Functions and Variables

Click `Labs` > `launch` button

Jupyter notebook on Ed Lesson

side bar

notebook cells (code, text)

run (▶️ or `shift+enter` or `ctrl+enter`)

autocomplete / syntax highlighting

---

markdown syntax: https://www.markdownguide.org/basic-syntax/

Guide to Using Lab Notebook

in-class exercises

notes

study guide

to reset: `...` > `Reset to Scaffold`

1. Hello World

```
Hello, World!
```

Anatomy of `print()`

```
print("Hello, World")
```

- **Function**: predefined rules
  - `print()`
- **Argument**: input to the function
  - `"Hello, World"`
- **Side effect**: output of the function
  - print to the screen

Bugs 🐞

```
print("Hello world"
```

```
  Cell In[1], line 1
    print("hello world"
                       ^
SyntaxError: incomplete input
```

2. Hello to You

```
What's your name? John
Hello, John!
```

Anatomy of `input()`

```
answer = input("What's your name? ")
```

- **Function**: predefined rules
  - `input()`
- **Argument**: input to the function
  - "What's your name? "
- **Side effect**: output of the function
  - prompt the user and wait for input
- **Return values**: output of the function
  - user input
- **Variable**: box to store something
  - `answer`

Printing variable

```
answer = input("What's your name? ")
print("Hello, answer")
```

Joining strings and variables (`+`)

```
answer = input("What's your name? ")

# Hello with space
print("Hello " + answer)
```

Joining strings and variables (multiple arguments)

```
answer = input("What's your name? ")

# Hello without space
print("Hello", answer)
```

`help()` to look up function details

```
help(print)
```

```
Help on built-in function print in module builtins:

print(*args, sep=' ', end='\n', file=None, flush=False)
    Prints the values to a stream, or to sys.stdout by default.

    sep
      string inserted between values, default a space.
    end
      string appended after the last value, default a newline.
    file
      a file-like object (stream); defaults to the current sys.stdout.
    flush
      whether to forcibly flush the stream.
```

Or, you can refer to the documentation online:

https://docs.python.org/3/library/functions.html#print

Joining strings and variables (f-string)

```
answer = input("What's your name? ")

# f-string
print(f"Hello, {answer}")
```

Comments (`#`)

```
# + operator
print("Hello " + answer)

# multiple arguments
print("Hello", answer)

# f-string
print(f"Hello {answer}")
```

3. Personalized Introduction 🖥️

Requirements:

- Use `input()` function to prompt the user for their name and age.
- Store these values in variables.
- Use `print()` function and string formatting to display a message
  - "Hello, my name is xx. I am xx years old."
  - Replace xx with the user's name and age.

Expected Outputs:

```
What's your name? Emily
How old are you? 25
Hello, my name is Emily. I am 25 years old.
```

4. Uncooperative users

```
What is your name?     john
Hello, John

What is your name? jAnE doE
Hello, Jane Doe
```

String methods

https://docs.python.org/3/library/stdtypes.html#string-methods

strip()

```
answer = input("What's your name? ")

# call the strip method on the variable answer
answer = answer.strip()

print("Hello " + answer)
```

capitalize()

```
answer = input("What's your name? ")

# call the strip and capitalize methods on the variable answer
answer = answer.strip().capitalize()

# Chaining methods together
# answer is a string
# answer.strip() is a string

print("Hello " + answer)
```

`title()`

```
answer = input("What's your name? ")

answer = answer.strip().title()

print("Hello " + answer)
```

replace()

```
sentence = "I like apples, but I don't like green apples."
new_sentence = sentence.replace("apples", "oranges")
print(new_sentence)
```

split()

```
sentence = "I like apples, but I don't like green apples."
words = sentence.split()
print(words)
```

5. Hello Function

```
hello()
# Output: Hello, World!

hello("John")
# Output: Hello, John
```

`def` to define a function

```
# Define a function called hello
def hello():
    # Indentation matters!

    # Function body: print "Hello world"
    print("Hello world")

answer = input("What's your name? ")

# Call the function
hello()
```

Functions can take arguments

```
# Define a function called hello that takes an argument "to"
def hello(to):

    # Function body: print "Hello " and the argument
    print("Hello ", to)

answer = input("What's your name? ")

# Call the function with the variable answer as an argument
hello(answer)
```

