HKUST Future-Ready Scholars
Introduction to Game Programming using Python

Part 1: Number Guessing Game

20 April 2024

THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

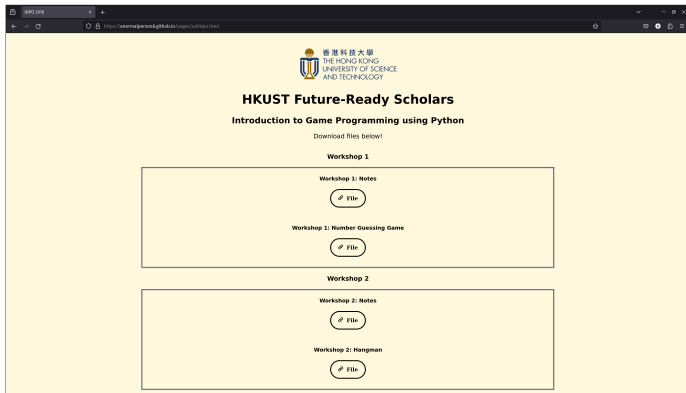Open a tab on your browser, then go to
https://www.menti.com/

Number Guessing Game

Let's play the number guessing game.

Set-up your Gmail account.

Then head to
https://colab.research.google.com/

All materials today are at:
https://bit.ly/ustidpo

Download all files that belong to **Workshop 1** today.

Jupyter Notebook

Now upload your Jupyter Notebook file with **Files** $\rightarrow$ **Open Notebook**.

Upload the file **Number**-**Guessing.ipynb**.

Using Jupyter Notebook

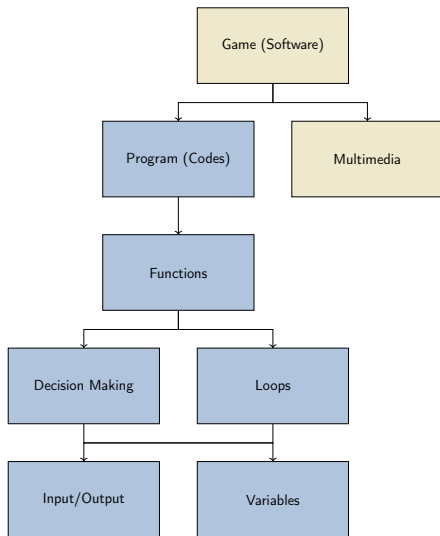You can type your code in these blocks. We call these blocks code cells.

```
print("Mum I am in HKUST typing code in a code block")
```

You can run a code cell with the button on the left.

```
print("Hello World!") # Prints "Hello World!"
```
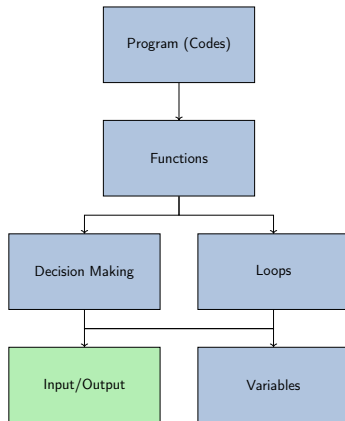
World of Game Coding

What is Python?

Did you know? Python was made by someone who was bored.
It's a language designed to be almost as understandable as English.
You will be using Python 3. Why? Because Python 1 are 2 are too old.

This is the logo of Python.

Contents

```
print("This is the print function.")
```

The first thing in Python - `print()` function

`print()` is a function that lets you print something, also known as text output.

```python
print("Word") # This prints the word "Word".
```

Examples:

```python
>>> print("Hello World")
Hello World
>>> print("Haha hehe")
Haha hehe
```

Printing multiple things

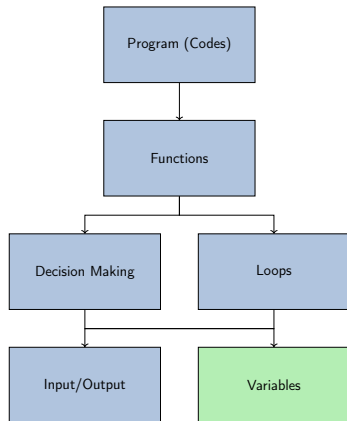You can use a comma (,) to separate different things with a space.

```
>>> print("Alpha", "Beta", "Gamma")
Alpha Beta Gamma
>>> print("Haha", "hehe")
Haha hehe
```

Contents

Imagine you borrow a box from the computer.

