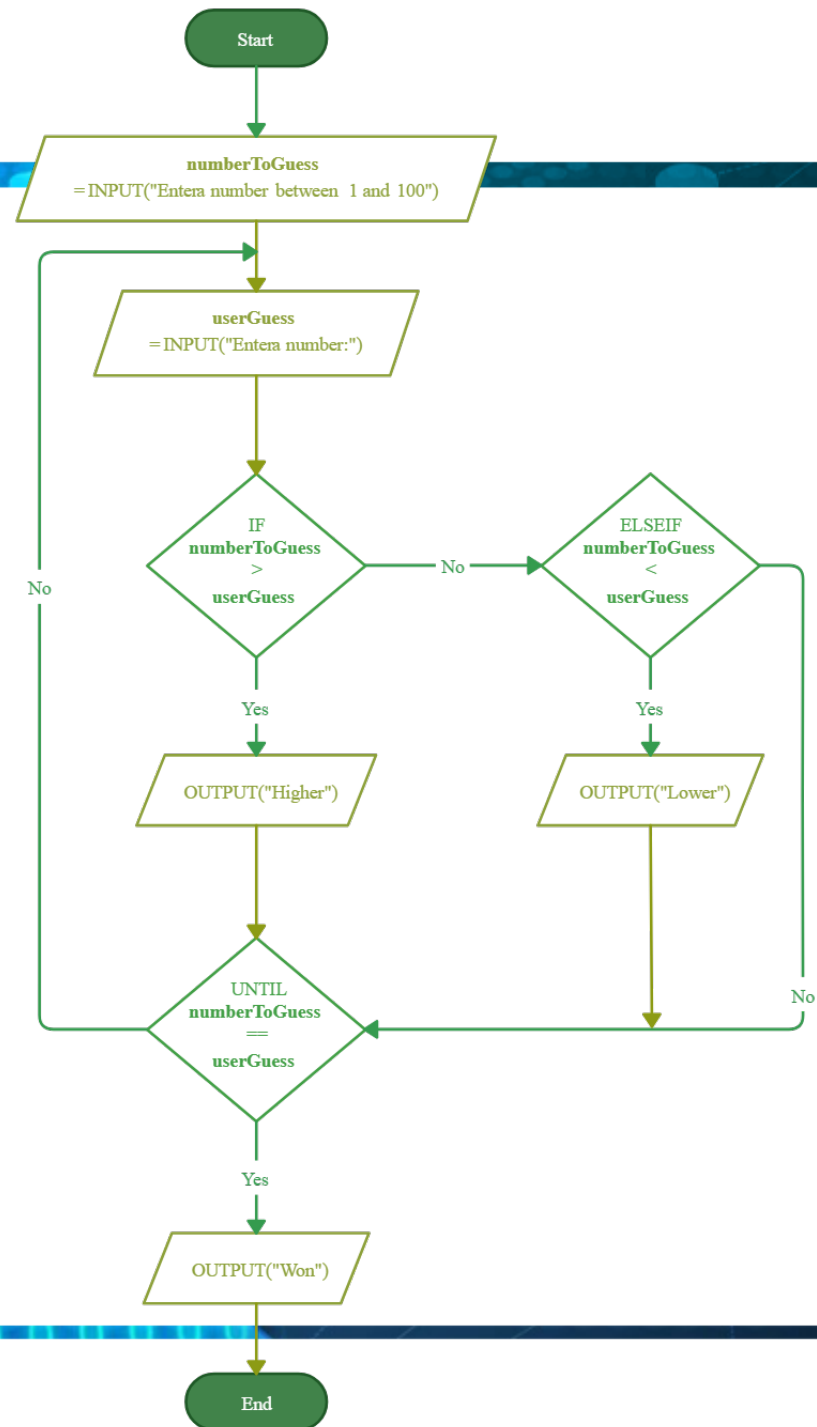
8. Test the Flowchart

- Walk through the process using the flowchart.
- Make sure it accurately represents the intended logic or procedure.





