Python Basics — Student Handout

Topic 1: Introduction to Computational Thinking

Goal: Understand what computational thinking is and why it matters.

- **Definition:** Breaking down problems into smaller steps that a computer can solve.
- **Key Concepts:**
  - **Decomposition:** Breaking a big problem into smaller ones.
  - **Pattern Recognition:** Identifying similarities.
  - **Abstraction:** Focusing on what's important.
  - **Algorithms:** Step-by-step instructions.

Mini Exercise:
List three steps you'd take to make a peanut butter sandwich.

Topic 2: Python as a Tool and Environment

Goal: Learn why Python is widely used and how to run Python code.

- Python is beginner-friendly and widely used in data science, AI, finance, and more.
- Runs on many platforms, including Codespaces and local machines.
- Ways to run Python: interactive shell, VS Code, Codespaces, or running `.py` files.

Mini Exercise:
Type the following in a Python interpreter:

```
print("Hello, world!")
```

Topic 3: Variables, Data Types, Arithmetic Operators, and Strings

Goal: Understand variables, primitive data types, arithmetic operations, and basic string handling.

Variables

- Store data for later use.
- Example:

```
name = "Alice"
age = 21
```

Data Types

- **int**, **float**, **str**, **bool**
- Check type with `type()`

Arithmetic Operators

Operator	Meaning	Example	Result
`+`	Addition	`3 + 2`	`5`
`-`	Subtraction	`7 - 4`	`3`
`*`	Multiplication	`5 * 6`	`30`
`/`	Division	`8 / 2`	`4.0`
`//`	Floor division	`7 // 2`	`3`
`%`	Modulus	`7 % 2`	`1`
`**`	Exponent	`2 ** 3`	`8`

Operator Precedence

1. Parentheses `()`
2. Exponent `**`
3. Multiplication/Division/Modulus `* / // %`
4. Addition/Subtraction `+ -`

Strings

- Strings are sequences of characters.
- Common methods:

```
msg = "Hello, World!"
```

```
print(msg.upper())   # 'HELLO, WORLD!'
print(msg.lower())   # 'hello, world!'
print(msg.capitalize())  # 'Hello, world!'
print(msg.replace('World', 'Python'))  # 'Hello, Python!'
print(msg.split(','))  # ['Hello', ' World!']
```

Mini Exercise:

1. Assign your name, age, and GPA to variables and print them.
2. Calculate `(10 + 2) * 5 / 2`.
3. Create a string variable and try at least three string methods.

Topic 4: Data Types, Data Structures, and Mutability

Goal: Learn about mutable and immutable types and basic Python data structures.

Mutable vs Immutable

- **Mutable:** objects that can be changed in place (list, dict, set)
- **Immutable:** objects that cannot be changed (int, float, str, tuple)
- **Important:** Mutability applies to the object, not the variable name.

```
numbers = [1, 2, 3]   # mutable object
numbers[0] = 10        # modifies the object in place
text = "hello"        # immutable object
# text[0] = 'H' would cause error
```

Data Structures

- **Lists:** ordered, mutable
- **Tuples:** ordered, immutable
- **Dictionaries:** key-value pairs, mutable
- **Sets:** unordered, unique items, mutable

```
fruits = ["apple", "banana"]
tuple_example = (1, 2, 3)
dict_example = {"a": 1, "b": 2}
set_example = {1, 2, 3}
```

Mini Exercise:

1. Create a list of three favorite movies and modify one.
2. Create a tuple of three numbers and try to modify an element (observe error).
3. Create a dictionary with two key-value pairs and update a value.

Topic 5: Input, Output, and File I/O

Goal: Learn how to interact with users and files.

User Input and Print

```
name = input("Enter your name: ")
print("Hello,", name)
```

File I/O

```
# Write to a file
with open("example.txt", "w") as f:
    f.write("Hello, file!\n")

# Read from a file
with open("example.txt", "r") as f:
    content = f.read()
    print(content)
```

Mini Exercise:

1. Ask the user for two numbers and print their sum.
2. Write a program to write user input to a file and then read it back.

Topic 6: Conditional Logic

Goal: Control program flow with conditions and boolean logic.

```
age = int(input("Enter your age: "))
if age >= 18:
    print("Adult")
elif age >= 13:
    print("Teenager")
else:
    print("Child")
```

- Boolean operators: `and`, `or`, `not`
- Comparison operators: `==`, `!=`, `<`, `>`, `<=`, `>=`

Mini Exercise:

1. Check if a number is even or odd.
2. Check if a number is within a specific range.
3. Combine conditions using `and`/`or`.

Topic 7: Loops and Control Statements

Goal: Repeat tasks efficiently and control loop behavior.

`for` Loop

```
for i in range(5):
    print(i)
```

`while` Loop

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Loop Control Statements

- `break` → exit the loop immediately
- `continue` → skip to the next iteration
- `pass` → placeholder, does nothing

```
for i in range(5):
    if i == 2:
        continue  # skip 2
    if i == 4:
        break     # exit loop
    print(i)
```

Mini Exercise:

1. Print numbers 0–10 but skip multiples of 3.
2. Stop printing when a number is greater than 8.

Topic 8: Functions and Mutability

Goal: Learn to define and use functions and understand how mutability affects arguments.

Defining Functions

```
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))
```

- Functions take parameters and can return values.
- Scope: local vs global variables.

Mutability in Functions

- **Primitives (immutable):** changes inside function do not affect original variable
- **Data structures (mutable):** changes inside function affect the original object

```
def change_number(n):
    n += 10

def change_list(lst):
    lst.append(4)

num = 5
change_number(num)
print(num)  # still 5

my_list = [1, 2, 3]
change_list(my_list)
print(my_list)  # [1, 2, 3, 4]
```

Mini Exercise

1. Write a function that takes a number and tries to increment it; observe the effect.
2. Write a function that takes a list and modifies it; observe the effect on the original list.

Topic 9: Comments, Spacing, and Brackets

Goal: Write readable and correctly formatted code.

- Single-line: `# comment`
- Multi-line: `""" comment """`
- Indentation matters in Python blocks
- Brackets:
  - `[]` → list or indexing
  - `()` → function calls or tuples
  - `{}` → dictionaries or sets

Mini Exercise:
Write a function that prints each element of a list with a comment explaining the action.