Imagine you borrow a box from the computer.

Give it a name and a value, you can now recall this value with the name!

Variables

The code usually goes:

`variable_name = data`

This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

`a = 5`

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5        # This is an integer (int) stored in a
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name
`variable_name`.

Some basic variable types:

```
a = 5          # This is an integer (int) stored in a
b = True
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5       # This is an integer (int) stored in a
b = True    # This is a boolean (bool) stored in b
```

Variables

The code usually goes:

variable_name = data

This means whatever data is, it is now stored in a variable with name variable_name.

Some basic variable types:

```
a = 5        # This is an integer (int) stored in a
b = True     # This is a boolean (bool) stored in b
c = 3.2
```

Variables

The code usually goes:

`variable_name = data`

This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5         # This is an integer (int) stored in a
b = True      # This is a boolean (bool) stored in b
c = 3.2       # This is a float (float)  stored in c
```

Variables

The code usually goes:
variable_name = data
This means whatever data is, it is now stored in a variable with name
variable_name.

Some basic variable types:

```python
a = 5          # This is an integer (int) stored in a
b = True       # This is a boolean (bool) stored in b
c = 3.2        # This is a float (float)  stored in c
d = "abc"
```

Variables

The code usually goes:

`variable_name = data`

This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5        # This is an integer (int) stored in a
b = True     # This is a boolean (bool) stored in b
c = 3.2      # This is a float (float)  stored in c
d = "abc"    # This is a string (str)   stored in d
```

Variables

The code usually goes:

`variable_name = data`

This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5          # This is an integer (int) stored in a
b = True       # This is a boolean (bool) stored in b
c = 3.2        # This is a float (float)  stored in c
d = "abc"      # This is a string (str)   stored in d
e = 'abc'
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name
`variable_name`.

Some basic variable types:

```
a = 5        # This is an integer (int) stored in a
b = True     # This is a boolean (bool) stored in b
c = 3.2      # This is a float (float)   stored in c
d = "abc"    # This is a string (str)    stored in d
e = 'abc'    # This is also a string     stored in e
```

What are integers?

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

`a = 5`

Variables - Integers

What are integers?

Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```
a = 5          # Valid
```

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```
a = 5          # Valid
b = 12
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```
a = 5        # Valid
b = 12       # Valid
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```
a = 5          # Valid
b = 12         # Valid
c = 69420
```

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```python
a = 5        # Valid
b = 12       # Valid
c = 69420    # Valid
```

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```
a = 5          # Valid
b = 12         # Valid
c = 69420      # Valid
d = -1984
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```
a = 5        # Valid
b = 12       # Valid
c = 69420    # Valid
d = -1984    # Valid
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```python
a = 5         # Valid
b = 12        # Valid
c = 69420     # Valid
d = -1984     # Valid
e = 32.5
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```
a = 5        # Valid
b = 12       # Valid
c = 69420    # Valid
d = -1984    # Valid
e = 32.5     # This would become a float instead
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```python
a = 5        # Valid
b = 12       # Valid
c = 69420    # Valid
d = -1984    # Valid
e = 32.5     # This would become a float instead
f = '5'
```

Variables - Integers

What are integers?