Is the previous flowchart for the number guessing game correct?



9. Finalize and Digitize (Optional)

- Redraw neatly by hand or use tools like:
 - **Lucidchart**
 - **draw.io**
 - **Microsoft Visio**
 - **Google Drawings**
- Label all steps clearly.

Pseudocode: pronounced as /'s(j)u:dəʊ,kəʊd/ 

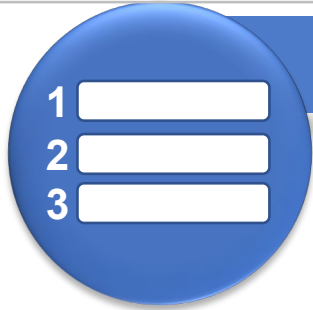
- IDEA: directly uses informal English to describe an algorithm step by step with one step per line
- Uses the structural conventions of a normal programming language
 - but is intended for human reading rather than machine reading

General Notes:

- No strict rules
- Uses informal language – combination of English and keywords

Common Keywords	Other Keywords
IF, ELSE, WHILE	READ, PRINT, INITIALIZE, COMPUTE, ADD, SUBTRACT

- Usually starts an operation sentence with a verb (description should be **concise** and **precise**)



Guidelines

- *Write one statement per line only.*
- *Capitalize the keywords.*
- *Indent to show hierarchy.*
- *End multi-line structures.*
- *Keep statements programming-language independent.*

<https://www.techgeekbuzz.com/how-to-write-pseudocode/>

How to Write Pseudocode? A Beginner's Guide with Examples

Posted in **PROGRAMMING LANGUAGE**



Operators

1. Assignment Operator:

=, <- or :=

2. Comparison Operator:

== , !=, <, >, <= , and >=

3. Arithmetic Operator:

+, -, *, /, MOD(%)

4. Logical Operator:

AND, NOT and OR

5. Sum, Product:

??

Special Keyword

1. **START**: To begin the pseudocode.

2. **INPUT** : Take input from the user.

3. **PRINT**: To print the output on the screen.

4. **READ/GET**: Input format while reading data from the file.

5. **SET, INIT**: Initialize a value.

6. **INCREMENT, BUMP**: Increase the value of the variable, equivalent to a++.

7. **DECREMENT**: Decrease the value of the variable, equivalent to a--.

8. **COMPUTE, CALCULATE, DETERMINE**: To calculate the expression result.

Conditional Statements:

if

```
IF condition
  THEN if body
ENDIF
```

if...else

```
IF condtion THEN
  if body
ELSE
  else body
ENDIF
```

if...else if...else

```
IF condition statement THEN
  if body
ELSE IF condition THEN
  else if statement
ELSE
  else body
ENDIF
```

for loops:

```
FOR initial_value TO end_value
  for body
ENDFOR
```

Example:

```
FOR i -> 0 to 20
  PRINT i
ENDFOR
```

while loop

```
WHILE condition
  while body

ENDWHILE
```

Number guessing game

```
BEGIN

Generate a random number n between 1 and 99

PROMPT user to "Enter an integer from 1 to 99"
READ guess

WHILE guess is not equal to n DO
  IF guess < n THEN
    PRINT "Guess is low"
  ELSE IF guess > n THEN
    PRINT "Guess is high"
  ENDIF

  PROMPT user to "Enter an integer from 1 to 99"
  READ guess
ENDWHILE

PRINT "You guessed it!"

END
```




Course Review
Identifier
datatype
Input
output

Variables - Place holders (Naming convention)

Keywords – Special words reserved in Python

Data types – assignment to variables

Built in functions – libraries

Argument input and output to function calls

Identifier: a name given to an entity in Python

- Helps in differentiating one entity from another
- Name of the entity must be unique to be identified during the execution of the program



Rules for Writing Identifiers

What can be used?

