CYS Programming with python

Session 3-4

Done By: Maryam Hajeb

01 Control Flow Statements

- Conditional Statements:
    - if-elif-else
    - match-case
- Iterative Statements:
    - for Loop
    - while Loop
- Transfer Statements : break - continue

controlflow.py

```python
x = int(input("Enter a number: "))
if x > 0:
    print("Positive")
elif x < 0:
    print("Negative")
else:
    print("Zero")
grade = 'A'
match grade:
    case 'A':
        print("Excellent")
    case 'B':
        print("Good")
    case _:
        print("Invalid grade")
```

controlflow.py

```python
for i in range(5):
    if i == 2:
        continue
    elif i == 4:
        break
    print(i)

count = 0
while count < 3:
    print("Count:", count)
    count += 1
```

02

Functions

- A function is a reusable block of code designed to perform a specific task.
- It improves modularity, readability, and reduces repetition.
- Syntax:

```
def function_name(parameters):
        # function body
        return result
```

- Types of functions: Built-In function , User-Defined functions

Built-in Functions

- Python provides many built-in functions ready to use:

Function	Description	Example
abs()	Returns absolute value	abs(-10)
bool()	Converts to Boolean	bool(0)
help()	Displays help/documentation	help(print)
id()	Returns object’s unique ID	id(x)
int()	Converts value to integer	int("5")
max()	Returns largest item	max(1,2,3)
min()	Returns smallest item	min(1,2,3)

User-Defined Functions

- You can create your own functions using the def keyword.
- Example:

```
def greet(name):
        print("Hello,", name)
greet("Ali")

def add(x, y):
        return x + y
result = add(5, 10)
print("Sum =", result)
```

Function Arguments

- Python supports different types of function arguments:
  - **Default Arguments**: Provide a default value if none is given.
  - **Positional Arguments**: Values are assigned based on position.
  - **Keyword Arguments**: Values are assigned by parameter name.
  - **Arbitrary Positional Arguments** (*args): Accepts multiple arguments as a tuple.
  - **Arbitrary Keyword Arguments** (**kwargs): Accepts key-value pairs as a dictionary.

arguments.py

```
def MyFunc_Default(empname, department='Research'):
    print("Employee:", empname, "| Department:",
department)

MyFunc_Default("Sara")
MyFunc_Default("Omar", "HR")

def newFunction(name, age):
    print(name, "is", age, "years old.")

newFunction('Sam', 12)
```

arguments.py

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```python
def myFunction(name, age):
    print(name, "is", age, "years old.")

myFunction(age=12, name='Sam')

def show_subjects(*subjects):
    print("Subjects are:")
    for sub in subjects:
        print("-", sub)
show_subjects("Math", "AI", "Statistics")
```

arguments.py

```python
def show_info(**info):
    for key, value in info.items():
        print(key, ":", value)

show_info(name="Tom", dept="Data Science", year=2025)
```

Recursive Functions

- A function that calls itself to solve smaller subproblems.
- Example: Factorial

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)

print("Factorial of 5:", factorial(5))
```

Lambda Functions

- A lambda function is a small, unnamed function written in one line using the keyword lambda.
- They are often used when a function is needed only once – typically with functions like filter(), map(), and reduce().
- Syntax: lambda arguments: expression
- Example:

```
print((lambda x: x * x)(4))
square = lambda x: x * x
print(square(5))
```

Lambda Functions

- Lambda with filter():
    - Used to filter elements from a sequence (like a list) that meet a certain condition.
    - It returns only those elements for which the function returns True.

- Lambda with map():
    - Used to apply a function to every element in a sequence.
    - It transforms all elements according to the given function.

lambda.py

```
1   print("X * Y = ", (lambda x,y:x*y),(5,2))
2
3   square = lambda x: x * x
4   print(square(5))
5
6   numbers = [1, 2, 3, 4, 5, 6]
7   even = list(filter(lambda x: x % 2 == 0, numbers))
8   print("Even numbers:", even)
9
10  numbers = [1, 2, 3, 4]
11  squares = list(map(lambda x: x ** 2, numbers))
12  print("Squares:", squares)
13
14
```

03 Modules

- A module is a Python file (.py) that contains functions, classes, or variables you can import and reuse in other programs.
- Type of Modules:
    - Built-in Modules: math, random, datetime, etc.
    - User-Defined Module:
        . Create a .py file.
        . Add custom variables and functions in it.
        . Import it in the file you want to use it.

Built-in-module.py

```
#Importing a Module:
import math
print(math.sqrt(16))
#Importing Specific Functions:
from math import sqrt, pow
print(sqrt(25))
#Importing Multiple Modules:
import math, random
print(random.randint(1, 10))
#Renaming a Module:
import math as m
print(m.pi)
#Importing All Functions:
from math import *
print(floor(9.8))
```

04

Packages

- A package is a collection of modules organized in directories.
- Creating a Package:
- - Create a folder.
- - Add modules(.py files) in the folder.
- - Add function to the modules.
- - import the modules from the package.

Thanks!

Do you have any questions?