الجامعة الدولية للعلوم والنهضة

كلية الهندسة المعلوماتية

برمجة نصية (بايثون) – عملي

إعداد:

أ.محمد جراد أ.خالد الإسماعيل

1- اكتب دالة لتنفيذ عمليات متعددة على الأعداد باستخدام العامل * (*args)

```
def process_numbers(*numbers,operation):
    result = numbers[0]

    match operation:
        case "+":
            for i in numbers[1:]:
                result += i
        case "-":
            for i in numbers[1:]:
                result -= i
        case "*":
            for i in numbers[1:]:
                result *= i
        case "/":
            for i in numbers[1:]:
                if i != 0:
                    result /= i
                else:
                    return "Error: Division by zero is not allowed"

    return f"The result is: {result}"

print(process_numbers(8,4,3,operation="-"))
print(process_numbers(8,0,operation="/"))
```

2- اكتب دالة لإنشاء تقرير باستخدام العامل ** (**kwargs)

```
def generate_report(title, **reportInfo):
    print(f"Report Title: {title}")
    for key,value in reportInfo.items():
        print(f"{key} : {value}")

generate_report("Financial Report", author="Muhammad", date="August 2024",
notes="Annual financial report.",)
generate_report("Financial Report 2", author="Ahmad")
```

3- اكتب برنامج لحساب الربح الصافي من عمليات البيع لعدة منتجات

```
total_sales = 0 # تعريف المتغير العالمي

def record_sale(*amounts):
    global total_sales
    total_sales += sum(amounts)

def calculate_profit(cost): # cost التكلفة الإجمالية
    profit = total_sales - cost
    print(f"Total profit: {profit}")

record_sale(120.50, 75.00, 78.00, 10.00, 132.00)  # تسجيل عمليات بيع بقيم متعددة
calculate_profit(cost=100) # حساب الربح الصافي
```

4- اكتب دالة لحساب الأس لعدد ما باستخدام العودية

```
def power(number, exp):
    if exp == 0:
        return 1
    else:
        return number * power(number, exp - 1)

print(power(2, 3))
```

number يمثل الرقم الذي نريد حساب قوته، و exp يمثل الأس، إذا كان الأس exp يساوي 0، فإن قيمة الدالة تكون 1 مباشرةً، لأن أي رقم مرفوع إلى قوة 0 يساوي 1.

إذا كان الأس ليس 0، فإن الدالة تقوم باستدعاء نفسها مرة أخرى مع تقليل قيمة الأس بمقدار 1، وتضرب الرقم (number) في النتيجة التي ستعيدها الدالة من الاستدعاء التالي. هذه العملية تستمر حتى يصبح الأس يساوي 0.

```
2^3 = 2 * 2^2
2^2 = 2 * 2^1
2^1 = 2 * 2^0
2^0 = 1
```

عند استدعاء power(2, 3) ، يتم تنفيذ العملية التالية:

power(2, 3) يستدعي power(2, 2)

power(2, 2) يستدعي power(2, 1)

power(2, 1) يستدعي power(2, 0)

power(2, 0) يعيد 1 لأن الأس أصبح 0

بعد ذلك:

power(2, 1) يعيد 2 * 1 = 2

power(2, 2) يعيد 2 * 2 = 4

power(2, 3) يعيد 2 * 4 = 8

5- اكتب دالة تقوم بتحليل عدد صحيح إلى عوامله الأولية باستخدام العودية

```python
def prime_factors(n, factor=2):
    if n <= 1:
        return []
    elif n % factor == 0:
        return [factor] + prime_factors(n // factor, factor)
    else:
        return prime_factors(n, factor + 1)

print(prime_factors(48)) # [2, 2, 2, 2, 3]
print(prime_factors(42)) # [2, 3, 7]
```

48	2
24	2
12	2
6	2
3	3
1	

42	2
21	3
7	7
1	

6- اكتب دالة لتحويل عدد ما من النظام العشري الى النظام الثنائي باستخدام العودية

```python
def to_binary(number):
    if number == 0:
        return ""

    remainder = number % 2

    return to_binary(number // 2) + str(remainder)

print(to_binary(13)) # 1101
```

7- اكتب دالة لحساب محيط ومساحة الدائرة

```
import math
def circle_area_and_circumference():
    radius = float(input("Enter radius: "))
    area = math.pi * math.pow(radius, 2)
    circumference = 2 * math.pi * radius
    return area, circumference

print(circle_area_and_circumference())
```