Integers are just like what you've learnt in Maths, numbers without decimal points. Are the following valid?

```python
a = 5          # Valid
b = 12         # Valid
c = 69420      # Valid
d = -1984      # Valid
e = 32.5       # This would become a float instead
f = '5'        # This would become a string instead
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2
```

You can do normal operations on integers:

```python
a = 1 + 2    # a stores the integer 3
```

You can do normal operations on integers:

```python
a = 1 + 2    # a stores the integer 3
b = 80 - 52
```

You can do normal operations on integers:

```python
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52  # b stores the integer 28
c = 69 * -2
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
```

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
d = 6 / 4
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52  # b stores the integer 28
c = 69 * -2  # c stores the integer -138
d = 6 / 4    # d stores the float 1.5
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52  # b stores the integer 28
c = 69 * -2  # c stores the integer -138
d = 6 / 4    # d stores the float 1.5
e = 18 / 2
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52  # b stores the integer 28
c = 69 * -2  # c stores the integer -138
d = 6 / 4    # d stores the float 1.5
e = 18 / 2   # e stores the float 9.0
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```python
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
d = 6 / 4   # d stores the float 1.5
e = 18 / 2  # e stores the float 9.0
```

Division in Python

Whether a number can be precisely divided or not, division returns a `float`.

Operations with variables:

```
a = 100
b = 12
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b
```

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
```

Variables - Integer Arithmetic Operations

Operations with variables:

```python
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a
```

Operations with variables:

```
a = 100
b = 12
c = a + b     # c stores the integer 112
d = b - a     # d stores the integer -88
```

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a    # d stores the integer -88
e = a * -b
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b   # c stores the integer 112
d = b - a   # d stores the integer -88
e = a * -b  # e stores the integer -1200
```

Operations with variables:

```
a = 100
b = 12
c = a + b   # c stores the integer 112
d = b - a   # d stores the integer -88
e = a * -b  # e stores the integer -1200
f = a / b
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b   # c stores the integer 112
d = b - a   # d stores the integer -88
e = a * -b  # e stores the integer -1200
f = a / b   # f stores the float 8.333333333333334
```

Then how do we get an integer output?

Then how do we get an integer output?

```
a = 100
b = 12
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b
```

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b   # c stores the integer 8
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b   # c stores the integer 8
             # // operator takes the closest and smaller
             # integer from the division operation
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
            # // operator takes the closest and smaller
            # integer from the division operation
d = a % b
```

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
            # // operator takes the closest and smaller
            # integer from the division operation
d = a % b   # d stores the integer 4
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
            # // operator takes the closest and smaller
            # integer from the division operation
d = a % b   # d stores the integer 4
            # % operator takes the remainder of a
            # division operation
```

Also, the power (exponent) operation:

```
a = 2
b = 5
```

Also, the power (exponent) operation:

```
a = 2
b = 5
c = a ** b
```

Variables - Integer Arithmetic Operations

Also, the power (exponent) operation:

```
a = 2
b = 5
c = a ** b  # c stores the integer 32
            # ** operator means power
```

What are floats?

Variables - Floats

What are floats?
Floats are numbers with decimal points.

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```python
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
```

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```python
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b
```

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```python
a = 0.2       # a stores the float 0.2
b = 3.0       # b stores the float 3.0
c = a + b     # c stores the float 3.2
```

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```python
a = 0.2        # a stores the float 0.2
b = 3.0        # b stores the float 3.0
c = a + b      # c stores the float 3.2
d = b / a
```

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```
a = 0.2       # a stores the float 0.2
b = 3.0       # b stores the float 3.0
c = a + b     # c stores the float 3.2
d = b / a     # d stores the float 15.0
```

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b
```

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b   # e stores the float 0.008000000000000002
```

Variables - Floats

What are floats?

Floats are numbers with decimal points.

Arithmetic operators we learnt can be applied as well.

```python
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b   # e stores the float 0.008000000000000002
```

Inaccuracies

Inaccuracies happen with decimals in Python. Be careful when dealing with floats.

What happens when you combine floats and integers?

