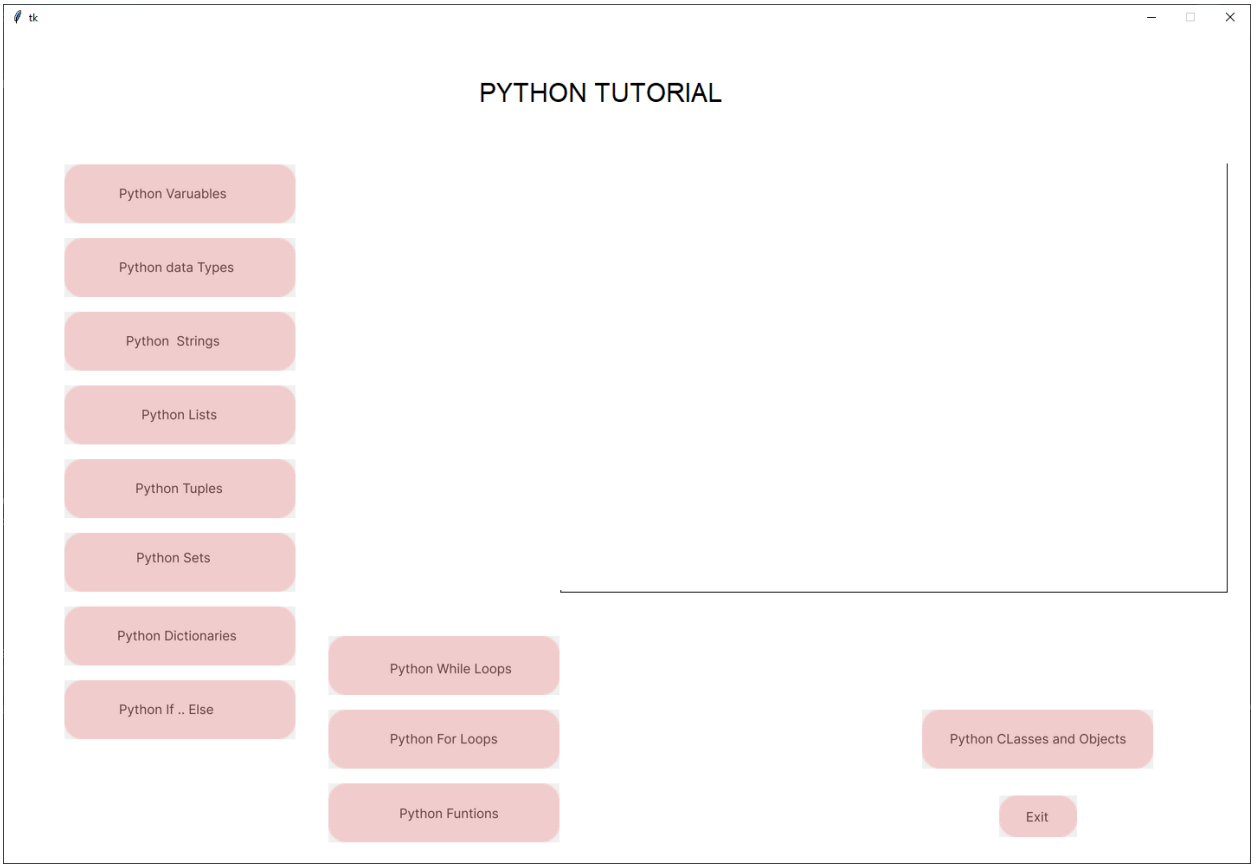
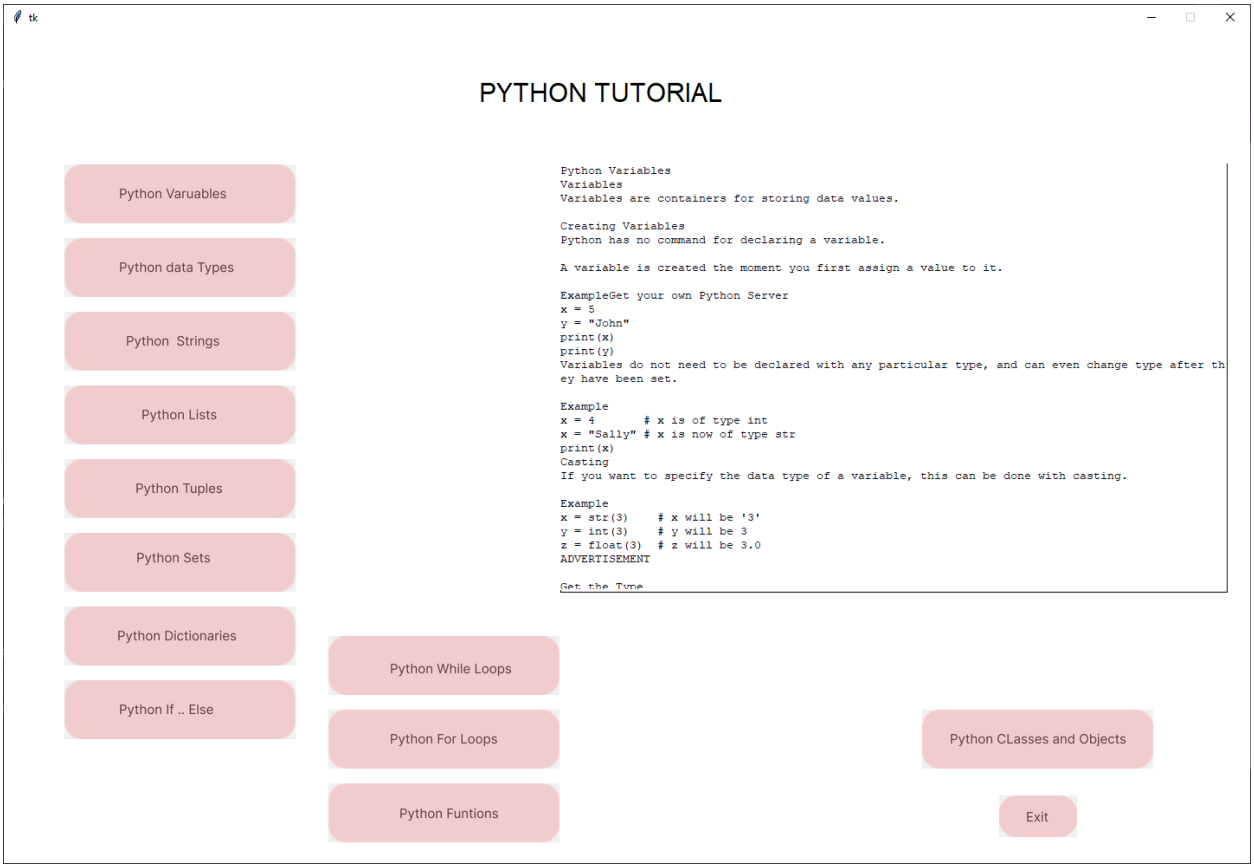
Câu 1:


tk
PYTHON TUTORIAL
Python Varuables
Python data Types
Python Strings
Python Lists
Python Tuples
Python Sets
Python Dictionaries
Python If .. Else
Python While Loops
Python For Loops
Python Funtions
Python CLasses and Objects
Exit



Code:

```
from pathlib import Path
from tkinter import Tk, Canvas, Text, Button, PhotoImage

OUTPUT_PATH = Path(__file__).parent
ASSETS_PATH = OUTPUT_PATH /
Path(r"C:\Users\hello1love\PycharmProjects\pythonProject\CodePython2024\Buoi
17\build\assets\frame0")

def relative_to_assets(path: str) -> Path:
    return ASSETS_PATH / Path(path)

def show_info(content):
    entry_1.delete("1.0", "end")  # Xóa nội dung hiện tại của Text widget
    entry_1.insert("1.0", content)  # Chèn nội dung mới vào Text widget

window = Tk()
window.geometry("1440x960")
window.configure(bg = "#FFFFFF")

canvas = Canvas(
    window,
    bg = "#FFFFFF",
    height = 960,
    width = 1440,
    bd = 0,
```

```
    highlightthickness = 0,
    relief = "ridge"
)
canvas.place(x = 0, y = 0)
canvas.create_text(
    549.0,
    54.0,
    anchor="nw",
    text="PYTHON TUTORIAL",
    fill="#000000",
    font=("Inter", 30 * -1)
)
topics = {
    "button_1": """Python Variables
Variables
Variables are containers for storing data values.

Creating Variables
Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

ExampleGet your own Python Server
x = 5
y = "John"
print(x)
print(y)
Variables do not need to be declared with any particular type, and can even
change type after they have been set.

Example
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
Casting
If you want to specify the data type of a variable, this can be done with
casting.

Example
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
ADVERTISEMENT

Get the Type
You can get the data type of a variable with the type() function.

Example
x = 5
y = "John"
print(type(x))
print(type(y))
You will learn more about data types and casting later in this tutorial.
Single or Double Quotes?
String variables can be declared either by using single or double quotes:

Example
```

```
x = "John"
# is the same as
x = 'John'
Case-Sensitive
Variable names are case-sensitive.

Example
This will create two variables:

a = 4
A = "Sally"
#A will not overwrite a""",
    "button_2": "Exit",
    "button_3": "Information about Python Classes and Objects",
    "button_4": """Python While Loops
Python Loops
Python has two primitive loop commands:

while loops
for loops
The while Loop
With the while loop we can execute a set of statements as long as a condition
is true.

ExampleGet your own Python Server
Print i as long as i is less than 6:

i = 1
while i < 6:
  print(i)
  i += 1
Note: remember to increment i, or else the loop will continue forever.

The while loop requires relevant variables to be ready, in this example we
need to define an indexing variable, i, which we set to 1.

The break Statement
With the break statement we can stop the loop even if the while condition is
true:

Example
Exit the loop when i is 3:

i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
ADVERTISEMENT

The continue Statement
With the continue statement we can stop the current iteration, and continue
with the next:

Example
Continue to the next iteration if i is 3:
```