8- اكتب دالة لحساب المعادلة التالية:

$$\cos(\theta) - \sin(\theta) + \sqrt{x^2 + 1} + {}^{x}e = y$$

```
import math
def calculate_expression(x, theta_degrees):
    # تحويل الزاوية من درجات إلى راديان
    theta_radians = math.radians(theta_degrees)

    sin_theta = math.sin(theta_radians)
    cos_theta = math.cos(theta_radians)

    # حساب e^x
    e = math.pow(math.e,x)
    # أو
    # e = math.exp(x)

    # حساب الجذر التربيعي
    sqrt = math.sqrt(x**2 + 1)

    # حساب المعادلة
    result = cos_theta - sin_theta + sqrt + e
    return f"result: {result:.3f}"

print(calculate_expression(10,30))
```

9- اكتب برنامج يقبل قائمة مختلطة من النصوص والاعداد وقم بتنفيذ ما يلي:

- تحويل الأرقام النصية الى اعداد صحيحة 3 <== "3"
- فرز الأعداد الزوجية بقائمة والاعداد الفردية بقائمة والنصوص بقائمة أخرى
- حذف العدد صفر من قائمة الأعداد الزوجية
- دمج الأعداد الفردية والزوجية معا وفرز العناصر تنازلياً
- إيجاد أطول سلسلة نصية في قائمة النصوص

```
def process_on_lists(mixed_list):
    even_numbers = []
    odd_numbers = []
    strings = []
    # تحويل الأرقام النصية إلى أعداد صحيحة
    for item in mixed_list:
        if isinstance(item, str) and item.isdigit():
            item = int(item)
        if isinstance(item, int):
            if item % 2 == 0:
                even_numbers.append(item)
            else:
                odd_numbers.append(item)
        elif isinstance(item, str):
            strings.append(item)
```

```
    # إزالة الرقم صفر من الأعداد الزوجية
    even_numbers = [num for num in even_numbers if num != 0]
    # دمج الأعداد الفردية والأعداد الزوجية غير الصفرية
    numbers_without_zero = even_numbers + odd_numbers
    # حذف النصوص ذات المحرف الواحد
    strings = [s for s in strings if not len(s) <= 1]
    # إيجاد أطول اسم
    longest_name = max(strings, key=len)
    # فرز الأعداد تنازلياً
    numbers_without_zero.sort(reverse=True)

    print("Even numbers: ", even_numbers)
    print("Odd numbers: ", odd_numbers)
    print("Numbers without zero: ", numbers_without_zero)
    print("Names and strings: ", strings)
    print("Longest name: ", longest_name)

mixed_list = [0, 'a', 12, 'Muhammad', 3, 'Ali', '10', '30', 0, "Ahmad" , 11 ,'' ]
process_on_lists(mixed_list)
```

❖ نشاط : ما هو خرج عمليات التقطيع الاتية المطبقة على السلسلة النصية التالية: text = 'ABCDEFGHI'

```
print(text[:5]) , print(text[-4:]) , print(text[7:]) ,
print(text[2:10]) , print(text[::-1]) , print(text[::2])
```

10- لديك قائمة من الـtuples، حيث يحتوي كل tuple على اسم الطالب وعمره ومعدله الدراسي، قم بتنفيذ ما يلي:

- استخراج قائمة بأسماء الطلاب
- إيجاد الطالب الذي لديه أعلى معدل
- إيجاد الطالب الأصغر عمراً
- إيجاد متوسط معدلات جميع الطلاب

```
students = [
    ("Ahmad", 20, 88.5),
    ("Muhammad", 22, 92.3),
    ("Ali", 19, 85),
    ("Hasan", 21, 95.2),
    ("Omar", 23, 90.4)
]
#  استخراج قائمة بأسماء الطلاب
student_names = [student[0] for student in students]
print("List of student names:", student_names)

#  إيجاد الطالب الذي لديه أعلى معدل دراسي
top_student = max(students, key=lambda x: x[2])
print(f"Student with the highest GPA: {top_student[0]}, GPA: {top_student[2]}")

#  إيجاد الطالب الأصغر عمرًا
youngest_student = min(students, key=lambda x: x[1])
print(f"Youngest student: {youngest_student[0]}, Age: {youngest_student[1]}")

#  إيجاد متوسط معدلات جميع الطلاب
average_gpa = sum(gpa for name, age, gpa in students) / len(students)
print(f"Average GPA of all students: {average_gpa:.2f}")
```

✓ في بايثون، عند التعامل مع الـ tuples داخل حلقات أو تعبيرات توليدية، يمكننا استخدام تفكيك الـ tuple

(tuple unpacking) للحصول على القيم الفردية التي يحتويها الـ tuple

```
student = ("Ahmad", 20, 88.5)
name, age, gpa = student
print(name,age,gpa) # Ahmad 20 88.5
```