- Uppercase and lowercase letters A through Z ($26 * 2 = 52$)
- The underscore, '_' (**1**)
- The digits 0 through 9, except for the first character (**10**)

$$52 + 1 + 10 = 63$$



Syntax Rules in Python

- Must begin with a letter or _
 - 'Ab123' and '_b123' are ok
 - '123ABC' is not allowed
- May contain letters, digits, and underscores

this_is_an_identifier_123
- Should **not** use keywords

- Upper case and lower case letters are different

'LengthOfRope' is **not** **'lengthofrope'**

 *Python is **case sensitive***

- Can be of any length
- Names starting with _ have special meaning

Keywords

- Special words reserved in Python
- Programmers should **not** use keywords to *name* things

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

slido

Please download and install the
Slido app on all computers you use



**Which of the following is true
about identifiers in Python?**

① Start presenting to display the poll results on this slide.



**Which of the following
options correctly identifies
the valid Python identifiers?**

① Start presenting to display the poll results on this slide.

Python Naming Conventions

```
import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat
```



VS.

```
import math
a = input("Enter the radius of your circle:")
b = float (a)
c = 2 * math.pi * b
d = math.pi * b * b
```



What is c? It is not immediately clear.

- Both programs work
 - They are different when **readability counts**
-
- variable names should be in lowercase, with words separated by underscores as necessary to improve readability
e.g. **radius_float**
 - mixedCase is allowed
e.g. **radiusFloat**

A large, irregular blue ink splatter or watercolor blotch serves as the background for the title text. The splatter has a textured, painterly appearance with various shades of blue and white, creating a dynamic and artistic look.

Course Review

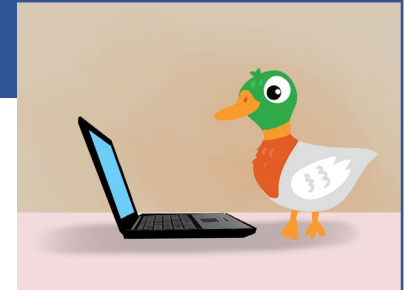
Data type



Compared to C and Java, how does Python know the data types?

Python uses *Duck-Typing*

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.” – James Whitcomb Riley



Examples

```
>>> a = 99
>>> b = 99.9
>>> c = '100'
>>> d = True
```



Four variables!




What are their data types?

Data Types (Cont'd)

Type Function

In Python, the **type()** function allows you to know the type of a **variable** or **literal**.



```
>>> x = 9
>>> type (x)
<class 'int'>
>>> x = 7.8
>>> type(x)
<class 'float'>
>>> x = "Welcome"
>>> type (x)
<class 'str'>
>>> x = 'Python'
>>> type (x)
<class 'str'>
>>> type (8.9)
<class 'float'>
```

- Python does not have variable declaration, like Java or C, to announce or create a variable.
- A variable is created **by just assigning a value to it** and the type of the value defines the type of the variable.
- If another value is re-assigned to the variable, **its type can change**.

Assignment Operator

```
import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat
```

Basic Syntax

Left Hand Side (LHS) = Right Hand Side (RHS)



a variable



an expression

Assignment means:

1. Evaluate the expression on RHS
2. Take the resulting value and assign it to the name (variable) on the LHS.

String - designated as **'str'**

- It is basically a sequence, typically a sequence of characters delimited by single quote ('...') or double quotes ("...") for a single line sentence, even triple quotes (""...") to display a paragraph with multiple lines
- First *collection type* that was discussed
- Collection type contains multiple objects organized as a single object



Examples

```
>>> a = 'Length'
>>> b = "8225 welcome"
>>> c = "ewwew sdcd &8 $5##"
>>> d = 'ewwew sdcd &8 $5##'
>>> e = '''

Welcome to Stock Master!

Please select a function to continue:

(1) Search for a stock code by
keyword

(2) View stock trend by code

(sample input 1 or 2)

'''
```

Basic string operation-string concatenation

String concatenation is the operation of joining two or more strings together. In Python, you can concatenate strings using the + operator or by using formatted strings.

```
string1 = "Hello"  
string2 = "World"  
result = string1 + " " + string2  
print(result)    # Output: Hello World
```

str() function

The `str()` function in Python is used to convert an object to a string representation. This is useful when you need to output or store data as a string. Here are some key details about the `str()` function:

`str(object)`

`object`: The object you want to convert to a string. This can be any Python object, including numbers, lists, dictionaries, tuples, etc.

