Python Basics — Instructor Guide

Topic 1: Introduction to Computational Thinking

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

Key Points to Discuss:

- Emphasize problem decomposition, pattern recognition, abstraction, and algorithms.
- Give real-life examples, like planning a trip or cooking a recipe.

Sample Discussion Prompts:

- "What steps would you take to solve a problem like making a sandwich?"
- "Can you identify patterns in daily routines?"

Sample Answers for Mini Exercise:

1. Open the peanut butter jar.
2. Spread peanut butter on bread.
3. Add jelly.
4. Put slices together.
5. Eat.

Topic 2: Python as a Tool and Environment

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

Instructor Notes:

- Highlight Python's simplicity and readability.
- Show examples of running code in different environments.

Sample Exercise Answer:

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

Expected Output: `Hello, world!`

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

Goal: Understand variables, primitive data types, arithmetic operations, and strings.

Instructor Notes:

- Emphasize variable assignment and types.
- Explain operator precedence.
- Demonstrate string indexing, slicing, and common methods.

Sample Exercise Answers:

1. Variables Example:

```
name = "Alice"
age = 21
gpa = 3.8
print(name, age, gpa)
```

2. Arithmetic Example:

```
result = (10 + 2) * 5 / 2
print(result)  # 30.0
```

3. String Methods Example:

```
word = "python"
print(word.upper())    # 'PYTHON'
print(word.capitalize())  # 'Python'
print(word.replace('thon','thon!'))  # 'python!'
```

Discussion Prompts:

- "What happens if you try to combine a string and int?"
- "Why does operator precedence matter?"

Topic 4: Data Types, Data Structures, and Mutability

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

Instructor Notes:

- Emphasize that mutability applies to objects, not variable names.
- Explain lists, tuples, dicts, sets, and when to use each.

Sample Exercise Answers:

1. List modification:

```
movies = ["Movie1", "Movie2", "Movie3"]
movies[0] = "New Movie"
print(movies)
```

2. Tuple modification (will fail):

```
t = (1, 2, 3)
# t[0] = 10  # Error
```

3. Dictionary update:

```
d = {"a":1, "b":2}
d["b"] = 20
print(d)
```

Discussion Prompts:

- "Why can we change a list inside a function but not a tuple?"
- "Give examples of when immutability is useful."

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

Goal: Learn how to interact with users and files.

Instructor Notes:

- Demonstrate `input()` and `print()`.
- Show file read/write with `with open()`.

Sample Exercise Answers:

1. User input sum:

```
x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
print("Sum:", x+y)
```

2. File I/O Example:

```
with open("myfile.txt", "w") as f:
    f.write("Hello\n")
with open("myfile.txt", "r") as f:
    print(f.read())
```

Discussion Prompts:

- "What happens if the file does not exist?"
- "Why use `with open()` instead of `open()` and `close()`?"

Topic 6: Conditional Logic

Goal: Control program flow with conditions and boolean logic.

Instructor Notes:

- Cover `if`, `elif`, `else` and boolean operators.
- Demonstrate combining multiple conditions.

Sample Exercise Answers:

1. Even/Odd:

```
num = 4
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

2. Range check:

```
num = 7
if 5 <= num <= 10:
    print("Within range")
```

3. Combining conditions:

```
x = 5
y = 10
if x > 0 and y > 5:
    print("Both conditions true")
```

Discussion Prompts:

- "How do `and` and `or` differ in evaluation?"
- "Give a real-world example of nested conditionals."

Topic 7: Loops and Control Statements

Goal: Repeat tasks efficiently and control loop behavior.

Instructor Notes:

- Demonstrate `for` and `while` loops.
- Explain `break`, `continue`, `pass`.

Sample Exercise Answers:

1. Skip multiples of 3:

```
for i in range(11):
    if i % 3 == 0:
        continue
    print(i)
```

2. Stop when number > 8:

```
for i in range(20):
    if i > 8:
        break
    print(i)
```

Discussion Prompts:

- "When would you use `continue` instead of an `if` statement?"
- "What happens if `break` is inside nested loops?"

Topic 8: Functions and Mutability

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

Instructor Notes:

- Explain function definition, parameters, return values.
- Demonstrate mutable vs immutable behavior inside functions.

Sample Exercise Answers:

1. Immutable argument:

```
def increment(n):
    n += 1
x = 5
increment(x)
print(x)  # still 5
```

2. Mutable argument:

```
def add_item(lst):
    lst.append(4)
my_list = [1,2,3]
add_item(my_list)
print(my_list)  # [1,2,3,4]
```

Discussion Prompts:

- "Why did the integer not change but the list did?"
- "How does understanding mutability affect your function design?"

Topic 9: Comments, Spacing, and Brackets

Goal: Write readable and correctly formatted code.

Instructor Notes:

- Explain single-line and multi-line comments.
- Emphasize indentation rules.
- Demonstrate bracket usage.

Sample Exercise Answer:

```
def print_elements(lst):
    # Loop through each element and print
    for item in lst:
        print(item)
```

Discussion Prompts:

- "Why is proper indentation critical in Python?"
- "Give an example of when multi-line comments are useful."
- Explain why some functions are called like int(num) and some like “some text”.upper()