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
The else Statement
With the else statement we can run a block of code once when the condition no
longer is true:

Example
Print a message once the condition is false:

i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
Test Yourself With Exercises
Exercise:
Print i as long as i is less than 6.

i = 1
 i < 6

  print(i)
  i += 1

Start the Exercise""",
    "button_5": "Information about Python For Loops",
    "button_6": "Information about Python While Loops",
    "button_7": "Information about Python If ... Else",
    "button_8": """Python Lists
    Lists are used to store multiple items in a single variable.

    Lists are one of 4 built-in data types in Python used to store
collections of data, the other 3 are Tuple, Set, and Dictionary, all with
different qualities and usage.

    Lists are created using square brackets:

    Example:
    thislist = ["apple", "banana", "cherry"]
    print(thislist)

    List Items:
    List items are ordered, changeable, and allow duplicate values.

    List items are indexed, the first item has index [0], the second item has
index [1], etc.

    Ordered:
    When we say that lists are ordered, it means that the items have a
defined order, and that order will not change.
```

```
    If you add new items to a list, the new items will be placed at the end
of the list.

    Note: There are some list methods that will change the order, but in
general, the order of the items will not change.

    Changeable:
    The list is changeable, meaning that we can change, add, and remove items
in a list after it has been created.

    Allow Duplicates:
    Since lists are indexed, lists can have items with the same value.

    Example:
    Lists allow duplicate values:

    thislist = ["apple", "banana", "cherry", "apple", "cherry"]
    print(thislist)

    List Length:
    To determine how many items a list has, use the len() function.

    Example:
    Print the number of items in the list:

    thislist = ["apple", "banana", "cherry"]
    print(len(thislist))

    List Items - Data Types:
    List items can be of any data type.

    Example:
    String, int, and boolean data types:

    list1 = ["apple", "banana", "cherry"]
    list2 = [1, 5, 7, 9, 3]
    list3 = [True, False, False]

    A list can contain different data types:

    Example:
    A list with strings, integers, and boolean values:

    list1 = ["abc", 34, True, 40, "male"]

    type():
    From Python's perspective, lists are defined as objects with the data
type 'list'.

    Example:
    What is the data type of a list?

    mylist = ["apple", "banana", "cherry"]
    print(type(mylist))

    The list() Constructor:
    It is also possible to use the list() constructor when creating a new
```

```
list.

    Example:
    Using the list() constructor to make a List:

    thislist = list(("apple", "banana", "cherry")) # note the double round-
brackets
    print(thislist)

    Python Collections (Arrays):
    There are four collection data types in the Python programming language:

    - List is a collection which is ordered and changeable. Allows duplicate
members.
    - Tuple is a collection which is ordered and unchangeable. Allows
duplicate members.
    - Set is a collection which is unordered, unchangeable, and unindexed. No
duplicate members.
    - Dictionary is a collection which is ordered and changeable. No
duplicate members.

    When choosing a collection type, it is useful to understand the
properties of that type. Choosing the right type for a particular data set
could mean retention of meaning, and, it could mean an increase in efficiency
or security."""
    ,
    "button_9": """""",
    "button_10": """Python Data Types
Built-in Data Types
In programming, data type is an important concept.

Variables can store data of different types, and different types can do
different things.