Example: When you need to concatenate a non-string type with a string, `str()` is handy:

```
age = 25
message = "I am " + str(age) + " years old."
print(message)    # Output: I am 25 years old.
```


A large, irregular blue ink splatter or blotch serves as the background for the title text. The splatter has a textured, painterly appearance with various shades of blue and white, creating a dynamic and artistic look.

FUNCTION CALL INPUT, OUTPUT

INPUT, PROCESSING, AND OUTPUT

- Typically, computer performs three-step process
 - Receive input
 - Input: any data that the program receives while it is running
 - Perform some process on the input
 - Example: mathematical calculation
 - Produce output

```
# 1. prompt user for the radius
# 2. apply circumference and area formulae
# 3. print the results

import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat

print()    # print a line break
print ("The circumference of your circle is:",circumference, ", and the area
is:", area)
```




		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Passing Arguments to Functions

- Argument: piece of data that is sent into a function
 - Function can use argument in calculations and processing
 - When calling the function, the argument is placed in parentheses following the function name

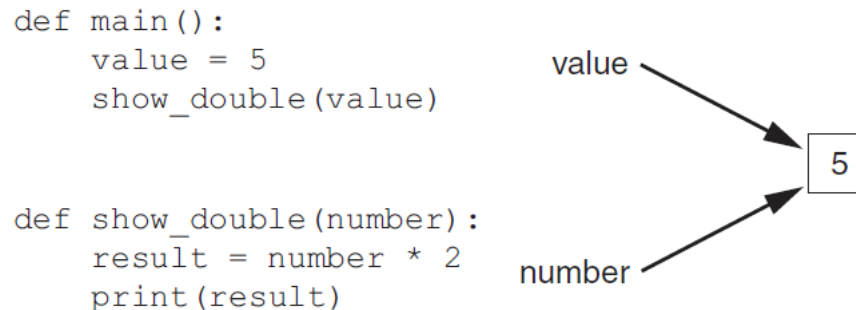
Figure 5-13 The `value` variable is passed as an argument

```
def main():  
    value = 5  
    show_double(value)  
      
  
def show_double(number):  
    result = number * 2  
    print(result)
```

Parameter variable in Functions

- Parameter variable: variable that is assigned the value of an argument when the function is called
 - The parameter and the argument reference the same value
 - General format:
 - `def function_name(parameter) :`
 - Scope of a parameter: the function in which the parameter is used

Figure 5-14 The `value` variable and the `number` parameter reference the same value



Reading Input from the Keyboard

- Most programs need to read input from the user
- Built-in `input` function reads input from keyboard
 - Returns the data as a string, even if the user enters numeric data.
 - Format: `variable = input(prompt)`
 - `prompt` is typically a string instructing user to enter a value
 - Does not automatically display a space after the prompt

```
math_str = input('What is your Math score?')
```



```
What is your Math score?81
```

slido

Please download and install the
Slido app on all computers you use



What does the following Python code do?
`name = input("Enter your name: ")`
`print("Hello, " + name + "!")`

① Start presenting to display the poll results on this slide.

Reading Numbers with the `input` Function

- `input` function always returns a string
- Built-in functions convert between data types
 - `int(item)` converts *item* to an `int`
 - `float(item)` converts *item* to a `float`
 - Type conversion only works if item is valid numeric value, otherwise, throws exception

```
math_str = input('What is your Math score? ')
math_float = float(math_str)
print(math_float)
```



```
What is your Math score? 81
81.0
```



**What is the correct way to
read an integer from the user
in Python?**

① Start presenting to display the poll results on this slide.

