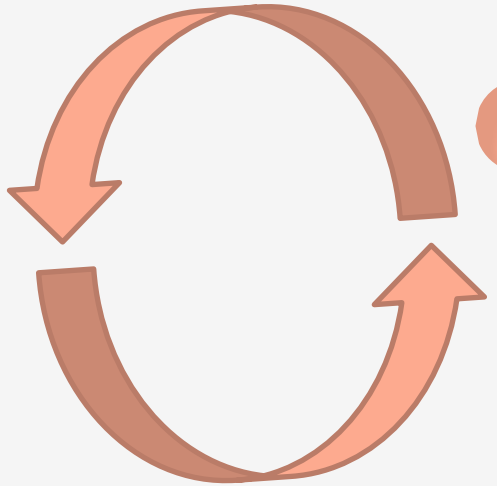
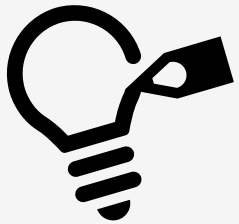
Loops



Lecture 14

Week 7



while loop

- Repetition using a while loop

New terminology:

- a pre-test loop is one whose condition must be True before it iterates
- It demonstrates zero-trip behaviour if the condition is never true

Strings in while loops

```
4   word = "hello"
5   i = 0
6   while i < len(word):
7       print(f"{i}:{word[i]}")
8       i = i + 1
```

- How many times does this loop iterate?
- What is printed in each print statement?
- Try using <= instead of <

Example: counting e’s

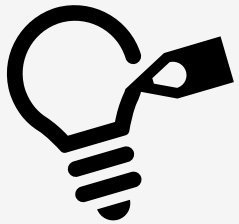
- Let’s write a program that calculates the number of e characters in a string.

```
users_input = input("Please type a word/phrase >>> ")
number_of_chars = len(users_input)

i = 0
count = 0

while i < len(users_input):
        if word[i]== "e" or word[i] == "E":
                count += 1
        i = i + 1

print(f"There are{count} e's in {word}” )
```



while loop

- Accumulators are variables that store the sum of a series of values

- We can iterate through a string, starting at 0 through to len(string), using [index] to access the characters within the string

augmented operators

- In many assignment statements, the variable on the left side of the = operator also appears on the right side of the = operator e.g. x = x + 1

- Augmented assignment operators special set of operators designed for this type of job

augmented operators

Operator	Example Usage	Equivalent To
+=	x += 5	x = x + 5
-=	y -= 2	y = y - 2
*=	z *= 10	z = z * 10
/=	a /= b	a = a / b
%=	c %= 3	c = c % 3
//=	x //= 3	x = x // 3
**=	y **= 2	y = y**2

Input validation

We often have to deal with mistakes made by the user

For example

- what if we ask for an age but the user enters “Fred”?
- Or if we ask for an age and the user enters “-3”?

If the user enters an invalid input we can ask again….

Input validation

```
age = int(input("Age: "))
if age < 0:
      age = int(input("Age: "))
```

- Check once and ask again....

Input validation

```
age = int(input("Age: "))
while age < 0:
      age = int(input("Age: "))
```

- Loop as long as the input is invalid

Input validation example

- Loop as long as a name is 2 characters or less

```
MIN_LENGTH_NAME = 2
name = input("What is your name? ")
while len(name) <= MIN_LENGTH_NAME:
    name = input("What is your name? ")
```

Input validation: do it yourself

- ✓ Ask the user to enter the grade they received on an exam
- ✓ Validate that the grade entered is between 0 and 100 i.e. it is a valid grade
- ✓ Once they enter a valid grade print it out

Input validation: do it yourself

- ✓ Ask the user to enter their name
- ✓ Validate that the name only contains alphabetic characters, if it does not ask the user to enter their name again
- ✓ Once they enter a valid name print it out

User mistakes – crashing code

```
1    # Author: Alison
2    # Purpose: dealing with input errors and exceptions
3
4    age = int(input("What is your age: "))
5
```

```
What is your age: Hello
Traceback (most recent call last):
  File "C:/Users/Alison.OShea/OneDrive - Munster Technological University/Documents 1/SOFT6018/PythonFiles/input_validation.py", line 4, in <module>
    age = int(input("What is your age: "))
ValueError: invalid literal for int() with base 10: 'Hello'
```

User mistakes – controlled crash

```
1   # Author: Alison
2   # Purpose: dealing with input errors and exceptions
3
4   try:
5       age = int(input("What is your age: "))
6   except ValueError:
7       print("Age should be a positive whole number.")
```

Controlled crash in a loop

```
# Author: Alison
# Purpose: dealing with input errors and exceptions

age = -1  # forces python into our loop

while age < 0:
    try:
        age = int(input("What is your age: "))
    except ValueError:
        print("Age should be a positive whole number.")
```