الجامعة الدولية للعلوم والنهضة

كلية الهندسة المعلوماتية

برمجة نصية (بايثون) – عملي

إعداد:

أ. محمد جراد

أ. خالد الإسماعيل

1. اكتب برنامج يحسب عدد المحارف الصغيرة والمحارف الكبيرة لسلسلة نصية

```
def process_text(text:str):
    chars_lower = 0; chars_capital = 0
    for i in text:
        if i.islower():
            chars_lower+=1
        else:
            chars_capital+=1
    return f"count chars lower: {chars_lower}\ncount chars capital: {chars_capital}"
print(process_text('Muhammad'))
```

2.اكتب برنامج يطلب من المستخدم إدخال سلسلة نصية وإدخال كلمة معينة لحذفها من هذه السلسلة.

```
text = input("Enter a text: ") # 'I Like Python'
word_to_delete = input("Enter a word to deletion: ")
words = text.split() # ['I', 'Like', 'Pyhton']
isFound = False
for i in words:
    if i == word_to_delete:
        words.remove(i)
        isFound = True
if isFound:
    text = ' '.join(words)
    print(f"Text after deletion word: {text}")
else:
    print("Word is not found in text")
```

❖ نشاط: اكتب برنامج لإضافة "ing" في نهاية سلسلة معطاة (يجب أن يكون طولها 3 أحرف على الأقل). إذا كانت السلسلة المعطاة تنتهي ب "ing" فأضف "ly" بدلاً من ذلك.

```
def add_ing(text:str):
    if len(text) > 2:
        if text.endswith(('ing')) :
            text += 'ly'
        else:
            text += 'ing'
    return text

print(add_ing('ab'))      # Output: 'ab'
print(add_ing('abc'))     # Output: 'abcing'
print(add_ing('string'))  # Output: 'stringly'
```

4.اكتب برنامج لإزالة تكرارات الكلمات من سلسلة نصية، مع الحفاظ على ترتيب ظهور الكلمات الأصلي.

```
def remove_duplicate_words(input_string):
    words = input_string.split()
    # مجموعة لتتبع الكلمات التي تمت رؤيتها
    seen_words = set()
    #   قائمة لحفظ الكلمات بدون تكرار
    result = []

    for word in words:
        # استخدام lower() لجعل التحقق غير حساس لحالة الأحرف
        if word.lower() not in seen_words:
            seen_words.add(word.lower())
            result.append(word)

    return ' '.join(result)
text = "This is a test. This test is only a test."
print(remove_duplicate_words(text))
```

5. لديك نص يمثل كلمة مرور، والمطلوب هو التحقق من مدى قوتها وفقًا لمعايير معينة مثل الطول، وجود أحرف كبيرة وصغيرة، وأرقام، ورموز خاصة.

```
def is_strong_password():
    password =input("Enter the password : ")
    if len(password) < 8:
        return "password is very week"
    has_upper = any(char.isupper() for char in password)
    has_lower = any(char.islower() for char in password)
    has_digit = any(char.isdigit() for char in password)
    has_special =any(not char.isalnum() for char in password)
    strong = has_upper and has_lower and has_digit and has_special
    if strong:
        return "passord is strong"
    else:
        return "password is week"

print(is_strong_password())
```

6.لديك قاموس يحتوي على أسماء الطلاب ودرجاتهم. قم بتصنيف الطلاب إلى ثلاثة فئات: "ممتاز" (90-100)، "جيد جدًا" (80-89)، "جيد" (70-79)، و"راسب" (أقل من 70).

```
def categorize_students(grades):
    categories = {
        "Excellent": [],
        "Very good": [],
        "Good": [],
        "Fail": []
    }
    for student, grade in grades.items():
        if grade >= 90:
            categories["Excellent"].append(student)
        elif grade >= 80:
            categories["Very good"].append(student)
        elif grade >= 70:
            categories["Good"].append(student)
        else:
            categories["Fail"].append(student)
    return categories

students_grades = {
    "Ahmad": 95,
    "Hasan": 85,
    "Ali": 72,
    "Sara": 60,
    "Khaled": 88,
    }
print(categorize_students(students_grades))
```

7.اكتب برنامج لحساب عدد الأحرف (بدون الفراغات) في سلسلة نصية (استخدام القاموس لتخزين الحرف وعدد تكراره)

```
def char_frequency(text):
    freq_dict = {}
    for n in text:
        if n != " ":
            if n in freq_dict:
                freq_dict[n] += 1
            else:
                freq_dict[n] = 1
    return freq_dict

print(char_frequency('The third session in Python'))
```

8. لديك قاموس يحتوي على أسماء الموظفين وأجورهم، والمطلوب هو العثور على الموظف ذو الأجر الأعلى والموظفين الذين أجورهم أكثر من 2000.

```
employee_salaries = {
    "Ahmad": 2750,
    "Hasan": 1500,
    "Ali": 2000,
    "Muhammad": 2300
}

top_employee = max(employee_salaries, key=employee_salaries.get)
highest_salary = employee_salaries[top_employee]
# البحث عن الموظفين الذين يتقاضون أكثر من 2000
employees_above_2000 = {k: v for k, v in employee_salaries.items() if v > 2000}

print(f"Top: {top_employee} with a salary of {highest_salary}.")
print(f"Employees earning more than 2000: {employees_above_2000}.")
```

9. اكتب برنامجاً يقوم بحساب متوسط درجات طالب حسب اسمه.

```
students_grades = [
    {"name": "Muhammad", "age": 18, "grades": [90, 85, 92]},
    {"name": "Ali", "age": 17, "grades": [88, 91, 87]},
    {"name": "Hasan", "age": 19, "grades": [90, 90, 90]}
]
def get_avg_grade(students_grades, name):
    total_grades = 0
    for student in students_grades:
        if student["name"] == name:
            for grade in student["grades"]:
                total_grades += grade
            return f"{(total_grades / len(student["grades"])):.2f}"
    return f"{name} is not founded"

print(get_avg_grade(students_grades, "Muhammad"))
print(get_avg_grade(students_grades, "Ali"))
print(get_avg_grade(students_grades, "Hasan"))
print(get_avg_grade(students_grades, "Samer"))
```