Passing Multiple Arguments

Python allows writing a function that accepts multiple arguments

- Parameter list replaces single parameter
 - Parameter list items separated by comma

Arguments are passed *by position* to corresponding parameters

- First parameter receives value of first argument, second parameter receives value of second argument, etc.

Passing Multiple Arguments (cont'd.)

Figure 5-16 Two arguments passed to two parameters

```
def main():  
    print('The sum of 12 and 45 is')  
    show_sum(12, 45)
```

```
def show_sum(num1, num2):  
    result = num1 + num2  
    print(result)
```





What will be the output of the following Python code?

```
def greet(name, times=2):  
    return ("Hello " + name + "! ") * times  
  
print(greet("Alice"))
```

Displaying Output with the `print` Function

```
print(  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

- `print` function: displays output on the screen
- Argument: data given to a function
 - Example: data that is printed to screen
- Statements in a program execute in the order that they appear
 - From top to bottom

```
print ("Hello")  
print ('Python')  
print ("1003")
```



```
Hello  
Python  
1003
```

Displaying Multiple Items with the `print` Function

- Python allows one to display multiple items with a single call to `print`
 - Items are separated by commas when passed as arguments
 - Arguments displayed in the order they are passed to the function
 - Items are automatically separated by a space when displayed on screen while using default sep

```
print(  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
print("Hello", 'Python', "1003") ➡ Hello Python 1003
```

Displaying Multiple Items with the `print` Function cont'

- `print` function uses space as item separator
 - Special argument `sep='delimiter'` causes `print` to use *delimiter* as item separator

```
print(  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
print("Hello", "Python", "1003", sep="#_#")
```




Hello#_#Python#_#1003

argument end in print function

- `print` function displays line of output
 - Newline character at end of printed data
 - Special argument `end='delimiter'` causes `print` to place *delimiter* at end of data instead of newline character

```
print(  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
print ("Hello",end = '*')  
print ('Python',end = '@')  
print ("1003")
```



```
Hello*Python@1003
```

slido

Please download and install the
Slido app on all computers you use



What will the following Python code output?
`print("Hello", "World", sep="-", end="!")`
`print(" How are you?")`

① Start presenting to display the poll results on this slide.

print function with multiple values

```
name = "Alice"
```

```
age = 21
```

```
country = "Singapore"
```

```
print("Hello, my name is", name, ".", "I am", age, "years old and I  
live in", country, ".")
```



```
Hello, my name is Alice. I am 21 years old and I live in Singapore.
```

using the `str()` function and string concatenation with the `+` operator to combine the variables into a single string

```
name = "Alice"
```

```
age = 21
```

```
country = "Singapore"
```

```
print("Hello, my name is " + name + ". I am " + str(age) + " years  
old and I live in " + country + ".")
```



Hello, my name is Alice. I am 21 years old and I live in Singapore.

print function with an f-string

F-strings are available in Python 3.6 and later versions and provide a more readable and concise way to format strings, especially when dealing with multiple variables.

purpose of the f in an f-string: tells Python to format the string

```
name = "Alice"
```

```
age = 21
```

```
country = "Singapore"
```

The f-string allows you to directly insert variables into a string by placing them inside curly braces {}.

Using f-string to format the output

```
print(f"Hello, my name is {name}. I am {age} years old and I live in {country}.")
```

slido

Please download and install the
Slido app on all computers you use



How can you include a variable's value and a string literal side by side using an f-string?

① Start presenting to display the poll results on this slide.

f-string: formatting numbers

f-strings in Python can handle formatting numbers, including specifying the number of digits after the decimal point, adding padding, and more.