Python has the following data types built-in by default, in these categories:

Text Type:  str
Numeric Types:  int, float, complex
Sequence Types: list, tuple, range
Mapping Type:   dict
Set Types:  set, frozenset
Boolean Type:   bool
Binary Types:   bytes, bytearray, memoryview
None Type:  NoneType
Getting the Data Type
You can get the data type of any object by using the type() function:

ExampleGet your own Python Server
Print the data type of the variable x:

x = 5
print(type(x))
Setting the Data Type
In Python, the data type is set when you assign a value to a variable:

Example Data Type  Try it
x = "Hello World"   str
```

```
x = 20  int
x = 20.5    float
x = 1j  complex
x = ["apple", "banana", "cherry"]   list
x = ("apple", "banana", "cherry")   tuple
x = range(6)    range
x = {"name" : "John", "age" : 36}   dict
x = {"apple", "banana", "cherry"}   set
x = frozenset({"apple", "banana", "cherry"})    frozenset
x = True    bool
x = b"Hello"    bytes
x = bytearray(5)    bytearray
x = memoryview(bytes(5))    memoryview
x = None    NoneType
ADVERTISEMENT

Setting the Specific Data Type
If you want to specify the data type, you can use the following constructor
functions:

Example Data Type  Try it
x = str("Hello World")  str
x = int(20) int
x = float(20.5) float
x = complex(1j) complex
x = list(("apple", "banana", "cherry")) list
x = tuple(("apple", "banana", "cherry"))    tuple
x = range(6)    range
x = dict(name="John", age=36)   dict
x = set(("apple", "banana", "cherry"))  set
x = frozenset(("apple", "banana", "cherry"))    frozenset
x = bool(5) bool
x = bytes(5)    bytes
x = bytearray(5)    bytearray
x = memoryview(bytes(5))    memoryview
Test Yourself With Exercises
Exercise:
The following code example would print the data type of x, what data type
would that be?

x = 5
print(type(x))


Start the Exercise""",
    "button_11": """Python Tuples
mytuple = ("apple", "banana", "cherry")
Tuple
Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of
data, the other 3 are List, Set, and Dictionary, all with different qualities
and usage.

A tuple is a collection which is ordered and unchangeable.

Tuples are written with round brackets.
```

ExampleGet your own Python Server
Create a Tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Tuple Items
Tuple items are ordered, unchangeable, and allow duplicate values.

Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered
When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.

Unchangeable
Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Allow Duplicates
Since tuples are indexed, they can have items with the same value:

Example
Tuples allow duplicate values:

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```


Tuple Length
To determine how many items a tuple has, use the len() function:

Example
Print the number of items in the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Create Tuple With One Item
To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.

Example
One item tuple, remember the comma:

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Tuple Items - Data Types
Tuple items can be of any data type:

Example
String, int and boolean data types:

```
tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)
A tuple can contain different data types:

Example
A tuple with strings, integers and boolean values:

tuple1 = ("abc", 34, True, 40, "male")
type()
From Python's perspective, tuples are defined as objects with the data type
'tuple':

<class 'tuple'>
Example
What is the data type of a tuple?

mytuple = ("apple", "banana", "cherry")
print(type(mytuple))
The tuple() Constructor
It is also possible to use the tuple() constructor to make a tuple.

Example
Using the tuple() method to make a tuple:

thistuple = tuple(("apple", "banana", "cherry")) # note the double round-
brackets
print(thistuple)
Python Collections (Arrays)
There are four collection data types in the Python programming language:

List is a collection which is ordered and changeable. Allows duplicate
members.
Tuple is a collection which is ordered and unchangeable. Allows duplicate
members.
Set is a collection which is unordered, unchangeable*, and unindexed. No
duplicate members.
Dictionary is a collection which is ordered** and changeable. No duplicate
members.
*Set items are unchangeable, but you can remove and/or add items whenever you
like.

**As of Python version 3.7, dictionaries are ordered. In Python 3.6 and
earlier, dictionaries are unordered.

When choosing a collection type, it is useful to understand the properties of
that type. Choosing the right type for a particular data set could mean
retention of meaning, and, it could mean an increase in efficiency or
security.""",


    "button_12": """Python If ... Else
Python Conditions and If statements
Python supports the usual logical conditions from mathematics:

Equals: a == b
Not Equals: a != b
```

Less than: a < b
Less than or equal to: a <= b
Greater than: a > b
Greater than or equal to: a >= b
These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

ExampleGet your own Python Server
If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

Indentation
Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

Example
If statement, without indentation (will raise an error):

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```


Elif
The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

Example

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

Else
The else keyword catches anything which isn't caught by the preceding conditions.

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
```

```
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```
In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

Example
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```
Short Hand If
If you have only one statement to execute, you can put it on the same line as the if statement.

Example
One line if statement:

```
if a > b: print("a is greater than b")
```
Short Hand If ... Else
If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

Example
One line if else statement:

```
a = 2
b = 330
print("A") if a > b else print("B")
```
This technique is known as Ternary Operators, or Conditional Expressions.

You can also have multiple else statements on the same line:

Example
One line if else statement, with 3 conditions:

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```
And
The and keyword is a logical operator, and is used to combine conditional statements:

Example
Test if a is greater than b, AND if c is greater than a:

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

Or
The or keyword is a logical operator, and is used to combine conditional statements:

Example
Test if a is greater than b, OR if a is greater than c:

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Not
The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

Example
Test if a is NOT greater than b:

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

Nested If
You can have if statements inside if statements, this is called nested if statements.

Example

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

The pass Statement
if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

Example

```
a = 33
b = 200

if b > a:
  pass
```

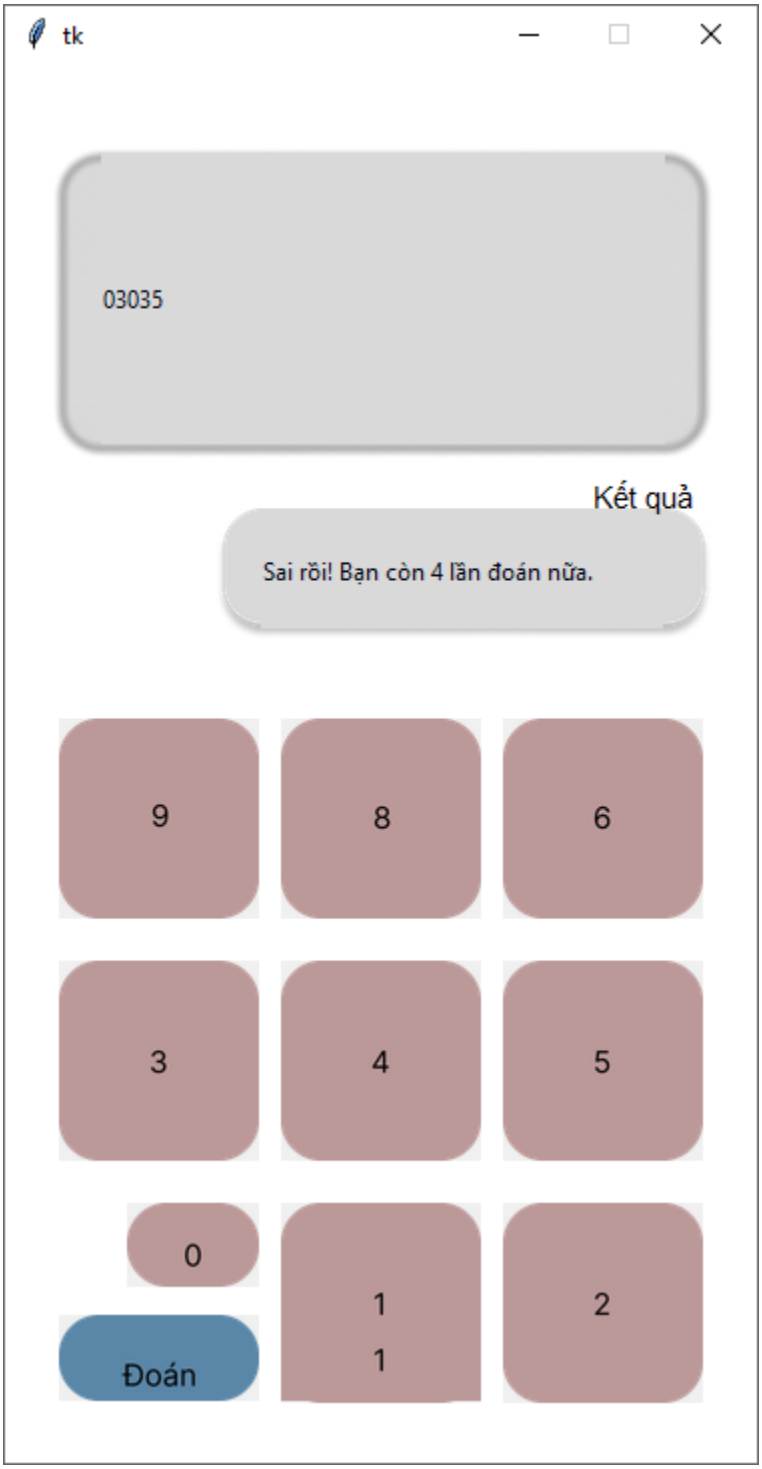
Test Yourself With Exercises
Exercise:
Print "Hello World" if a is greater than b.

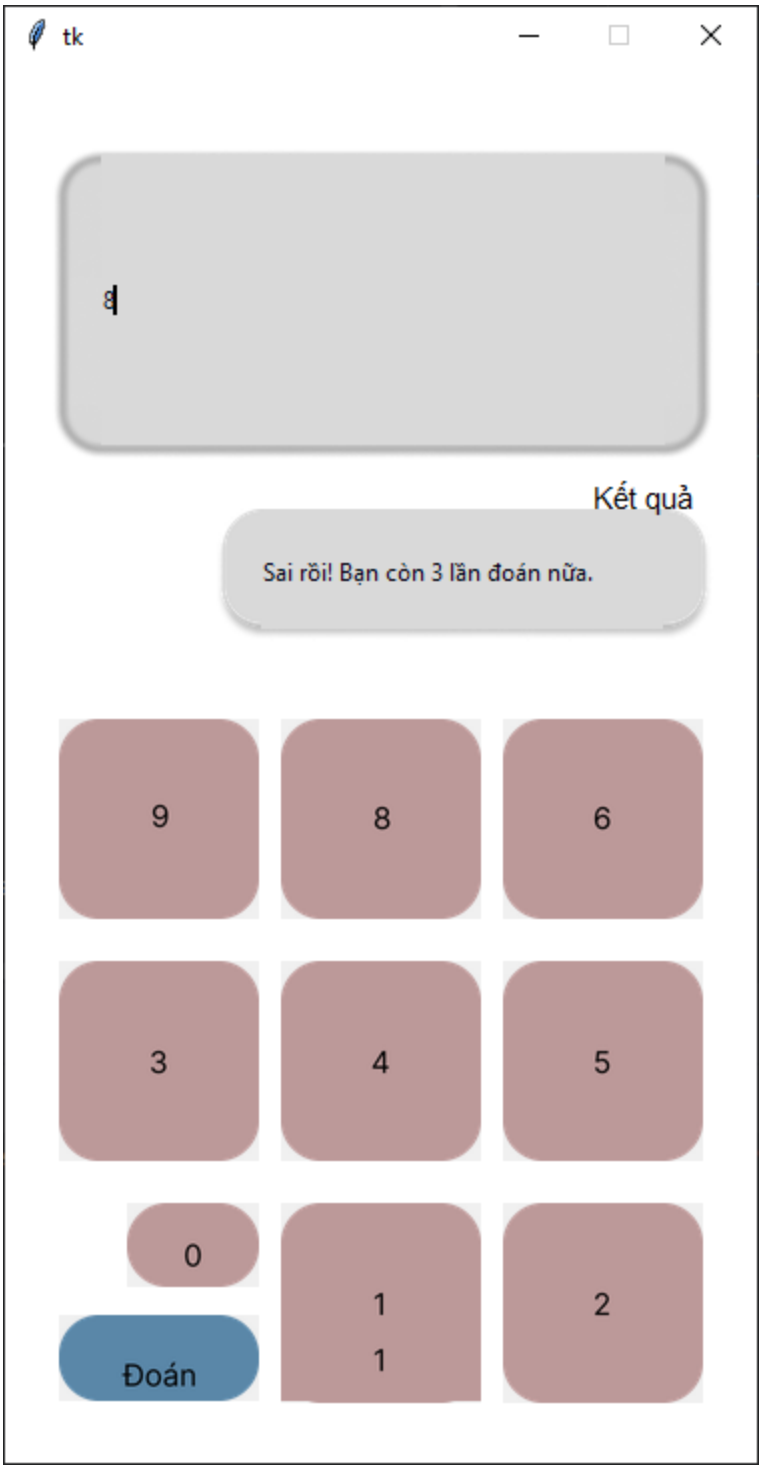
```
a = 50
b = 10
 a
 b