Positional vs Keyword arguments

```
# positional arguments
hello("john")

# keyword arguments
hello(to="john")
```

Arguments with default values

```
# Argument "to" with a default value "world"
def hello(to="world"):

    # Function body: print "Hello " and the argument
    print("Hello ", to)

answer = input("What's your name? ")

# Call the function with the variable answer as an argument
hello(answer)

# Call the function without an argument
hello()
```

🤔 Arguments

```
def hello(a, b="Doe"):
    print("Hello", a, "and", b)
# 1
hello("John", "Doe")
# 2
hello("Doe", "John")
# 3
hello(b="Doe", a="John")
# 4
hello("John")
# 5
hello(b="John")
```

`main()` to organize program flow in one place

```
def main():
    # 1. ask the user for their name

    # 2. call hello() to say hello
```

```
# Write main first to define the program flow
def main():
    # 1. ask the user for their name
    answer = input("What's your name? ")
    # 2. call hello() to say hello
    hello()

# Then write hello
def hello():
    ...

# call main to start the program
main()
```

Scope

```
def main():
    answer = input("What's your name? ")
    hello()

def hello():
    print("Hello ", answer)

main()
```

Fixing scope by passing arguments

```
def main():
    answer = input("What's your name? ")
    hello(answer)   # pass the variable answer to the function

def hello(to):      # take the argument "to"
    print("Hello ", to)

main()
```

return

```
def main():
    answer = input("What's your name? ")
    message = hello_message(answer)

    print(message)

def hello_message(to="world"):
    msg = "Hello " + to

    return msg

main()
```

6. Personalized Introduction 2 🖥️

Requirements:

- Define a function `ask_name()` that prompts the user for their name using Python's `input()` function and returns the name.
- Define another function `ask_age()` that prompts the user for their age and returns the age.
- Define a function `introduce_message()` that takes name and age as parameters and returns a string in the format "Hello, my name is [name]. I am [age] years old."

Expected Outputs:

```
Name: Emily
Age: 25
Hello, my name is Emily. I am 25 years old.
```

7. Calculator

```
Enter a number: 5
Enter another number: 3
8
```

```
def calculator():
    x = input("Enter a number: ")
    y = input("Enter another number: ")
    z = x + y

    print(z)

calculator()
```

`int()` to convert string to integer

```
def calculator():
    x = input("Enter a number: ")
    y = input("Enter another number: ")
    z = int(x) + int(y)

    print(z)

calculator()
```

`type()` to check variable type

```
x = input("Enter a number: ")
type_x = type(x)
print(type_x)

y = int(x)
type_y = type(y)
print(type_y)
```

Which style do you prefer?

```
def calculator():
    x = input("Enter a number: ")
    y = input("Enter another number: ")
    z = int(x) + int(y)
    print(z)
```

vs.

```
def calculator():
    x = int(input("Enter a number: "))          # nest input inside int
    y = int(input("Enter another number: "))    # nest input inside int
    print(x+y)
```

vs.

```
def calculator():   # nest all the way
    print(int(input("Enter a number: ")) + int(input("Enter another number: ")))
```

`float()` to convert string to floating-point numbers

```
def calculator():
    x = input("Enter a number: ")
    y = input("Enter another number: ")

    z = float(x) + float(y)

    print(z)

calculator()
```

type conversion functions

- `int()`
- `float()`
- `str()`

...

round()

```
def calculator():
    x = input("Enter a number: ")
    y = input("Enter another number: ")

    z = float(x) / float(y)

    z = round(z, 2)

    print(z)

# try 2 and 3
calculator()

# Output: 0.67
```

8. Personalized Introduction 3 🖥️

Requirements:

- Define a function `ask_birthyear()` that prompts the user for their birth year and returns it.
- Define another function `calc_age()` that takes the birth year as an argument, calculates the age based on the current year, and returns it.
- Utilize the previously defined `ask_name()` and `introduce_message()` functions.
- Define a `main()` function that orchestrates the execution of these functions and prints the final introduction message.

Expected Outputs:

```
What's your name? Emily
What's your birth year? 1998
My name is Emily and I am 25 years old.
```

Takehome exercise 1

- Sign-up instructions: `Tools` > `DataCamp`
- **Use your `mail.mcgill.ca email` to sign up**
- Complete **"Introduction to Python"**
- Due next week before the class