```
value = 3.14159
formatted_value = f"{value:.2f}"
print(formatted_value)    # Output: 3.14
```

f-string: Formatting a Number as a Percentage

`:.2%:` This formats the number as a percentage with two decimal places.

The number 0.1234 is converted to 12.34%.

If you want to format the percentage without any decimal places, you can specify 0 decimal places:

```
value = 0.1234
formatted_percentage = f"{value:.2%}"
print(formatted_percentage)    # Output: 12.34%
```

```
value = 0.5678
formatted_percentage = f"{value:.0%}"
print(formatted_percentage)    # Output: 57%
```



Which of the following f-strings correctly formats the variable `percentage = 0.875` as a percentage with no decimal places?

① Start presenting to display the poll results on this slide.

f-string: more examples

f-strings in Python can be created using single quotes `'`, double quotes `"`, or triple quotes `'''` or `"""`. The choice between these depends on your needs, such as whether your string spans multiple lines or contains quotes inside it.

```
name = 'Alice'  
greeting = f'Hello, {name}!'  
print(greeting)
```

```
name = "Bob"  
greeting = f"Hello, {name}!"  
print(greeting)
```

```
name = "Charlie"  
message = f"""  
Hello, {name}.  
Welcome to our service!  
"""  
print(message)
```


f-string: Including Expressions:

You can include any valid Python expression within the curly braces.

```
result = f"The sum of 4 and 5 is {4 + 5}."  
print(result)
```



COMMON MISTAKE



slido



What is the output of the following code?

```
alice = 50  
peter = 70  
sum = alice + peter  
print (sum, end = "#")  
scores = [1,2,3,4]  
total = sum(scores)  
print (total)
```

① Start presenting to display the poll results on this slide.

```
alice = 50
peter = 70
sum = alice + peter
print (sum, end = "#")  120#
scores = [1,2,3,4]
total = sum(scores)
print (total) 
```

```
Traceback (most recent call last):
  File "F:\LF work\LF running courses\CX1103\Lectures\test1.py", line 7, in <module>
    total = sum(scores)
TypeError: 'int' object is not callable
```

		Built-in Functions		
<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

slido



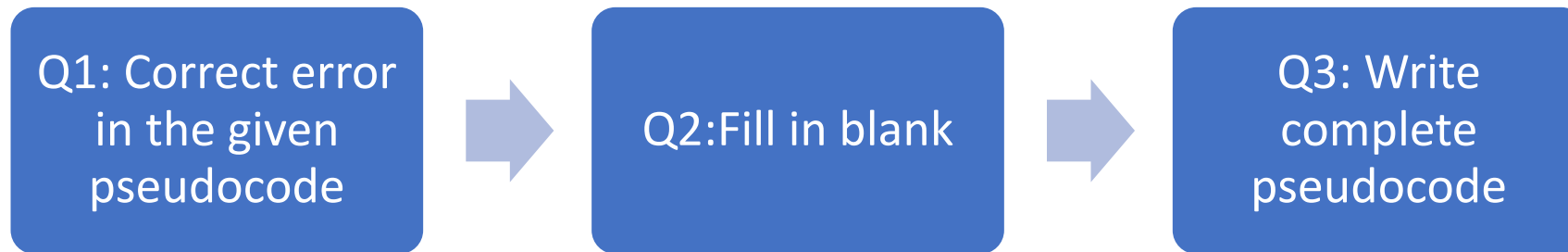
If you get syntax error for the following code which line of code we need to correct?

① Start presenting to display the poll results on this slide.



Discussion briefing

Pseudocode (Q1-Q3)



Discussion 1

The following Pseudocode read in 10 students' scores and calculate the average.