What happens when you combine floats and integers?

```python
a = 0.2        # a stores the float 0.2
b = 3          # b stores the integer 3
```

Variables - Floats

What happens when you combine floats and integers?

```
a = 0.2       # a stores the float 0.2
b = 3         # b stores the integer 3
c = a + b     # c stores the float 3.2
d = b / a     # d stores the float 15.0
e = a ** b    # e stores the float 0.008000000000000002
```

Variables - Floats

What happens when you combine floats and integers?

```python
a = 0.2      # a stores the float 0.2
b = 3        # b stores the integer 3
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b   # e stores the float 0.008000000000000002
```

Arithmetic operations between `int` and `float`

Arithmetic operations between integers and floats converts the integer into a float first before operating.

Variables - Boolean values

What are boolean values?

Variables - Boolean values

What are boolean values?
There are only 2 boolean values in existence: `True` and `False`.

Variables - Boolean values

What are boolean values?
There are only 2 boolean values in existence: `True` and `False`.

```
a = True
b = False
```

Variables - Boolean values

What are boolean values?

There are only 2 boolean values in existence: `True` and `False`.

```
a = True
b = False
```

That's it.

Variables - Boolean values

What are boolean values?
There are only 2 boolean values in existence: `True` and `False`.

```
a = True
b = False
```

Just kidding, we will elaborate more on boolean values later.

Variables - Strings

What are strings?

What are strings?

```
a = "word"
```

What are strings?

```python
a = "word"  # a stores the string "word"
```

Variables - Strings

What are strings?

```python
a = "word"  # a stores the string "word"
b = 'word2'
```

Variables - Strings

What are strings?

```python
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
```

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
```

What are strings?

```python
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
d = 'abc'
```

What are strings?

```python
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
d = 'abc"   # error
```

Variables - Strings

What are strings?

```python
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
d = 'abc"   # error
```

Quotes

In Python you must use corresponding quotation marks for strings.

How do I put the symbols ' and " into a string?

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

```
a = "word\"" # a stores the string "word""
```

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

```
a = "word\"" # a stores the string "word""
b = 'word"'  # b stores the same string as a
```

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

```
a = "word\"" # a stores the string "word""
b = 'word"'  # b stores the same string as a
```

Same goes for single quotes ':

```
a = 'word\'' # a stores the string "word'"
b = "word'"  # b stores the same string as a
```

There are additional symbols in strings.

```python
a = "word\n" # \n represents the newline character
b = "word\t" # \t represents the tab character
```

Example:

```
a = "haha"
b = "hehe"
c = a + b
```

Example:

```
a = "haha"
b = "hehe"
c = a + b     # c stores the string "hahahehe"
```

Example:

```
a = "haha"
b = "hehe"
c = a + b      # c stores the string "hahahehe"
```

Concatenation of strings

You can concatenate (add) strings together with the addition symbol.

Contents
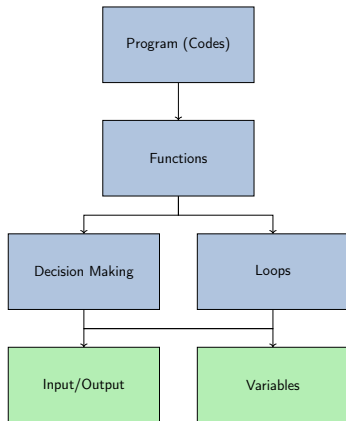
How do we print variables?

```
a = 5
print(a)
```

Variables in output - Revisiting the print() function

How do we print variables?

```
a = 5
print(a)        # 5
```

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)
```

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
```

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + 2)
```

How do we print variables?

```python
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + 2)    # 7
```

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + 2)    # 7
print(b + "a")
```

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + 2)    # 7
print(b + "a")  # hahaa
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + 2)    # 7
print(b + "a")  # hahaa
```

Calculation

We can calculate expressions inside the `print()` function.

How do we print variables?

```
a = 5
print(a)
```

How do we print variables?

```
a = 5
print(a)        # 5
```

How do we print variables?

```python
a = 5
print(a)        # 5
b = "haha"
print(a, b)
```

How do we print variables?

```python
a = 5
print(a)        # 5
b = "haha"
print(a, b)     # 5 haha
```

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(a, b)     # 5 haha
print(b, b)
```

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(a, b)     # 5 haha
print(b, b)     # haha haha
```

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(a, b)       # 5 haha
print(b, b)       # haha haha
```

The comma

Using `,` in `print()` would add a space in between the 2 items.

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
```

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5")
```

How do we print variables?