  print("Hello World")
```

```
Start the Exercise""",
    "button_13": "Information about Python Data Types"
}


def button_click(button_id):
    content = topics[button_id]
    if content == "Exit":
        window.quit()
    else:
        show_info(content)

button_image_1 = PhotoImage(file=relative_to_assets("button_1.png"))
button_1 = Button(image=button_image_1, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_1"), relief="flat")
button_1.place(x=70.0, y=154.0, width=267.0, height=68.0)

button_image_2 = PhotoImage(file=relative_to_assets("button_2.png"))
button_2 = Button(image=button_image_2, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_2"), relief="flat")
button_2.place(x=1150.0, y=882.0, width=90.0, height=48.0)

button_image_3 = PhotoImage(file=relative_to_assets("button_3.png"))
button_3 = Button(image=button_image_3, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_3"), relief="flat")
button_3.place(x=1061.0, y=783.0, width=267.0, height=68.0)

button_image_4 = PhotoImage(file=relative_to_assets("button_4.png"))
button_4 = Button(image=button_image_4, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_4"), relief="flat")
button_4.place(x=375.0, y=698.0, width=267.0, height=68.0)

button_image_5 = PhotoImage(file=relative_to_assets("button_5.png"))
button_5 = Button(image=button_image_5, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_5"), relief="flat")
button_5.place(x=375.0, y=868.0, width=267.0, height=68.0)

button_image_6 = PhotoImage(file=relative_to_assets("button_6.png"))
button_6 = Button(image=button_image_6, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_6"), relief="flat")
button_6.place(x=70.0, y=664.0, width=267.0, height=68.0)

button_image_7 = PhotoImage(file=relative_to_assets("button_7.png"))
button_7 = Button(image=button_image_7, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_7"), relief="flat")
button_7.place(x=70.0, y=579.0, width=267.0, height=68.0)

button_image_8 = PhotoImage(file=relative_to_assets("button_8.png"))
button_8 = Button(image=button_image_8, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_8"), relief="flat")
button_8.place(x=70.0, y=409.0, width=267.0, height=68.0)

button_image_9 = PhotoImage(file=relative_to_assets("button_9.png"))
button_9 = Button(image=button_image_9, borderwidth=0, highlightthickness=0,
command=lambda: button_click("button_9"), relief="flat")
button_9.place(x=70.0, y=324.0, width=267.0, height=68.0)
```

```
button_image_10 = PhotoImage(file=relative_to_assets("button_10.png"))
button_10 = Button(image=button_image_10, borderwidth=0,
highlightthickness=0, command=lambda: button_click("button_10"),
relief="flat")
button_10.place(x=70.0, y=239.0, width=267.0, height=68.0)

button_image_11 = PhotoImage(file=relative_to_assets("button_11.png"))
button_11 = Button(image=button_image_11, borderwidth=0,
highlightthickness=0, command=lambda: button_click("button_11"),
relief="flat")
button_11.place(x=70.0, y=494.0, width=267.0, height=68.0)

button_image_12 = PhotoImage(file=relative_to_assets("button_12.png"))
button_12 = Button(image=button_image_12, borderwidth=0,
highlightthickness=0, command=lambda: button_click("button_12"),
relief="flat")
button_12.place(x=70.0, y=749.0, width=267.0, height=68.0)

button_image_13 = PhotoImage(file=relative_to_assets("button_13.png"))
button_13 = Button(image=button_image_13, borderwidth=0,
highlightthickness=0, command=lambda: button_click("button_13"),
relief="flat")
button_13.place(x=375.0, y=783.0, width=267.0, height=68.0)

entry_image_1 = PhotoImage(file=relative_to_assets("entry_1.png"))
entry_bg_1 = canvas.create_image(1027.5, 399.5, image=entry_image_1)
entry_1 = Text(bd=0, bg="#FFFFFF", fg="#000716", highlightthickness=0)
entry_1.place(x=642.0, y=152.0, width=771.0, height=493.0)

window.resizable(False, False)
window.mainloop()
```

câu 2:


tk
03035
Kết quả
Sai rồi! Bạn còn 4 lần đoán nữa.
9
8
6
3
4
5
0
1
1
2
Đoán



Code:

```
from pathlib import Path
import random

from tkinter import Tk, Canvas, Entry, Text, Button, PhotoImage

OUTPUT_PATH = Path(__file__).parent
```

```
ASSETS_PATH = OUTPUT_PATH / Path(r"D:\pyth0n\Python Designer\Tkinter-
Designer\build\assets\frame0")

def relative_to_assets(path: str) -> Path:
    return ASSETS_PATH / Path(path)

secret_number = random.randint(1, 100)
attempts = 5

def check_guess():
    global attempts
    guess = int(entry_1.get())
    if guess == secret_number:
        entry_2.delete(0, 'end')
        entry_2.insert(0, "Chúc mừng! Bạn đã đoán đúng số.")
        reset_game()
    else:
        attempts -= 1
        if attempts > 0:
            entry_2.delete(0, 'end')
            entry_2.insert(0, f"Sai rồi! Bạn còn {attempts} lần đoán nữa.")
        else:
            entry_2.delete(0, 'end')
            entry_2.insert(0, f"Bạn đã hết lượt đoán. Số bí mật là
{secret_number}.")
            reset_game()

def reset_game():
    global secret_number, attempts
    secret_number = random.randint(1, 100)
    attempts = 5
    entry_1.delete(0, 'end')
    entry_2.delete(0, 'end')
    entry_2.insert(0, "Hãy đoán một số từ 1 đến 100")

def append_digit(digit):
    current_value = entry_1.get()
    entry_1.delete(0, 'end')
    entry_1.insert(0, current_value + str(digit))