```
Initialize student_counter to zero
```

```
While student_counter is less than or equal to ten
```

```
    Input the next score
```

```
    Add the score into the total
```

```
EndWhile
```

```
Set the class average to the total divided by ten
```

There are some errors in the above Pseudocode. Please indicate where the errors are and how to correct them.

Discussion 2

Fill in the blanks to complete the following Pseudocode to read in 10 students' scores and calculate the number of passes and failures.

Initialize passes to zero

Initialize failures to zero

Initialize student_counter to one

While student_counter is less than or equal to ten

 Input the next score

 add one to student_counter

EndWhile

Discussion 3

Write the FizzBuzz algorithm using pseudocode.

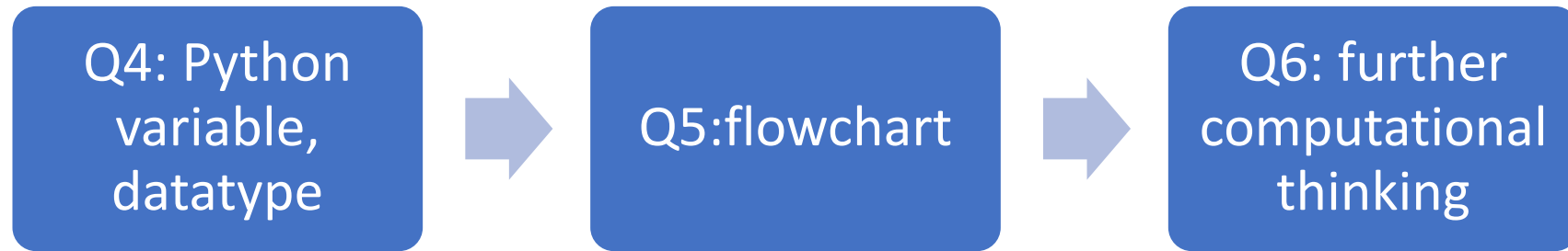
FizzBuzz is a standard interview problem. The Problem state:

- Write a code that prints each number from 1 to 20 on a new line.
- Print "Fizz" if the number is the multiple of 3.
- Print "Buzz" if the number is multiple of 5
- For number which is multiple of both 3 and 5 print "FizzBuzz"

The *sample run* is as follows:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
```

<https://www.techgeekbuzz.com/how-to-write-pseudocode/>



Discussion 4

For each of the following, discuss what the outcome will be if they are executed by a Python interpreter (e.g.IDLE3) in the sequence shown.

```
c = 10
7 = a
a = d
a = c + 1
a + c = c
3 + a
7up = 10
import = 1003
b = math.pi * c

int = 500

a ** 3
a,b,c = c,1,a
b,c,a = a,b
c = b = a = 7
print( A )
print( "b*b + a*a = c*c" )
print( 'A' )
print( "c" = 1 )
```

Discussion 5

Write a program that asks the user for the number of boys and that of girls in a class. The program should calculate and display the percentage of boys and girls in the class. A sample run is as follows:

Enter the number of boys: 65

Enter the number of girls: 77

Boys: 46%

Girls: 54%

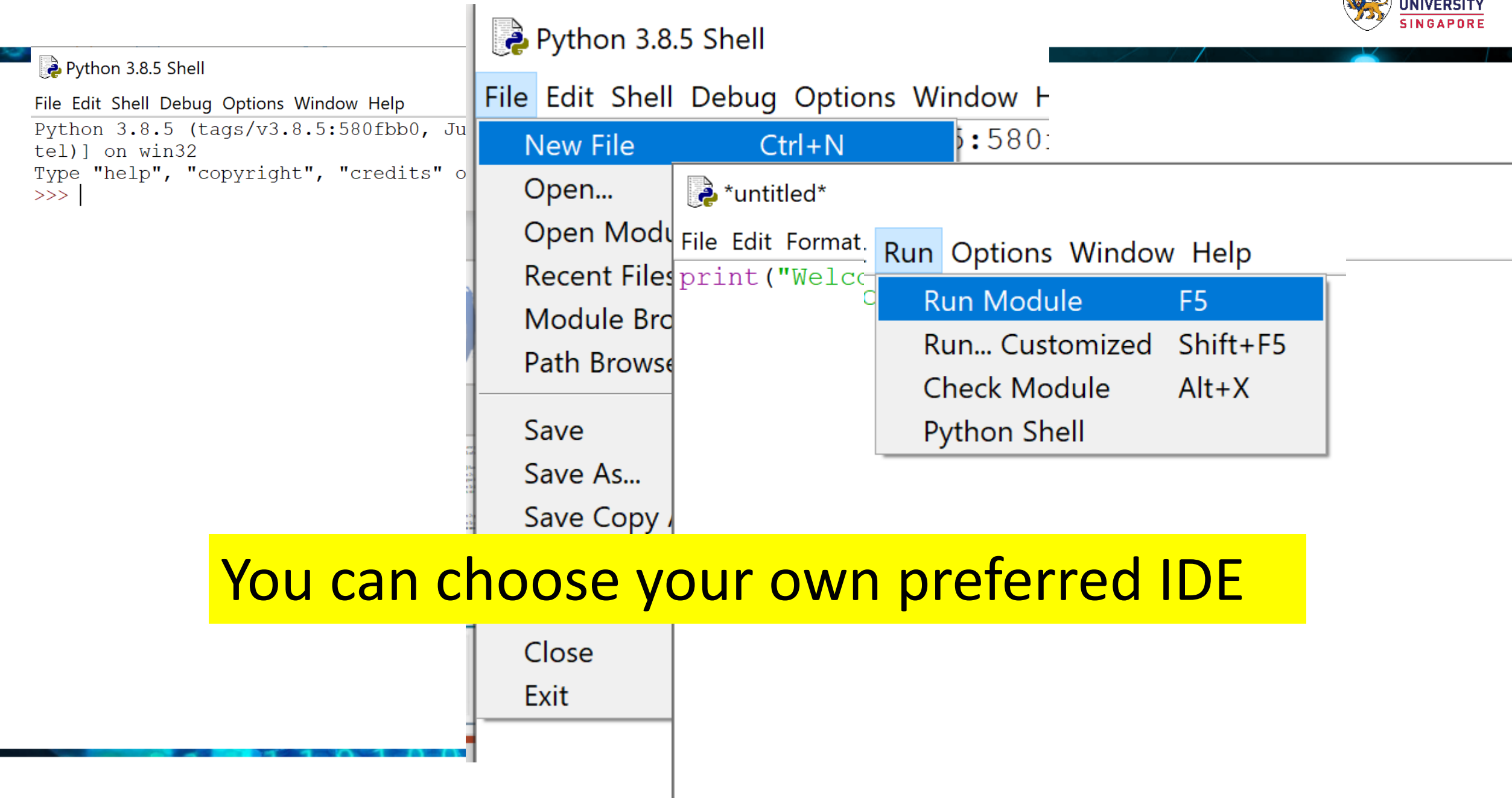
- a) Design the algorithm and use flowchart to present.
- b) Write the Python program. *(optional)*

Imagine you are a travel agency staff member. You have a list of attractions your tourists must visit, and you know roughly how long it takes to travel between each place. You want to plan a route that starts at the hotel, visits each attraction once, and returns to the hotel — while keeping travel time as short as possible.

Propose a solution to the above problem and describe it using pseudocode. You do **NOT** need to find the absolute best route — a good enough route that is logical and easy to follow will do. There is **NO** need to consider data structure details; assume that the distance matrix has already been created.



LAB BRIEFING



Battleship game

Understand
the logic of the
game



Exercise 1: Pseudo Code

Pseudo Code:

- Write pseudo code for initializing the game board.
- Write pseudo code for placing a ship on the board.

Exercise 2: Flowcharts

Flowcharts:

- Create a flowchart for initializing the game board.
- Create a flowchart for the process of placing a ship on the board.

Data Types and Variables:

- Define the data types and variables required for the Battleships game (e.g., board, ships, coordinates).
- Write a short program to declare these variables and print their initial values.

Input and Output Operations:

- Take user input for attack coordinates and display the result.

Exercise : Basic coding-If you are new to Python programming

If you are new to Python programming, copy the following hint code to your IDE, e.g., IDLE, follow the TODO task lists and sample output as follows to complete the exercises.

Follow the sample output
Fill in the blank according to comments