```
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + "5") # error
```

How do we print variables?

```python
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + "5")  # error
print(b + 2)
```

How do we print variables?

```
a = 5
print(a)       # 5
b = "haha"
print(b)       # haha
print(a + "5") # error
print(b + 2)   # error
```

How do we print variables?

```python
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + "5")  # error
print(b + 2)    # error
print(a + b)
```

How do we print variables?

```python
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + "5")  # error
print(b + 2)    # error
print(a + b)    # error
```

How do we print variables?

```python
a = 5
print(a)        # 5
b = "haha"
print(b)        # haha
print(a + "5")  # error
print(b + 2)    # error
print(a + b)    # error
```

Addition

You cannot use addition to print things of incompatible types.
`int` and `float` types are not incompatible because all `int` are converted to `float` if needed during operation.

How do we print variables?

```
a = 5
b = 32
c = 32.0
print(a * b)      # 160
print(a * c)      # 160.0
```

How do we print variables?

```
a = 5
b = 32
c = 32.0
print(a * b)    # 160
print(a * c)    # 160.0
```

Takeaway

`print()` function evaluates the expression inside the brackets first before actually printing.

Now go to your Number Guessing Game file's task 1.

All lines you have to do is marked with "# TODO:".
Try to finish the print() statements!

The End
Made in LaTeX
Last updated: 29 Mar 2024

Here are some additional content that we didn't have time to mention in the workshop.

More on `print()` function

In Python, the `print()` function automatically adds a new line after execution. We, however, can stop that.

The `end=` tag allows us to define the character added when `print()` is executed.

```python
print(5, end="")
print(4)
print("a", end="abc")
print("d", end=" ")
print("e")
```

More on `print()` function

In Python, the `print()` function automatically adds a new line after execution. We, however, can stop that.

The `end=` tag allows us to define the character added when `print()` is executed.

```python
print(5, end="")
print(4)
print("a", end="abc")
print("d", end=" ")
print("e")
# What is the output?
```

More on `print()` function

In Python, the `print()` function automatically adds a new line after execution. We, however, can stop that.

The `end=` tag allows us to define the character added when `print()` is executed.

```python
print(5, end="")
print(4)
print("a", end="abc")
print("d", end=" ")
print("e")
# What is the output?
# Output: 54
#         aabcd e
```

End of line

Remember to include a new line `\n` in the last line of a printed string. Else it may mess up the future outputs from other lines of the code or the computer terminal.

More on `print()` function

We mentioned that whenever , is used in `print()`, the items would be separated by a space.
This can actually be changed using the `sep=` tag.

More on `print()` function

We mentioned that whenever `,` is used in `print()`, the items would be separated by a space.

This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
```

More on `print()` function

We mentioned that whenever , is used in `print()`, the items would be separated by a space.

This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
>>> 100 100
    3
```

More on `print()` function

We mentioned that whenever , is used in `print()`, the items would be separated by a space.
This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
>>> 100 100
    3
>>> print("100", 100, sep="a", end="\n3\n")
```

We mentioned that whenever , is used in `print()`, the items would be separated by a space.

This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
>>> 100 100
    3
>>> print("100", 100, sep="a", end="\n3\n")
>>> 100a100
    3
```

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
```

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
>>> 5101520
>>> print(a, b, a + b, end="20\n", sep="")
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
>>> 5101520
>>> print(a, b, a + b, end="20\n", sep="")
>>> 5101520
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
>>> 5101520
>>> print(a, b, a + b, end="20\n", sep="")
>>> 5101520
```

Command Parameters

As long as you mark `sep` and `end` clearly **and** after the things you want to print, the ordering doesn't matter!