window = Tk()

window.geometry("376x698")
window.configure(bg="#FFFFFF")

canvas = Canvas(
    window,
    bg="#FFFFFF",
    height=698,
    width=376,
    bd=0,
    highlightthickness=0,
    relief="ridge"
)

canvas.place(x=0, y=0)
button_image_1 = PhotoImage(
```

```
    file=relative_to_assets("button_1.png"))
button_1 = Button(
    image=button_image_1,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(1),
    relief="flat"
)
button_1.place(
    x=249.0,
    y=447.0,
    width=100.0,
    height=100.0
)

button_image_2 = PhotoImage(
    file=relative_to_assets("button_2.png"))
button_2 = Button(
    image=button_image_2,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(2),
    relief="flat"
)
button_2.place(
    x=249.0,
    y=568.0,
    width=100.0,
    height=100.0
)

button_image_3 = PhotoImage(
    file=relative_to_assets("button_3.png"))
button_3 = Button(
    image=button_image_3,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(3),
    relief="flat"
)
button_3.place(
    x=138.0,
    y=568.0,
    width=100.0,
    height=100.0
)

button_image_4 = PhotoImage(
    file=relative_to_assets("button_4.png"))
button_4 = Button(
    image=button_image_4,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(4),
    relief="flat"
)
button_4.place(
```

```
    x=27.0,
    y=326.0,
    width=100.0,
    height=100.0
)

button_image_5 = PhotoImage(
    file=relative_to_assets("button_5.png"))
button_5 = Button(
    image=button_image_5,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(5),
    relief="flat"
)
button_5.place(
    x=138.0,
    y=447.0,
    width=100.0,
    height=100.0
)

button_image_6 = PhotoImage(
    file=relative_to_assets("button_6.png"))
button_6 = Button(
    image=button_image_6,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(6),
    relief="flat"
)
button_6.place(
    x=61.0,
    y=568.0,
    width=66.0,
    height=42.0
)

button_image_7 = PhotoImage(
    file=relative_to_assets("button_7.png"))
button_7 = Button(
    image=button_image_7,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(7),
    relief="flat"
)
button_7.place(
    x=27.0,
    y=447.0,
    width=100.0,
    height=100.0
)

button_image_8 = PhotoImage(
    file=relative_to_assets("button_8.png"))
button_8 = Button(
```

```
    image=button_image_8,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(8),
    relief="flat"
)
button_8.place(
    x=249.0,
    y=326.0,
    width=100.0,
    height=100.0
)

button_image_9 = PhotoImage(
    file=relative_to_assets("button_9.png"))
button_9 = Button(
    image=button_image_9,
    borderwidth=0,
    highlightthickness=0,
    command=lambda: append_digit(9),
    relief="flat"
)
button_9.place(
    x=138.0,
    y=326.0,
    width=100.0,
    height=100.0
)

entry_image_1 = PhotoImage(
    file=relative_to_assets("entry_1.png"))
entry_bg_1 = canvas.create_image(
    189.0,
    117.5,
    image=entry_image_1
)
entry_1 = Entry(
    bd=0,
    bg="#D9D9D9",
    fg="#000716",
    highlightthickness=0
)
entry_1.place(
    x=48.0,
    y=44.0,
    width=282.0,
    height=145.0
)

entry_image_2 = PhotoImage(
    file=relative_to_assets("entry_2.png"))
entry_bg_2 = canvas.create_image(
    228.5,
    254.0,
    image=entry_image_2
)
entry_2 = Entry(
```

```
    bd=0,
    bg="#D9D9D9",
    fg="#000716",
    highlightthickness=0
)
entry_2.place(
    x=128.0,
    y=225.0,
    width=201.0,
    height=56.0
)

canvas.create_text(
    294.0,
    207.0,
    anchor="nw",
    text="Kết quả",
    fill="#000000",
    font=("Inter", 15 * -1)
)
button_image_10 = PhotoImage(
file=relative_to_assets("button_10.png"))
button_10 = Button(
image=button_image_10,
borderwidth=0,
highlightthickness=0,
command=check_guess,
relief="flat"
)
button_10.place(
x=27.0,
y=624.0,
width=100.0,
height=43.0
)
button_image_0 = PhotoImage(
file=relative_to_assets("button_3.png"))
button_0 = Button(
image=button_image_0,
borderwidth=0,
highlightthickness=0,
command=lambda: append_digit(0),
relief="flat"
)
button_0.place(
x=138.0,
y=624.0,
width=100.0,
height=43.0
)
window.resizable(False, False)
window.mainloop()
```